PROACTIVE SCHEMES: ADVERSARIAL ATTACKS FOR SOCIAL GOOD

By

Vishal Asnani

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science—Doctor of Philosophy

2025

ABSTRACT

Adversarial attacks in computer vision typically exploit vulnerabilities in deep learning models, generating deceptive inputs that can lead AI systems to incorrect decisions. However, proactive schemes approaches designed to embed purposeful signals into visual data can serve as “adversarial attacks for social good,” harnessing similar principles to enhance the robustness, security, and interpretability of AI systems. This research explores application of proactive schemes in computer vision, diverging from conventional passive methods by embedding auxiliary signals known as "templates" into input data, fundamentally improving model performance, attribution capabilities, and detection accuracy across diverse tasks. This includes novel techniques for image manipulation detection and localization, which introduce learned templates to accurately identify and pinpoint alterations made by multiple, previously unseen Generative Models (GMs). The Manipulation Localization Proactive scheme (MaLP), for example, not only detects but also localizes specific pixel changes caused by manipulations, showing resilient performance across a broad range of GMs. Extending this approach, the Proactive Object Detection (PrObeD) scheme utilizes encoder-decoder architectures to embed task-specific templates within images, enhancing the efficacy of object detectors, even under challenging conditions like camouflaged environments. This research further expands proactive schemes into generative models and video analysis, enabling attribution and action detection solutions. ProMark, for instance, introduces a novel attribution framework by embedding imperceptible watermarks within training data, allowing generated images to be traced back to specific training concepts—such as objects, motifs, or styles—while preserving image quality. Building on ProMark, CustomMark offers selective and efficient concept attribution, allowing artists to opt into watermarking specific styles and easily add new styles over time, without the need to retrain the entire model. Inspired by the proactive structure of PrObeD for 2D object detection, PiVoT introduces a video-based proactive wrapper that enhance action recognition and spatio-temporal action detection. By integrating action-specific templates through a template-enhanced Low-Rank Adaptation (LoRA) framework, PiVoT seamlessly augments various action detectors, preserving computational efficiency while significantly boosting detection performance.

Lastly, the thesis presents a model parsing framework that estimates "fingerprints" for the generative models, extracting unique characteristics from generated images to predict the architecture and loss functions of underlying networks—a particularly valuable tool for deepfake detection and model attribution. Collectively, these proactive schemes offer significant advancements over passive methods, establishing robust, accurate, and generalizable solutions for diverse computer vision challenges. By addressing key issues related to the different vision applications caused by conventional passive approaches, this research lays the groundwork for a future where proactive frameworks can improve AI-driven applications.

This thesis is dedicated to my Father and Mother. Thank you for always being there for me and believing in me.

ACKNOWLEDGMENTS

This PhD journey has been an incredible and transformative experience, one that would not have been possible without the support of many individuals. First and foremost, I extend my deepest gratitude to my advisor, Dr. Xiaoming Liu, for his mentorship, guidance, and patience throughout my PhD. His support and belief in me, especially during times when I doubted myself, have been invaluable. He took a chance on me and continuously pushed me to do better at every step, ensuring I stayed on track even when things got difficult. His insights and encouragement have helped shape my research and given me the confidence to take on challenging projects. Without his support, I would not have achieved the progress I have made in my PhD.

Among those who shaped my PhD, Dr. Xi Yin holds a special place. She entered my life when I was struggling to find direction, becoming not just a brilliant mentor but a guiding force. Beyond research, she taught me how to navigate the PhD journey itself, offering both intellectual guidance and emotional support during critical moments. Involved in most of my projects, she pushed me toward excellence while ensuring I never felt lost. Her patience, kindness, and belief in me have been invaluable, and I will always be grateful for her support.

I am also immensely grateful to my committee members, Dr. Arun Ross and Dr. Yu Kong, and to my collaborator Dr. Sijia Liu, whose expertise and thoughtful insights have been instrumental in shaping my research. Their guidance has gone far beyond formal meetings-they have continuously challenged me to think critically, refine my methodologies, and push the boundaries of my work. Their feedback has not only strengthened the technical aspects of my dissertation but has also encouraged me to explore new perspectives and approaches that I would not have considered on my own.

I never imagined that a single email with Dr. John Collomosse would shape my PhD journey, leading to two summer internships under him and Dr. Shruti Agarwal, and ultimately to my full-time position with the same team. Dr. Collomosse provided the perfect mix of guidance and independence, helping me bridge research with real-world applications. If he opened the door, Dr. Agarwal made sure I thrived. Her technical expertise, hands-on approach, and clear, practical

advice kept our projects on track and made problem-solving seamless. Beyond work, she ensured our time was filled with memorable experiences, from Indian restaurant outings to movie nights and fun gatherings, making my internships truly special.

I am deeply grateful to Dr. Tal Hassner for his support, especially during a critical moment when a last-minute conflict with Meta's legal team jeopardized a key paper submission. With just a week left, he went above and beyond to push through approvals, and thanks to his and Dr. Xi Yin's efforts, we secured approval two days before the deadline, ensuring the paper's submission and eventual publication. Beyond this, both Dr. Hassner and Dr. Yin have been invaluable mentors, offering continuous guidance, feedback, and encouragement, shaping my growth as a researcher.

I also want to acknowledge my fellow members of CVLab—Andrew, Yiyang, Abhinav, Feng, Shengjie, Xiao, Zhiyuan, Girish, Jie, Zhizhong, Zhihao, Minchul, Dingqiang, Zhang, Masa, Yaojie, Garrick, Amin, Luan, and Morteza, whose stimulating discussions, and unwavering support have made this journey all the more enjoyable and intellectually fulfilling. The lab has been more than just a workplace; it has been a community where I have grown both as a researcher and as a person.

Beyond academia, I owe everything to my family, who have been my pillars of strength. This PhD is dedicated to my father (Shyam Lal Asnani) and mother (Jaya Asnani), whose unconditional love, sacrifices, and encouragement have shaped the person I am today. My mother is not here to see me achieve this milestone, having lost her to COVID, but I know she would be proud of me. Her belief in my dreams gave me the resilience to push forward, even in the hardest moments, and her absence is felt deeply in this achievement. I am equally grateful to my sisters, Neetu and Deepika Didi, and my nephew and nieces—Mannan, Dimple, Anushka, and Nyysa—for their constant support, love, and for always reminding me of the joys of life beyond research. Their presence has been a source of strength, and I carry my mother's love with me as I reach this milestone.

To my wife, Nikita, thank you for being my anchor through this journey. Your love, patience, and unwavering faith in me have been my greatest source of motivation. Our story has unfolded alongside this PhD journey—from the moment I first met you at Lansing airport to the day I proposed and eventually married you. Through the highs and lows, from late-night research struggles to the

small moments of joy, you have been my greatest companion. I am beyond grateful to have you by my side, making every challenge easier and every milestone even more meaningful.

Beyond my family, my friends-my second family-have been an integral part of my PhD journey, bringing laughter and encouragement into my life. Everyone has played a role in some way, shaping this experience into something far more meaningful than just academic work.

It all began with Ashish and Himanshu, my school friends who have been constants in my life, keeping me grounded no matter how much time passed. Then came my bachelor's years, where Manu, Yalaj, Mayank, Amartya, and Aman made every challenge easier with their support, whether through late-night conversations, shared struggles, or moments of pure fun. During my master's years, I found another incredible circle with Thanish, Ahamad, Saloni, Navya, Abhishek, and Snehal. Beyond academics, these years were filled with shared courses, endless hangouts, and game nights—especially during COVID, making some of my best memories despite the challenges.

Then came the introduction of the Desi Boys group, where I met some of the most amazing people: Bharat, Hitesh, Abubakr, Sai, Ankit, Siddharth, and Abhiroop. Friday hangouts became a tradition, filled with game nights and what we called "restaurant exploration"- which, in reality, was just revisiting the one and only chosen restaurant over and over again. These friendships made my PhD experience feel so much lighter, bringing moments of fun that balanced out the intensity of research.

Among all the people, meeting Nisha was truly special. She quickly became one of my best friends, someone I could talk to about anything and everything. Our drives were never just about getting coffee—they turned into adventures where we'd end up two hours away at a beach, completely unplanned. She was always there, through the ups and downs, making life in EL (East Lansing) all the more exciting. Then there was Nidhi, one of the closest friends I found in East Lansing. Our drives, our shared love for coffee, and our endless conversations made for some of my best moments. From convincing Nikita about me to our many hangouts, she was always there. Nothing hit me harder than when she left EL, and her absence was deeply felt.

Through this journey, I built a bond that felt like home with Ishita and Aditya. Whether it was

potluck nights, coffee hunts, or simply being there for each other, we always found time despite our packed schedules. Exploring countless coffee places together still feels like an achievement in itself. I was also fortunate to meet Konika and Mudita, whose warmth and friendship made everyday moments more enjoyable. Mudita's dad, with his kindness and wisdom, felt like family, always offering encouragement and support that meant a lot to me. Through Nikita, I got to know Gauri, Gaurav, Shruti, Raj, and Nihar, who started as her roommates but soon became close friends. From trips and hangouts to visiting them in Chicago and South Bend, every moment together strengthened our bond. Whether it was sharing meals, planning getaways, or simply catching up, their presence made my PhD years even more fulfilling.

Finally, in the later part of my PhD, I found a group of people who became close to my heart in no time—Nabasmita, Ritam, Devika, Soni, Deepak, and Ritwik. From celebrating birthdays to planning trips, this group made my last phase of PhD truly special (special shout-out to Nabasmita's therapy sessions and Devika's chai). The friendships I formed with them in such a short time feel just as deep and meaningful as the ones that have been with me for years. Each of these friendships has added something irreplaceable to my PhD journey. Beyond the research, papers, and long nights in the lab, it is these people who made this experience worthwhile.

In meeting all of these wonderful people along my journey, there was one constant that remained close to my heart-"VLHALA," my first car. She was more than just a vehicle; she was my companion through every phase of this PhD. She has seen me at my best and my worst, from moments of pure joy to times when I broke down in frustration. She was there for the late-night drives that helped clear my mind, for the spontaneous road trips that brought excitement, and for the quiet moments when I just needed to escape and reflect. I don't think I would have survived this journey without her-those drives were not just about getting from one place to another; they were my space to breathe, to think, and to keep pushing forward.

This research would not have been possible without the generous support of my PhD sponsors: Adobe, DARPA, Meta, and DEVCOM Army Research Laboratory—whose funding enabled me to pursue my work with the necessary resources and opportunities. Their support has been

instrumental in allowing me to explore new ideas and contribute meaningfully to my field.

Finally, I must acknowledge one of the most consistent companions throughout this PhD—coffee. While the exact origins of coffee remain a mystery, its contribution to this dissertation is undeniable. It has powered countless late nights, early mornings, and moments of deep contemplation, ensuring that I stayed focused and driven. This journey has been filled with challenges, growth, and countless memories, and I am forever grateful to everyone who has been a part of it.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Traditional CV tasks have evolved significantly with the advent of deep learning models like CNNs and transformers [127, 281, 310]. These advancements have enhanced tasks such as real-time object detection, advanced image classification, visions and large language models, and facial recognition, leading to substantial improvements in accuracy and efficiency. All the methods which takes the image as is for the input are treated as passive schemes [5, 6, 7, 4]. Adversarial attacks have also become more sophisticated, exploiting deep neural networks' vulnerabilities to create misleading inputs that appear normal to humans [34, 109, 206].

Adversarial attacks in computer vision underscore a significant societal problem, highlighting the vulnerabilities inherent in the deployment of machine learning technologies. The subtle manipulations used in these attacks can lead to misinterpretations by AI systems, potentially causing widespread harm in critical applications such as security surveillance, healthcare diagnostics, and autonomous transportation [138, 78]. Moreover, the exploitation of these vulnerabilities by malicious actors could undermine public trust in AI technologies, stalling progress and adoption. The challenge of adversarial attacks extends beyond technical hurdles, posing ethical, legal, and safety concerns that society must address to ensure the responsible and secure advancement of computer vision applications.

While adversarial attacks in computer vision are often viewed through the lens of their potential for harm, there exists a transformative perspective that leverages these techniques for social good [5, 6, 7]. By understanding and harnessing the principles behind adversarial perturbations, researchers have innovated protective measures which utilizes techniques that enhance various computer vision applications using imperceptible signals added onto the original media, known as ***templates*** [5, 7, 6], as shown in Fig. 1.1. The methods that encrypt input data using templates, allowing the encrypted data to enhance the performance for an application, are referred to as ***proactive schemes***. In contrast, all the methods which takes the input data as is are treated as ***passive schemes*** [5, 7, 6, 4].

Proactive schemes have been used for a long time, using different methodologies. Previously,

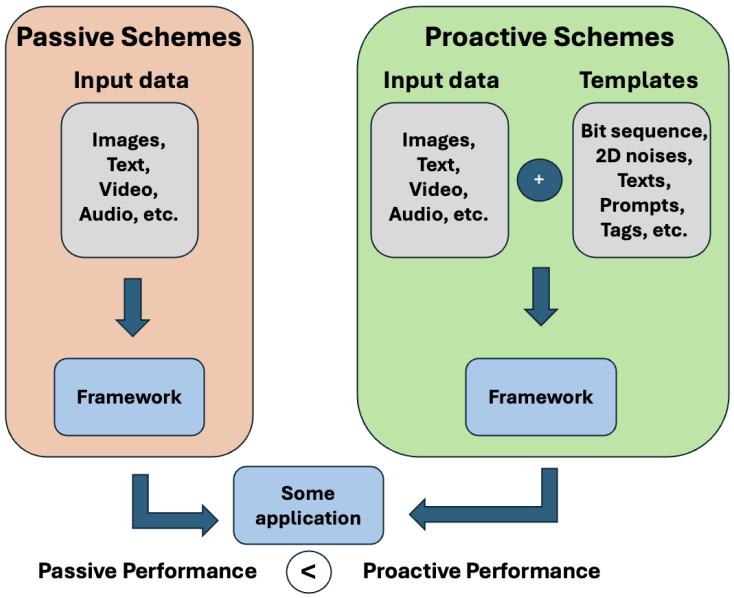


Figure 1.1 **Passive vs. Proactive Schemes**: Passive schemes take input as is for their method, while proactive schemes use templates to encrypt the input and then use the encrypted input for the particular method.

proactive schemes focused on simple enhancements in image processing, with applications like steganography, encryption, and security surveillance [155, 234]. Proactive schemes also share the similar idea from approaches using stochastic resonance in signal processing [95] and non-linear systems [274]. Stochastic resonance occurs when a weak signal that is too faint to be detected by a system is enhanced by the addition of noise, allowing the system to cross a detection threshold. This happens because the noise helps to push the weak signal above the threshold intermittently, making it detectable by the system. The interplay between the noise and the signal can amplify the signal's effects at certain points, leading to an overall improvement in the system's ability to process or detect the signal. The noise level is tuned to an optimal range—too little noise won't help the signal, and too much noise will overwhelm it. However, deep learning has opened up the door for utilizing stochastic resonance in improving the performance by thresholding neural networks [47], noise-boosted activation functions [261], non-linear stochastic dynamics [277], Fourier domain [253] *etc*. Similarly, many works inject noise in the data or labels as augmentations, to improve the robustness of the deep learning networks [231, 360, 358, 181]. Although the above methods resemble proactive schemes, the focus of this thesis is on the usage of these schemes for social good in the current

deep learning era for a variety of applications in the realm of computer vision and natural language processing.

A general framework for proactive schemes is shown in Fig. 1.1. Each method has a specific **encryption process** and **learning process** associated with it, which depends on the **application**. Firstly, the encryption process is a critical component in the design of proactive schemes. This process involves the use of various innovative methods or operations to embed template information within digital media. The templates used for encryption can take the form of many different types of signals like bit sequences, 2D noises, texts, visual prompts, predefined tags, audio, etc. The templates are added onto different types of media, such as images, texts, videos, audios, etc. The goal of the encryption process is to create a secure framework that can withstand potential attacks while maintaining the quality of the encrypted media compared to the original. As technology evolves, so do the techniques used for encryption, making it an ever-growing area of research.

Next, the learning process involves training models to recognize and incorporate these templates, whether they are bit sequences, 2D templates, text signals, or visual prompts—into various forms of digital content. This integration is achieved through specialized learning paradigms, *eg.* encoder decoder frameworks, learning via objective functions, adversarial learning, specialized architectures like GANs, transformers, *etc.*, tailored to the unique characteristics of each template type. The effectiveness of the learning process is constrained, optimized and evaluated using a range of objective functions and metrics. This encompasses the stage of learning objectives, which govern the efficacy of the proactive schemes for various applications. The learning objectives are heavily dependent on the application for which the method is being used.

These schemes are used for a plethora of applications, including encryption, GenAI and LLM defense, preservation of authorship rights, ownership verification, improving CV applications, and privacy protection. Based on each application, the researchers have explored various combinations of respective modules of proactive schemes, *i.e.*,type of template, encryption process, and learning process. In this thesis we explore various applications for proactive schemes. The main innovation for proactive schemes comes in choosing the right kind of template to be added

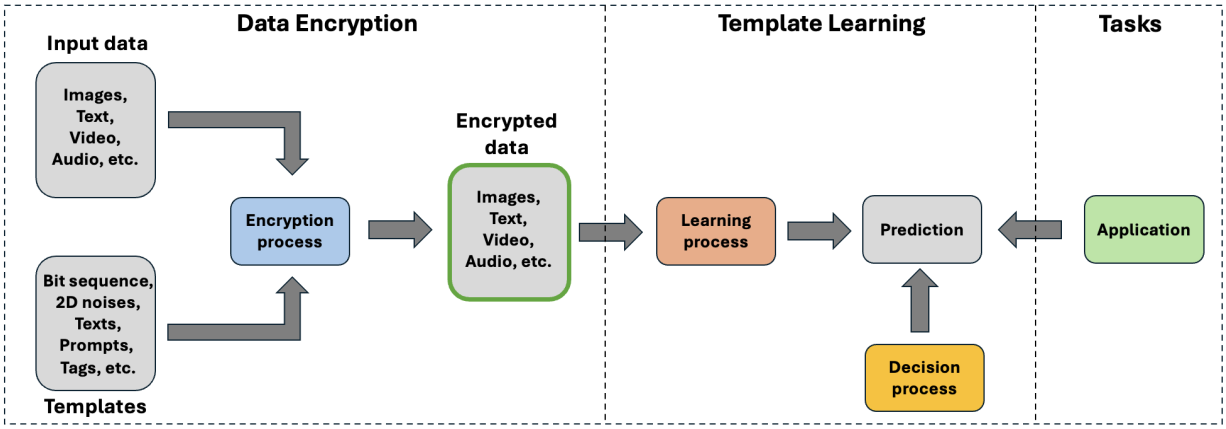


Figure 1.2 **A general overview of the proactive framework**. The method starts by encrypting the input data with some kind of template. This is known as Encryption process. The framework passes through some learning process, and is evaluated based on certain decision process. Finally, every method is associated with some application.

onto the media type. This step is crucial, as it'll guide the overall path for different blocks of proposed approach. We propose various work in this thesis which explore different application domains which are benefiting by the usage of proactive schemes as compared to their passive counterparts. We show the effectiveness of proactive schemes across image manipulation detection, image manipulation localization, 2D generic and camouflaged object detection, concept attribution for media provenance, and action recognition.

Image manipulation detection algorithms are traditionally designed to differentiate between images altered by specific Generative Models (GMs) and authentic images, but they often struggle to generalize when encountering images manipulated by previously unseen GMs. Typically, these detection methods operate in a passive manner [69, 265, 340, 65], simply analyzing the input image as it is. In contrast, we introduce a proactive approach to image manipulation detection, which is based on the recovery of the template from encrypted real and manipulated images. The core innovation of our method lies in the estimation of templates that, when superimposed onto the original image, enhance the accuracy of detecting manipulations. Specifically, a real image protected by these templates, along with its manipulated counterpart, can be more effectively distinguished than a plain real image compared to its altered version. These templates are crafted based on specific constraints designed to ensure their effectiveness. Unlike prior works, we use

unsupervised learning to estimate this template set based on certain constraints. We define different loss functions to incorporate properties including small magnitude, more high frequency content, orthogonality and classification ability as constraints to learn the template set. In comparison, our approach differs from related proactive works [267, 356, 272, 325] in its purpose (detection vs other tasks), template learning (learnable vs predefined), the number of templates, and the generalization ability.

As the quality of images generated by various Generative Models (GMs) continues to improve, there is an increasing need not only to detect whether an image has been manipulated but also to pinpoint the specific pixels that have been altered. However, existing methods [194, 141, 65], often described as passive, show limited ability to generalize across unseen GMs and different types of modifications. To address this challenge, we propose a proactive manipulation localization strategy, named MaLP. In this approach, real images are encrypted with a specially learned template. If the image is later manipulated by a GM, this template not only aids in the binary detection of the manipulation but also assists in identifying the exact pixels that were modified. We design a two-branch architecture consisting of a shallow CNN network and a transformer to optimize the template during training. While the former leverages local-level features due to its shallow depth, the latter focuses on global-level features to better capture the affinity of the far-apart regions. The joint training of both networks enables the MaLP to learn a better template, having embedded the information of both levels. During inference, the CNN network alone is sufficient to estimate the fakeness map with a higher inference efficiency. Our results demonstrate that MaLP outperforms previous passive methods. We further validate the robustness of MaLP by testing it on 22 different GMs, establishing a new benchmark for future research in manipulation localization.

Traditional object detection research in 2D images has primarily focused on tasks such as detecting objects in both generic [260, 254, 43, 32, 117, 128] and camouflaged scenarios [82, 81, 149, 178, 120, 122, 121]. These approaches are typically considered passive, as they process the input images in their original form. However, since convergence to a global minimum is not necessarily optimal in neural networks, the resulting trained weights in object detectors may

not be ideal. To address this issue, we propose a proactive wrapper scheme called PrObeD, designed to enhance the performance of existing object detectors by learning an auxiliary signal. PrObeD utilizes an encoder-decoder architecture where the encoder generates an image-specific signal, referred to as a template, which is used to encrypt the input images. The decoder is then responsible for recovering this template from the encrypted images. We posit that by learning an optimal template, the object detector's performance can be significantly improved. The template functions as a mask, emphasizing semantic features that are particularly useful for the object detector. Fine-tuning the object detector with these encrypted images results in enhanced detection performance for both generic and camouflaged objects.

Generative AI (GenAI) is revolutionizing creative workflows by enabling the synthesis and manipulation of images through high-level prompts. However, the current systems fall short in adequately supporting creatives in receiving recognition or compensation when their content is used for training GenAI models [11, 269, 328]. To address this gap, we introduce ProMark, a causal attribution method designed to trace the origin of synthetically generated images back to specific training data concepts, such as objects, motifs, templates, artists, or styles. ProMark works by proactively embedding concept information into the input training images through imperceptible watermarks, which are then retained in the images generated by diffusion models—whether unconditional or conditional.

Building on top of ProMark, CustomMark is proposed for concept attribution offering greater flexibility and efficiency in attribution within pre-trained generative AI models. Unlike ProMark, which requires embedding attribution markers across all training data concepts upfront, CustomMark enables selective, concept-specific watermarking, allowing artists to opt-in only for specific styles or concepts without impacting the rest of the model. This approach is more scalable and computationally efficient, as it avoids the need for retraining the entire model with pre-defined attribution markers. Furthermore, CustomMark supports sequential learning, allowing the model to seamlessly add new attributions as additional styles emerge, achieving rapid customization with only a fraction of the retraining time. This means CustomMark can embed watermarks for

new concepts in a streamlined way, maintaining image quality and ensuring resilience against modifications.

Using principles of proactive learning, we introduce PiVoT, a pioneering video-based proactive wrapper that enhances the functionality of video action detectors, specifically targeting Action Recognition (AR) and Spatio-Temporal Action Detection (STAD). AR and STAD are essential for interpreting dynamic scenes and human activities, and they benefit from advancements in deep learning architectures such as CNNs and Transformers. In line with proactive scheme applications,PiVoT is crafted to seamlessly integrate with existing detector architectures while minimizing training costs. It adopts a template-enhanced Low-Rank Adaptation (LoRA) strategy, leveraging a 3D U-Net to produce action-specific templates that effectively elevate detection capabilities. This targeted adaptation fine-tunes select elements of the detector, like the CNN backbone or transformer attention modules, preserving the core structure.

Further, we also discuss our work for reverse engineering othe parameters of the generative models. State-of-the-art (SOTA) Generative Models (GMs) have the capability to produce photo-realistic images that are nearly indistinguishable from real photographs by the human eye [158, 51, 156, 164, 29, 42, 74]. As these models become increasingly sophisticated, the need to identify and understand manipulated media is essential to address the societal concerns surrounding the potential misuse of GMs. In response to this challenge, we introduce a novel approach that involves reverse engineering GMs to infer their underlying hyperparameters based solely on the images they generate. We define this new problem as "model parsing," which entails estimating the network architectures and training loss functions of GMs by analyzing their output images—an exceedingly difficult task for humans. To address this problem, we propose a two-component framework: a Fingerprint Estimation Network (FEN), which derives a unique GM fingerprint from a generated image by training with four specific constraints that guide the fingerprint to exhibit desirable properties; and a Parsing Network (PN), which uses the estimated fingerprints to predict the GM's network architecture and loss functions. Although this work is not directly said to be proactive in terms of adding a template to any media, the fingerprint estimated for every generative model

serves the role of template. This fingerprint is left by every generative model serving as additional signal onto the images helping the task of model parsing.

1.1 Contributions of the Thesis

The thesis focuses on proactive solutions to problem in different application domains. These approaches not only outperform their passive counterparts, but they also help better generalization in the respective fields. We discuss 7 different applications as mentioned below:

◇ Image Manipulation Detection: This proactive approach estimates templates that, when added to real images, improve the precision of detecting manipulations by various Generative Models, offering superior generalization across multiple unseen models.

◇ MaLP: MaLP encrypts real images with learned templates that not only aid in binary manipulation detection but also effectively localize altered pixels, demonstrating robust performance across 22 different Generative Models.

◇ PrObeD: PrObeD introduces a proactive scheme that uses an encoder-decoder architecture to generate and embed image-specific templates, significantly enhancing object detection performance in both generic and camouflaged scenarios across various datasets.

◇ ProMark: ProMark embeds imperceptible watermarks into training images, enabling the causal attribution of generated images to their original concepts, while maintaining high image quality and outperforming correlation-based methods.

◇ CustomMark: CustomMark is a versatile and efficient approach for concept attribution in pre-trained generative AI models, allowing targeted and incremental watermarking of specific concepts without requiring full model retraining.

◇ PiVoT: PiVoT is a proactive framework that boosts the accuracy of video-based action detectors by embedding action-specific templates through a LoRA-enhanced architecture, delivering consistent performance gains across multiple detectors and datasets with minimal computational costs.

◇ Model Parsing: A framework that reverse engineers Generative Models by extracting fingerprints from generated images to predict the models' network architectures and loss functions, showing effectiveness in deepfake detection and image attribution tasks.

1.2 Dissertation Organization

We organize the remaining chapters of the dissertation as follows. Chapter 2 introduces the overall framework of the proposed proactive scheme for image manipulation detection. We propose to learn a set of templates with desired properties, achieving higher performance than a single template approach. Chapter 3 describes the proactive methodology, termed MaLP, for image manipulation localization, applicable to both face and generic images. The framework uses two-branch architecture capturing both local and global level features to learn a set of templates in an unsupervised manner. Chapter 4 proposes a novel proactive approach *PrObeD* for the object detection task. We mathematically demonstrate that, under certain assumptions, the proactive method leads to a more effectively converged model compared to the passive detector, thereby resulting in a superior object detector. Chapter 5 discusses ProMark, which performs causal attribution of synthetic images to the predefined concepts in the training images that influenced the generation. Chapter 6 discusses CustomMark which offers flexible, efficient concept attribution in generative AI models by enabling selective, concept-specific watermarking without full model retraining. Chapter 7 introduces a proactive wrapper that enhances video action detection by integrating seamlessly with existing architectures and improving accuracy across variety of video-based detectors and datasets. Chapter 8 discusses about going beyond model classification by formulating a novel problem of model parsing for GMs by using a framework with fingerprint estimation and clustering of GMs to predict the network architecture and loss functions, given a single generated image.

CHAPTER 2

PROACTIVE IMAGE MANIPULATION DETECTION

Image manipulation detection algorithms are often trained to discriminate between images manipulated with particular Generative Models (GMs) and genuine/real images, yet generalize poorly to images manipulated with GMs unseen in the training. Conventional detection algorithms receive an input image passively. By contrast, we propose a proactive scheme to image manipulation detection. Our key enabling technique is to estimate a set of templates which when added onto the real image would lead to more accurate manipulation detection. That is, a template protected real image, and its manipulated version, is better discriminated compared to the original real image vs. its manipulated one. These templates are estimated using certain constraints based on the desired properties of templates. For image manipulation detection, our proposed approach outperforms the prior work by an average precision of 16% for CycleGAN and 32% for GauGAN. Our approach is generalizable to a variety of GMs showing an improvement over prior work by an average precision of 10% averaged across 12 GMs[1].

2.1 Introduction

It's common for people to share personal photos on social networks. Recent developments of image manipulation techniques via Generative Models (GMs) [107] result in serious concerns over the authenticity of the images. As these techniques are easily accessible [303, 194, 52, 238, 385, 53, 214], the shared images are at a greater risk for misuse after manipulation. Generation of fake images can be categorized into two types: entire image generation and partial image manipulation [325, 330]. While the former generates entirely new images by feeding a noise code to the GM, the latter involves the partial manipulation of a real image. Since the latter alters the semantics of real images, it is generally considered as a greater risk, and thus partial image manipulation detection is the focus of this work.

Detecting such manipulation is an important step to alleviate societal concerns on the authenticity

[1] *Vishal Asnani*, Xi Yin, Tal Hassner, Sijia Liu, and Xiaoming Liu. "Proactive image manipulation detection." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022.

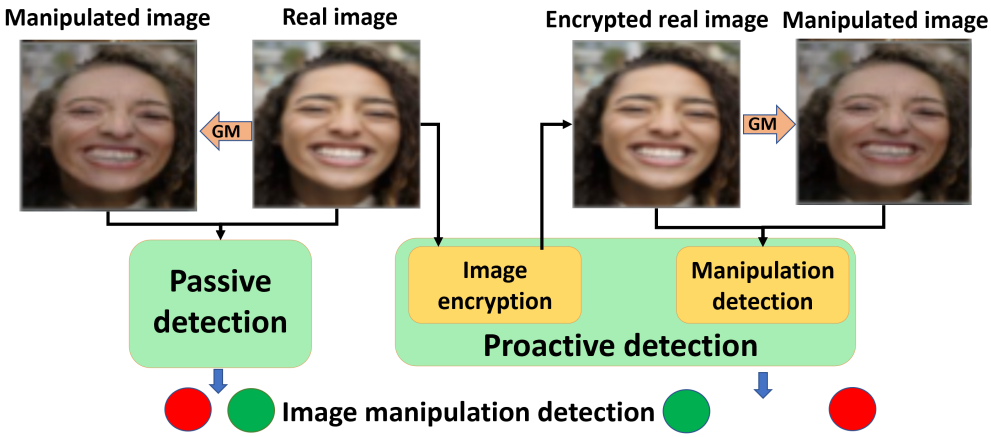


Figure 2.1 **Passive *vs*.proactive image manipulation detection** Classic passive schemes take an image as it is to discriminate a real image *vs*.its manipulated one created by a Generative Model (GM). In contrast, our proactive scheme performs encryption of the real image so that our detection module can better discriminate the encrypted real image *vs*.its manipulated counterpart.

Table 2.1 Comparison of our approach with prior works. Generalizable column means if the performance is reported on datasets unseen during training. [Keys: Img. man. det.: Image manipulation detection, Img. ind.: Image independent].

Method	Year	Detection scheme	Purpose	Manipulation type	Generalizable	Add perturbation	Recover perturbation	Template learning method	# of templates	Img. ind. templates
Cozzolino *et al*. [60]	2018	Passive	Img. man. det.	Entire/Partial	✔	✖	✖	-	-	-
Nataraj *et al*. [222]	2019	Passive	Img. man. det.	Entire/Partial	✔	✖	✖	-	-	-
Rossler *et al*. [265]	2019	Passive	Img. man. det.	Entire/Partial	✖	✖	✖	-	-	-
Zhang *et al*. [371]	2019	Passive	Img. man. det.	Partial	✔	✖	✖	-	-	-
Wang *et al*. [330]	2020	Passive	Img. man. det.	Entire/Partial	✔	✖	✖	-	-	-
Wu *et al*. [340]	2020	Passive	Img. man. det.	Entire/Partial	✖	✖	✖	-	-	-
Qian *et al*. [250]	2020	Passive	Img. man. det.	Entire/Partial	✖	✖	✖	-	-	-
Dang *et al*. [65]	2020	Passive	Img. man. det.	Partial	✖	✖	✖	-	-	-
Masi *et al*. [211]	2020	Passive	Img. man. det.	Partial	✖	✖	✖	-	-	-
Nirkin *et al*. [230]	2021	Passive	Img. man. det.	Partial	✖	✖	✖	-	-	-
Asnani *et al*. [8]	2021	Passive	Img. man. det.	Entire/Partial	✔	✖	✖	-	-	-
Segalis *et al*. [272]	2020	Proactive	Deepfake disruption	Partial	✖	✔	✖	Adversarial attack	1	✔
Ruiz *et al*. [267]	2020	Proactive	Deepfake disruption	Partial	✖	✔	✖	Adversarial attack	1	✔
Yeh *et al*. [356]	2020	Proactive	Deepfake disruption	Partial	✖	✔	✖	Adversarial attack	1	✔
Wang *et al*. [325]	2021	Proactive	Deepfake tagging	Partial	✖	✔	✔	Fixed template	> 1	✖
Ours	-	Proactive	Img. man. det.	Partial	✔	✔	✔	Unsupervised learning	> 1	✔

of shared images. Prior works have been proposed to combat manipulated media [69]. They leverage properties that are prone to being manipulated, including mouth movement [265], steganalysis features [340], attention mechanism [65, 197], *etc*.. However, these methods are often overfitted to the image manipulation method and the dataset used in training, and suffer when tested on data with a different distribution.

All the aforementioned methods adopt a *passive* scheme since the input image, being real or manipulated, is accepted as is for detection. Alternatively, there is also a *proactive* scheme proposed for a few computer vision tasks, which involves adding signals to the original image. For example, prior works add a predefined template to real images which either disrupt the output of

the GM [267, 356, 272] or tag images to real identities [325]. This template is either a one-hot encoding [325] or an adversarial perturbation [267, 356, 272].

Motivated by improving the generalization of manipulation detection, as well as the proactive scheme for other tasks, this paper proposes a *proactive* scheme for the purpose of image manipulation detection, which works as follows. When an image is captured, our algorithm adds an imperceptible signal (termed as *template*) to it, serving as an encryption. If this encrypted image is shared and manipulated through a GM, our algorithm accurately distinguishes between the encrypted image and its manipulated version by recovering the added template. Ideally, this encryption process could be incorporated into the camera hardware to protect all images after being captured. In comparison, our approach differs from related proactive works [267, 356, 272, 325] in its purpose (detection vs other tasks), template learning (learnable vs predefined), the number of templates, and the generalization ability.

Our key enabling technique is to learn a template set, which is a non-trivial task. First, there is no ground truth template for supervision. Second, recovering the template from manipulated images is challenging. Third, using one template can be risky as the attackers may reverse engineer the template. Lastly, image editing operations such as blurring or compression could be applied to encrypted images, diminishing the efficacy of the added template.

To overcome these challenges, we propose a template estimation framework to learn a set of *orthogonal templates*. We perform image manipulation detection based on the recovery of the template from encrypted real and manipulated images. Unlike prior works, we use unsupervised learning to estimate this template set based on certain constraints. We define different loss functions to incorporate properties including small magnitude, more high frequency content, orthogonality and classification ability as constraints to learn the template set. We show that our framework achieves superior manipulation detection than State-of-The-Art (SoTA) methods [325, 371, 60, 222]. We propose a novel evaluation protocol with 12 different GMs, where we train on images manipulated by one GM and test on unseen GMs. In summary, the contributions of this paper include:

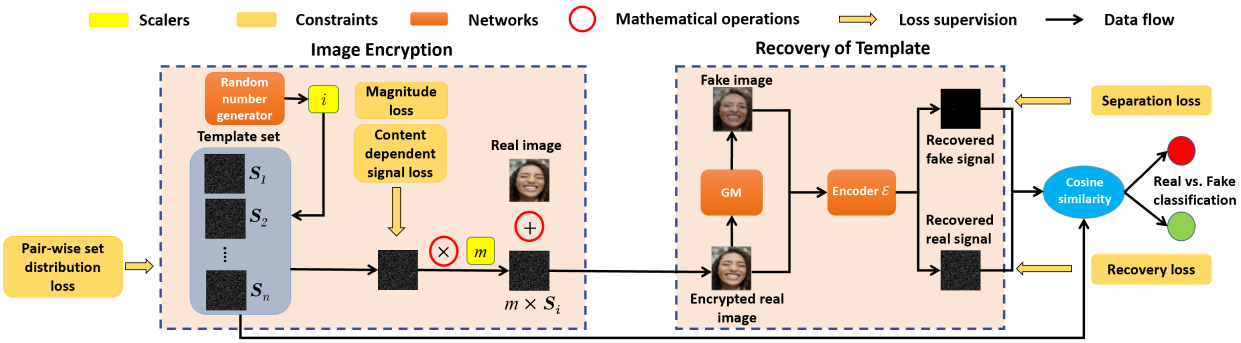


Figure 2.2 Our proposed framework includes two stages: 1) selection and addition of templates; and 2) the recovery of the estimated template from encrypted real images and manipulated images using an encoder network. The GM is used in the inference mode. Both stages are trained in an end-to-end manner to output a set of templates. For inferences, the first stage is mandatory to encrypt the images. The second stage is used only when there is a need of image manipulation detection.

- We propose a novel proactive scheme for image manipulation detection.
- We propose to learn a set of templates with desired properties, achieving higher performance than a single template approach.
- Our method substantially outperforms the prior works on image manipulation detection. Our method is more generalizable to different GMs showing an improvement of 10% average precision averaged across 12 GMs.

2.2 Related Works

Passive deepfake detection. Most deepfake detection methods are passive. Wang *et al*. [330] perform binary detection by exploring frequency domain patterns from images. Zhang *et al*. [371] propose to extract the median and high frequencies to detect the upsampling artifacts by GANs. Asnani *et al*. [8] propose to estimate fingerprint using certain desired properties for generative models which produce fake images. Others use autoencoders [60], hand-crafted features [222], face-context discrepancies [230], mouth and face motion [265], steganalysis features [340], xception-net [54], frequency domain [211] and attention mechanisms [65]. These aforementioned passive deepfake detection methods suffer from generalization. We propose a novel proactive scheme for manipulation detection, aiming to improve the generalization.

Proactive schemes. Recently, some proactive methods are proposed by adding an adversarial noise onto the real image. Ruiz *et al*. [267] perform deepfake disruption by using adversarial attack in image translation networks. Yeh *et al*. [356] disrupt deepfakes to low quality images by performing adversarial attacks on real images. Segalis *et al*. [272] disrupt manipulations related to face-swapping by adding small perturbations. Wang *et al*. [325] propose a method to tag images by embedding messages and recovering them after manipulation. Wang *et al*. [325] use a one-hot encoding message instead of adversarial perturbations. Compared with these works, our method focuses on image manipulation detection rather than deepfake disruption or deepfake tagging. Our method learns a set of templates and recovers the added template for image manipulation detection. Our method also generalizes better to unseen GMs than prior works. Tab. 2.1 summarizes the comparison with prior works.

Watermarking and cryptography methods. Digital watermarking methods have been evolving from using classic image transformation techniques to deep learning techniques. Prior work have explored different ways to embed watermarks through pixel values [14] and spatial domain [282]. Others [152, 161, 355] use frequency domains including transformation coefficients obtained via SVD, discrete wavelet transform (DWT), discrete cosine transform (DCT) and discrete fourier transform (DFT) to embed watermarks. Recently, deep learning techniques proposed by Zhu *et al*. [384], Baluja *et al*. [13] and Tancik *et al*. [297] use an encoder-decoder architecture to embed watermarks into an image. All of these methods aim to either hide sensitive information or protect the ownership of digital images. While our algorithm shares the high-level idea of image encryption, we develop a novel framework for an entirely different purpose, *i.e*., proactive image manipulation detection.

2.3 Proposed Approach

2.3.1 Problem Formulation

We only consider GMs which perform partial image manipulation that takes a real image as input for manipulation. Let $\boldsymbol{X}^a$ be a set of real images which when given as input to a GM G would output $G(\boldsymbol{X}^a)$, a set of manipulated images. Conventionally, passive image manipulation detection

methods perform binary classification on $\boldsymbol{X}^a$ *vs.*$G(\boldsymbol{X}^a)$. Denote $\boldsymbol{X} = \{\boldsymbol{X}^a, G(\boldsymbol{X}^a)\} \in \mathbb{R}^{128\times128\times3}$ as the set of real and manipulated images, the objective function for passive detection is formulated as follows:

$$\min_{\theta}\Big\{-\sum_{j}\Big(y_j.\log(\mathcal{H}(\boldsymbol{X}_j;\theta)) - (1-y_j).\log(1-\mathcal{H}(\boldsymbol{X}_j;\theta))\Big)\Big\}. \tag{2.1}$$

where y is the class label and $\mathcal{H}$ refers to the classification network used with parameters θ.

In contrast, for our proactive detection scheme, we apply a transformation $\mathcal{T}$ to a real image from set $\boldsymbol{X}^a$ to formulate a set of encrypted real images represented as: $\mathcal{T}(\boldsymbol{X}^a)$. We perform image encryption by adding a learnable template to the image which acts as a defender's signature. Further, the set of encrypted real images $\mathcal{T}(\boldsymbol{X}^a)$ is given as input to the GM, which produces a set of manipulated images $G(\mathcal{T}(\boldsymbol{X}^a))$. We propose to learn a set of templates rather than a single one to increase security as it is difficult to reverse engineer all templates. Thus for a real image $\boldsymbol{X}^a_j \in \boldsymbol{X}^a$, we define $\mathcal{T}$ via a set of n orthogonal templates $\mathcal{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, ...\boldsymbol{S}_n\}$ where $\boldsymbol{S}_i \in \mathbb{R}^{128\times128}$ as follows:

$$\mathcal{T}(\boldsymbol{X}^a_j) = \boldsymbol{X}^a_j + \boldsymbol{S}_i, \text{ where } i \in \{1, 2, ..., n\}. \tag{2.2}$$

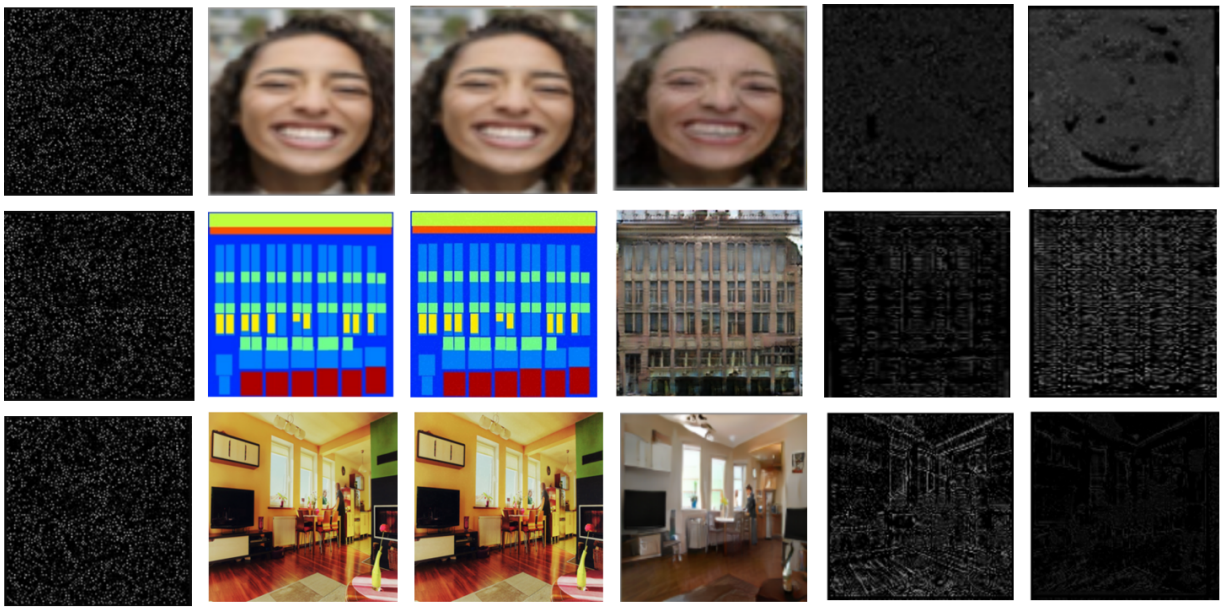
After applying the transformation $\mathcal{T}$, the objective function defined in Eq. (2.1) can be re-written as:

$$\begin{aligned}\min_{\theta,\boldsymbol{S}_i}\Big\{-\sum_{j}\Big(&y_j.\log(\mathcal{H}(\mathcal{T}(\boldsymbol{X}_j);\theta,\boldsymbol{S}_i))+\\ &(1-y_j).\log(1-\mathcal{H}(\mathcal{T}(\boldsymbol{X}_j);\theta,\boldsymbol{S}_i))\Big)\Big\}.\end{aligned} \tag{2.3}$$

The goal is to find $\boldsymbol{S}_i$ for which corresponding images in $\boldsymbol{X}^a$ and $\mathcal{T}(\boldsymbol{X}^a)$ have no significant visual difference. More importantly, if $\mathcal{T}(\boldsymbol{X}^a)$ is modified by any GM, this would improve the performance for image manipulation detection.

2.3.2 Proposed Framework

As shown in Fig. 2.2, our framework consists of two stages: image encryption and recovery of template. The first stage is used for selection and addition of templates, while the second stage involves the recovery of templates from images in $\mathcal{T}(\boldsymbol{X}^a)$ and $G(\mathcal{T}(\boldsymbol{X}^a))$. Both stages are trained



(a) (b) (c) (d) (e) (f)

Figure 2.3 Visualization of (a) a template set with the size of 3, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Each row corresponds to image manipulation by different GM (top: StarGAN, middle: CycleGAN, bottom: GauGAN). The template recovered from encrypted real images is more similar to the template set than the one from manipulated images. The addition of the template creates no visual difference between real and encrypted real images. We provide more examples of real images evaluated using our framework in the supplementary material.

in an end-to-end manner with GM parameters fixed. For inference, each stage is applied separately. The first stage is a mandatory step to encrypt the real images while the second stage would only be used when image manipulation detection is needed.

2.3.2.1 Image Encryption

We initialize a set of n templates as shown in Fig. 2.2, which is optimized during training using certain constraints. As formulated in Eq. (2.2), we randomly select and add a template from our template set to every real image. Our objective is to estimate an optimal template set from which any template is capable of protecting the real image in $\boldsymbol{X}^a$.

Although we constrain the magnitude of the templates using L_2 loss, the added template still degrades the quality of the real image. Therefore, when adding the template to real images, we

control the strength of the added template using a hyperparameter m. We re-define $\mathcal{T}$ as follows:

$$\mathcal{T}(\boldsymbol{X}_j^a) = \boldsymbol{X}_j^a + m \times \boldsymbol{S}_i \text{ where } i \in \{1, 2, ..., n\}. \tag{2.4}$$

We perform an ablation study of varying m in Sec. 2.4.3, and find that setting m at 30% performs the best.

2.3.2.2 Recovery of Templates

To perform image manipulation detection as shown in Fig. 2.2, we attempt to recover our added template from images in $\mathcal{T}(\boldsymbol{X}^a)$ using an encoder $\mathcal{E}$ with parameters $\theta_{\mathcal{E}}$. For any real image $\boldsymbol{X}_j^a \in \boldsymbol{X}^a$, we define the recovered template from encrypted real image $\mathcal{T}(\boldsymbol{X}_j^a)$ as $\boldsymbol{S}_R = \mathcal{E}(\mathcal{T}(\boldsymbol{X}_j^a))$ and from manipulated image $G(\mathcal{T}(\boldsymbol{X}_j^a))$ as $\boldsymbol{S}_F = \mathcal{E}(G(\mathcal{T}(\boldsymbol{X}_j^a)))$. As template selection from the template set is random, the encoder receives more training pairs to learn how to recover any template from an image, which contributes positively to the robustness of the recovery process. We visualize our trained template set $\mathcal{S}$, and the recovered templates $\boldsymbol{S}_{R/F}$ in Fig. 2.3.

The main intuition of our framework design is that $\boldsymbol{S}_R$ should be much more similar to the added template and vice-versa for $\boldsymbol{S}_F$. Thus, to perform image manipulation detection, we calculate the cosine similarity between $\boldsymbol{S}_{R/F}$ and all learned templates in the set $\mathcal{S}$ rather than merely using a classification objective. For every image, we select the maximum cosine similarity across all templates as the final score. Therefore, we update logit scores in Eq. (2.3) by cosine similarity scores as shown below:

$$\begin{aligned}\min_{\theta_{\mathcal{E}}, \boldsymbol{S}_i} \Bigg\{ -\sum_j \Big(y_j . \log(\max_{i=1...n} (\text{Cos}(\mathcal{E}(\mathcal{T}(\boldsymbol{X}_j); \theta_{\mathcal{E}}), \boldsymbol{S}_i))) + \\ (1 - y_j) . \log(1 - \max_{i=1...n} (\text{Cos}(\mathcal{E}(\mathcal{T}(\boldsymbol{X}_j); \theta_{\mathcal{E}}), \boldsymbol{S}_i))) \Big) \Bigg\}.\end{aligned} \tag{2.5}$$

2.3.2.3 Unsupervised Training of Template Set

Since there is no ground truth for supervision, we define various constraints to guide the learning process. Let $\boldsymbol{S}$ be the template selected from set $\mathcal{S}$ to be added onto a real image. We formulate five loss functions as shown below.

Magnitude loss. The real image and the encrypted image should be as similar as possible visually as the user does not want the image quality to deteriorate after template addition. Therefore, we

propose the first constraint to regularize the magnitude of the template:

$$J_m = ||\boldsymbol{S}||_2^2. \tag{2.6}$$

Recovery loss. We use an encoder network to recover the added template. Ideally, the encoder output, *i.e.*, the recovered template $\boldsymbol{S}_R$ of the encrypted real image, should be the same as the original added template $\boldsymbol{S}$. Thus, we propose to maximize the cosine similarity between these two templates:

$$J_r = 1 - \text{Cos}(\boldsymbol{S}, \boldsymbol{S}_R). \tag{2.7}$$

Content independent template loss. Our main aim is to learn a set of universal templates which can be used for detecting manipulated images from unseen GMs. These templates, despite being trained on one dataset, can be applied to images from a different domain. Therefore, we encourage the high frequency information in the template to be data independent. We propose a constraint to minimize low frequency information:

$$J_c = ||\mathcal{L}(\mathbb{F}(\boldsymbol{S}), k)||_2^2, \tag{2.8}$$

where $\mathcal{L}$ is the low pass filter selecting the $k \times k$ region in the center of the 2D Fourier spectrum, while assigning the high frequency region to zero. $\mathbb{F}$ is the Fourier transform.

Separation loss. We want the recovered template $\boldsymbol{S}_F$ from manipulated images $G(\mathcal{T}(\boldsymbol{X}))$ to be different than all the templates in set $\mathcal{S}$. Thus, we optimize $\boldsymbol{S}_F$ to be orthogonal to all the templates in the set $\mathcal{S}$. Therefore, we take the template for which the cosine similarity between $\boldsymbol{S}_F$ and the template is maximum, and minimize its respective cosine similarity:

$$J_s = \max_{i=1...n} (\text{Cos}(\mathcal{N}(\boldsymbol{S}_i), \mathcal{N}(\boldsymbol{S}_F))), \tag{2.9}$$

where $\mathcal{N}(\boldsymbol{S})$ is the normalizing function defined as $\mathcal{N}(\boldsymbol{S}) = (\boldsymbol{S} - \min(\boldsymbol{S}))/(\max(\boldsymbol{S}) - \min(\boldsymbol{S}))$. Since this loss minimizes the cosine similarity to be 0, we normalize the templates before similarity calculation.

Pair-wise set distribution loss. A template set would ensure that if the attacker is somehow able to get access to some of the templates, it would still be difficult to reverse engineer other templates.

Table 2.2 Performance comparison with prior works.

Method	Train GM	Set size	Test GM Average precision (%)		
			CycleGAN	StarGAN	GauGAN
[222]	CycleGAN	-	**100**	88.20	56.20
[60]	ProGAN	-	77.20	91.70	83.30
[371]	AutoGAN	-	**100**	**100**	61.00
[330]	ProGAN	-	84.00	**100**	67.00
Ours	STGAN	3	96.12	**100**	91.62
		20	99.66	**100**	90.58
	AutoGAN	3	97.87	97.89	86.57
		20	97.05	97.18	84.24
	STGAN + AutoGAN	3	**100**	**100**	**99.69**

Therefore, we propose a constraint to minimize the inter-template cosine similarity to prompt the diversity of the templates in $\mathcal{S}$:

$$J_p = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \text{Cos}(\mathcal{N}(\boldsymbol{S}_i), \mathcal{N}(\boldsymbol{S}_j)). \tag{2.10}$$

The overall loss function for template estimation is thus:

$$J = \lambda_1 J_m + \lambda_2 J_r + \lambda_3 J_c + \lambda_4 J_s + \lambda_5 J_p, \tag{2.11}$$

where λ_1, λ_2, λ_3, λ_4, λ_5 are the loss weights for each term.

2.4 Experiments

2.4.1 Settings

Experimental setup and dataset. We follow the experimental setting of Wang *et al*. [330], and compare with four baselines: [330], [371], [60] and [222]. For training, [330] uses $720K$ images from which the manipulated images are generated by ProGAN [157]. However, as our method requires a GM to perform partial manipulation, we choose STGAN [194] in training as ProGAN synthesizes entire images. We use $24K$ images in CelebA-HQ [157] as the real images and pass them through STGAN to obtain manipulated images for training. For testing, we use 200 real images and pass them through *unseen* GMs such as StarGAN [52], GauGAN [238] and CycleGAN [385]. The real images for testing GMs are chosen from the respective dataset they are trained on, *i.e*.CelebA-HQ for StarGAN, Facades [385] for CycleGAN, and COCO [30] for GauGAN.

Table 2.3 Performance comparison with Wang *et al*. [330].

Method	Train GM	Test GM TDR (%) at low FAR (0.5%)		
		CycleGAN	StarGAN	GauGAN
[330]	ProGAN	55.98	93.88	37.14
Ours	STGAN	**88.50**	**100.00**	**43.00**

Table 2.4 Average precision of 12 testing GMs when our method is trained on only STGAN. All the GMs have different architectures and are trained on diverse datasets. The average precision of almost all GMs are over 90% showing the generalization ability of our method.

GM	UNIT [195]	MUNIT [139]	StarGAN2 [53]	BicycleGAN [386]	CONT_Enc. [240]	SEAN [387]	ALAE [245]	Pix2Pix [144]	DualGAN [357]	CouncilGAN [232]	ESRGAN [333]	GANimation [249]	Average
[330]	64.94	95.33	**100**	**100**	98.18	67.81	92.73	91.26	**98.91**	74.13	57.04	55.19	82.97
Ours	**100**	**100**	**100**	99.05	**98.75**	**97.63**	**93.10**	**92.50**	92.49	**89.71**	**87.30**	**58.69**	**92.43**

Table 2.5 Performance comparison of our proposed method with Ruiz *et al*. [267]. The performance for our proposed method is better than [267] when the testing GM is unseen. Both methods use StarGAN as the training GM.

Method	Test GM Average precision (%)			
	StarGAN	CycleGAN	GANimation	Pix2Pix
[267]	**100**	51.50	52.43	49.08
Ours	**100**	**95.26**	**60.12**	**91.85**

To further evaluate generalization ability of our approach, we use 12 additional unseen GMs that have diverse network architectures and loss functions, and are trained on different datasets. We manipulate each of 200 real images with these 12 GMs which gives $2,400$ manipulated images. The real images are chosen from the dataset that the respective GM is trained on. The list of GMs and their training datasets are provided in the supplementary.

Implementation details. Our framework is trained end-to-end for 10 epochs via Adam optimizer with a learning rate of 10^{-5} and a batch size of 4. The loss weights are set to ensure similar magnitudes at the beginning of training: $\lambda_1 = 100$, $\lambda_2 = 30$, $\lambda_3 = 5$, $\lambda_4 = 0.003$, $\lambda_5 = 10$. If not specified, we set the template set size $n = 3$. We set $k = 50$ in the content independent template loss. All experiments are conducted using one NVIDIA Tesla K80 GPU.

Evaluation metrics. We report average precision as adopted by [330]. To mimic real-world scenarios, we further report true detection rate (TDR) at a low false alarm rate (FAR) of 0.5%.

2.4.2 Image Manipulation Detection Results

As shown in Tab. 2.2, when our training GM is STGAN, we can outperform the baselines by a large margin on GauGAN-based test data, while the performance on StarGAN-based test data remains the same at 100%. When training on STGAN, our method achieves lower performance on CycleGAN. We hypothesis that it is because AutoGAN and CycleGAN share the same model architecture. To validate this, we change our training GM to AutoGAN and observe improvement when tested on CycleGAN. However, the performance drops on other two GMs because the amount of training data is reduced ($24K$ for STGAN and $1.5K$ for AutoGAN). Increasing the number of templates can improve the performance for when trained on STGAN and test on CycleGAN, but degrades for others. The degradation is more when train on AutoGAN. It suggests that it is challenging to find a larger template set on a smaller training set. Finally, using both STGAN and AutoGAN training data can achieve the best performance.

TDR at low FAR. We also evaluate using TDR at low FAR in Tab. 2.3. This is more indicative of the performance in the real world application where the number of real images are exponentially larger than manipulated images. For comparison, we evaluate the pretrained model of [330] on our test set. Our method performs consistently better for all three GMs, demonstrating the superiority of our approach.

Generalization ability. To test our generalization ability, we perform extensive evaluations across a large set of GMs. We compare the performance of our method with [330] by evaluating its pretrained model on a test set of different GMs. Our framework performs quite well on almost all the GMs compared to [330] as shown in Tab. 2.4. This further demonstrates the generalization ability of our framework in the real world where an image can be manipulated by any unknown GM. Compared to [330], our framework achieves an improvement in the average precision of almost 10% averaged across all 12 GMs.

Comparison with proactive scheme work. We compare our work with previous work in proactive scheme [267]. As [267] proposes to disrupt the GM's output, they only provide the distortion results of the manipulated image. To enable binary classification, we take their adversarial real

Table 2.6 Performance comparison of our proposed method with steganography and adversarial attack methods.

Method	Type	Test GM Average precision (%)		
		CycleGAN	StarGAN	GauGAN
Baluja [13]	Steganography	85.64	88.06	81.26
PGD [207]	Adversarial	90.28	98.22	57.71
FGSM [109]	attack	89.21	98.29	63.81
Ours	-	**99.95**	**100**	**98.23**

and disrupted fake images to train a classifier with the similar network architecture as our encoder. Tab. 2.5 shows that [267] works perfectly when the testing GM is the same as the training GM. Yet if the testing GM is unseen, the performance drops substantially. Our method performs much better showing the high generalizability.

Comparison with steganography works. Our method aligns with the high-level idea of digital steganographhy methods [14, 282, 355, 387, 13] which are used to hide an image onto other images. We compare our approach to the recent deep learning-based steganography method, Baluja *et al*. [13], with its publicly available code. We hide and retrieve the template using the pre-trained model provided by [13]. Our approach has far better average precision for each test GM compared to [13] as shown in Tab. 2.6. This validates the effectiveness of template learning and concludes that the digital steganography methods are less generalizable across unknown GMs than our approach.

Comparison with benign adversarial attacks. Adversarial attacks are used to optimize a perturbation to change the class of the image. The learning of the template using our framework is similar to a benign usage of adversarial attacks. We conduct an ablation study to compare our method with common attacks such as benign PGD and FGSM. We remove the losses in Eq. (2.6), Eq. (2.8), and Eq. (2.10) responsible for learning the template and replace them with an adversarial noise constraint. Our approach has better average precision for each test GM than both adversarial attacks as shown in Tab. 2.6. We observe that adversarial noise performed similar to passive schemes offering poor generalization to unknown GMs. This shows the importance of using our proposed constraints to learn the universal template set.

Table 2.7 Average precision (%) with various augmentation techniques in training and testing for three GMs. We apply data augmentation to three scenarios: (1) in training only (2) in testing only and (3) in both training and testing. [Keys: aug.=augmentation, B.=blur, J.=JPEG compression, Gau. No.=Gaussian Noise].

Augmentation		Augmentation type	Method	Test GMs		
Train	Test			CycleGAN	StarGAN	GauGAN
✖	✖	No augmentation	[330]	84.00	**100**	67.00
			Ours	96.12	**100**	91.62
✔	✖	Blur	[330]	90.10	**100**	74.70
			Ours	93.55	**100**	92.35
		JPEG	[330]	93.20	91.80	97.50
			Ours	98.74	98.30	91.85
		B+J (0.5)	[330]	96.80	95.40	98.10
			Ours	94.44	**100**	98.16
		B+J (0.1)	[330]	93.50	84.50	89.50
			Ours	95.79	**100**	95.94
		Resizing	Ours	**100**	**100**	98.97
		Crop		84.45	84.92	94.43
		Gau. No.		99.95	**100**	**99.11**
✖	✔	Blur	Ours	95.74	84.87	70.74
		JPEG		91.91	82.96	84.16
		B+J (0.5)		89.23	82.18	75.53
		Resizing		93.12	77.41	91.45
		Crop		84.04	73.87	70.12
		Gau. No.		73.83	69.47	66.70
✔	✔	Blur	Ours	92.16	**100**	90.15
		JPEG		94.00	97.92	85.91
		B+J (0.5)		87.37	84.92	74.68
		Resizing		99.98	**100**	92.73
		Cropping		77.63	89.22	79.96
		Gau. No.		97.44	**100**	82.32

Data augmentation. We apply various data augmentation schemes to evaluate the robustness of our method. We adopt some of the image editing techniques from Wang *et al*. [330], including (1) Gaussian blurring, (2) JPEG compression, (3) blur + JPEG (0.5), and (4) blur + JPEG (0.1), where 0.5 and 0.1 are the probabilities of applying these image editing operations. In addition, we add resizing, cropping, and Gaussian noise. The implementation details of these techniques are in the supplementary. These techniques are applied after addition of our template to the real images.

We evaluate in three scenarios when augmentation is applied in (1) training, (2) testing, (3) both training and testing. As shown in Tab. 2.7, for the augmentation techniques adopted from [330], we outperform [330] in almost all techniques. We observe significant improvement when

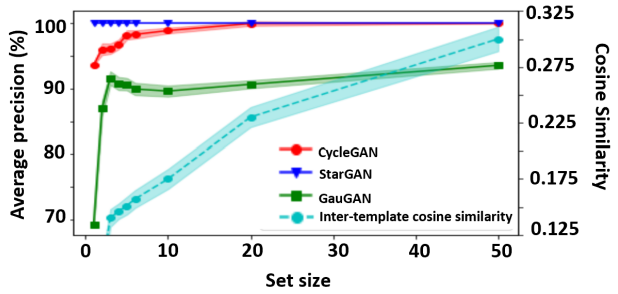


Figure 2.4 Ablation study with varying template set sizes. The performance improves when the set size increases, while the inter-template cosine similarity also increases.

blurring or JPEG compression is applied jointly but the improvement is less when they are applied separately.

As for the different scenarios on when data augmentation is applied, scenario 2 performs the worst because the augmentation applied in testing has not been seen during training. Scenario 3 performs better than scenario 2 in most cases. There is a much larger performance drop when blurring and JPEG are applied together than separately. Cropping performs the worst for both Scenario 1 and 3.

2.4.3 Ablation Studies

Template set size. We study the effects of the template set size. As shown in Fig. 2.4, the average precision increases as the set size is expanding from 1 and saturates around the set size 10. In the meantime, the average cosine similarity between templates within the set increases consistently, as it gets harder to find many orthogonal templates. We also test our framework's run-time for different set sizes. On a Tesla K80 GPU, for the set size of $1, 3, 10, 20$ and 50, the per-image run-time of our manipulation detection is 26.19, 27.16, 28.44, 34.26, and 43.76 ms respectively. Thus, despite increasing the set size enhances our accuracy and security, there is a trade-off with the detection speed which is a important factor too. For comparison, we also test the pretrained model of [330] which gives a per-image run-time of 54.55 ms. Our framework is much faster even with a larger set size which is due to the shallow network in our proactive scheme compared to a deeper network in passive scheme.

Template strength. We use a hyperparameter m to control the strength of our added template.

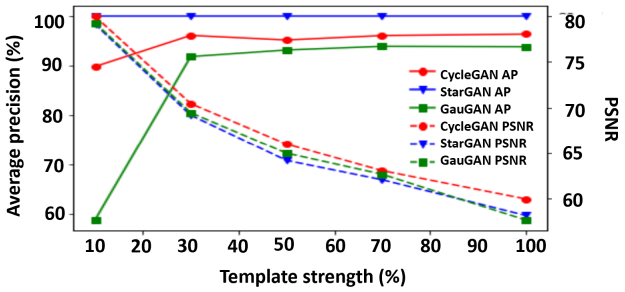


Figure 2.5 Ablation with varying template strengths in the encrypted real images. The lower the template strength, the higher the PSNR is and the harder it is for our encoder to recover it, which leads to lower detection performance.

Table 2.8 Ablation study to remove losses used in our training. Removing any one loss deteriorates the performance compared to our proposed method. Fixing the template or performing direct classification made the results worse. This shows the importance of a variable template and using an encoder for classification purposes.

Loss removed	Test GM Average precision (%)		
	CycleGAN	StarGAN	GauGAN
Magnitude loss (J_m)	94.43	**100**	87.44
Pair-wise set distribution loss (J_p)	66.60	79.99	74.55
Recovery loss (J_r)	51.59	94.18	90.61
Content independent template loss (J_c)	92.01	**100**	80.54
Separation loss (J_s)	92.24	**100**	64.06
J_m, J_p and J_c (fixed template)	46.93	59.88	43.64
J_r and J_s (removing encoder)	50.00	98.24	55.00
None (ours)	**96.12**	**100**	**91.62**

We ablate m and show the results in Fig. 2.5. Intuitively, the lower the strength of the template added, the lower the detection performance since it would be harder for the encoder to recover the original template. Our results support this intuition. For all three GMs, the precision increases as we enlarge the template strength, and converges after 50% strength. We also show the PSNR between the encrypted real image and the original real image. The PSNR decreases as we enlarge the strength as expected. We choose $m = 30\%$ for a trade-off between the detection precision and the visual quality.

Loss functions. Our training process is guided by an objective function with five losses (Eq. (2.11)). To demonstrate the necessity of each loss, we ablate by removing each loss and compare with our full model. As shown in Tab. 2.8, removing any one of the losses results in performance degradation. Specifically, removing the pair-wise set distribution loss, recovery loss

or separation loss causes a larger drop.

To better understand the importance of the data-driven template set, we fix the template set during training, *i.e.*, removing the three losses directly operating on the template and only considering recovery and separation losses for training. We observe a significant performance drop, which shows that the learnable template is indeed crucial for effective image manipulation detection.

Finally, we remove the encoder from our framework and use a classification network with similar number of layers. Instead of recovering templates, the classification network is directly trained to perform binary image manipulation detection via cross-entropy loss. The performance drops significantly. This observation aligns with the previous works [326, 60, 371] stating that CNN networks trained on images from one GM show poor generalizability to unseen GMs. The performance drops for all three GMs but CycleGAN and GauGAN are affected the most, as the datasets are different. For our proposed approach, when we are recovering the template, the encoder ignores all the low frequency information of the images which are data dependent. Thus, being more data (*i.e.*, image content) independent, our encoder is able to achieve a higher generalizability.

Template selection. Given a real image, we randomly select a template from the learnt template set to add to the image. Thus, every image has an equal chance of selecting any one template from the set, resulting in many combinations for the entire test set. This raises the question of finding a worst and best combination of templates for all images in the test set. To answer this, we experiment with a template set size of 50 as a large size may offer higher variation in performance. For each image in $\mathcal{T}(\boldsymbol{X}^a)$ and $G(\mathcal{T}(\boldsymbol{X}^a))$, we calculate the cosine similarity between added template $\boldsymbol{S}$ and recovered template $\boldsymbol{S}_{R/F}$. For the worst/best case of every image, we select the template with the minimum/maximum difference between the real and manipulated image cosine similarities. As shown in Tab. 2.9, GauGAN gives much more variation in the performance compared to CycleGAN and StarGAN. This shows that the template selection is an important step for image manipulation detection. This brings up the idea of training a network to select the best template for a specific image, by using the best case described above as a pseudo ground truth to supervise the network.

Table 2.9 Ablation of template selection schemes at set size of 50.

Selection scheme	Test GM Average precision (%)		
	CycleGAN	StarGAN	GauGAN
Random selection	99.90 ± 0.02	**100 ± 0.00**	93.56 ± 0.52
Biasing one template	99.05 ± 0.37	**100 ± 0.00**	91.21 ± 0.97
Network based	95.46	**100**	90.47
Worst case	94.85	**100**	80.55
Best case	**99.95**	**100**	**98.23**

We hypothesis template selection could be important, but with experiments, the difference of performance among different templates is nearly zero and the network's selection doesn't help in the performance compared with selecting the template randomly as shown in Tab. 2.9. Therefore, we cannot have a pseudo ground truth to train another network for template selection.

Another option for template selection is to select the *same* template for every test image which is equivalent to using one template compromising the security of our method. Nevertheless, we test this option to see the performance variation of biasing one template for all images. The performance variation is larger than our random selection scheme. This shows that each template has a similar contribution to image manipulation detection.

2.5 Conclusion

In this paper, we propose a proactive scheme for image manipulation detection. The main objective is to estimate a set of templates, which when added to the real images improves the performance for image manipulation detection. This template set is estimated using certain constraints and any template can be added onto the image right after it is being captured by any camera. Our framework is able to achieve better image manipulation detection performance on different unseen GMs, compared to prior works. We also show the results on a diverse set of 12 additional GMs to demonstrate the generalizability of our proposed method.

Limitations. First, although our work aims to protect real images in a proactive manner and can detect whether an image has been manipulated or not, it cannot perform general deepfake detection on entirely synthesized images. Second, we try our best to collect a diverse set of GMs to validate the generalization of our approach. However, there are many other GMs that do not have open-sourced codes to be evaluated in our framework. Lastly, how to supervise the training of a

network for template selection is still an unanswered question.

Potential societal impact. We propose a proactive scheme which uses encrypted real images and their manipulated versions to perform manipulation detection. While this offers more generalizable detection, the encrypted real images might be used for training GMs in the future, which could make the manipulated images more robust against our framework, and thus warrents more research.

CHAPTER 3

MALP: MANIPULATION LOCALIZATION USING A PROACTIVE SCHEME

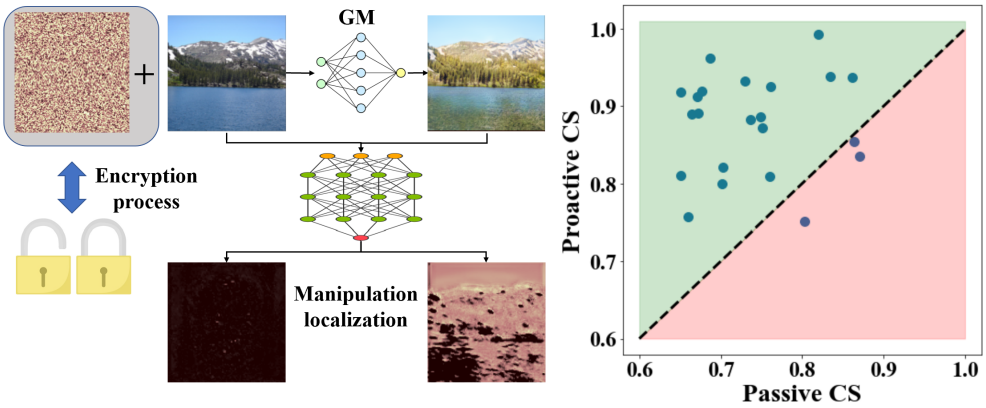
Advancements in the generation quality of various Generative Models (GMs) has made it necessary to not only perform binary manipulation detection but also localize the modified pixels in an image. However, prior works termed as *passive* for manipulation localization exhibit poor generalization performance over unseen GMs and attribute modifications. To combat this issue, we propose a *proactive* scheme for manipulation localization, termed MaLP. We encrypt the real images by adding a learned template. If the image is manipulated by any GM, this added protection from the template not only aids binary detection but also helps in identifying the pixels modified by the GM. The template is learned by leveraging local and global-level features estimated by a two-branch architecture. We show that MaLP performs better than prior passive works. We also show the generalizability of MaLP by testing on 22 different GMs, providing a benchmark for future research on manipulation localization. Finally, we show that MaLP can be used as a discriminator for improving the generation quality of GMs[1].

3.1 Introduction

We witness numerous Generative Models (GMs) [107, 304, 194, 52, 238, 385, 53, 214, 159, 157, 327, 113, 264, 162] being proposed to generate realistic-looking images. These GMs can not only generate an entirely new image [159, 157], but also perform partial manipulation of an input image [53, 194, 53, 385]. The proliferation of these GMs has made it easier to manipulate personal media for malicious use. Prior methods to combat manipulated media focus on binary detection [197, 279, 97, 40, 345, 6, 265, 340, 65, 8], using mouth movement, model parsing, hand-crafted features, *etc*..

Recent works go one step further than detection, *i.e.manipulation localization*, which is defined as follows: given a partially manipulated image by a GM (*e.g.*STGAN [194] modifying hair colors of a face image), the goal is to identify which pixels are modified by estimating a *fakeness*

[1] *Vishal Asnani*, Xi Yin, Tal Hassner, and Xiaoming Liu. "Malp: Manipulation localization using a proactive scheme." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023.



(a) **(b)**

Figure 3.1 **(a) High-level idea of MaLP.** We encrypt the image by adding a learnable template, which helps to estimate the fakeness map. **(b)** The cosine similarity (CS) between ground-truth and predicted fakeness maps for 22 unseen GMs. The performance is better for almost all GMs when using our proactive approach.

map [141]. Identifying modified pixels helps to determine the severity of the fakeness in the image, and aid media-forensics [141, 65]. Also, manipulation localization provides an understanding of the attacker's intent for modification which may further benefit identifying attack toolchains used [80].

Recent methods for manipulation localization [180, 287, 225] focus on estimating the manipulation mask of face-swapped images. They localize modified facial attributes by leveraging attention mechanisms [65], patch-based classifier [36], and face-parsing [141]. The main drawback of these methods is that they do not generalize well to GMs unseen in training. That is when the test images and training images are modified by different GMs, which will likely happen given the vast number of existing GMs. Thus, our work aims for a localization method generalizable to unseen GMs.

All aforementioned methods are based on a *passive* scheme as the method receives an image as is for estimation. Recently, proactive methods are gaining success for deepfake tasks such as detection [6], disruption [267, 356], and tagging [325]. These methods are considered *proactive* as they add different types of signals known as *templates* for encrypting the image before it is manipulated by a GM. This template can be one-hot encoding [325], adversarial perturbation [267], or a learnable noise [6], and is optimized to improve the performance of the defined tasks.

Motivated by [6], we propose a Proactive scheme for MAnipulation Localization, termed as MaLP, in order to improve generalization. Specifically, MaLP learns an optimized template which,

when added to real images, would improve manipulation localization, should they get manipulated. This manipulation can be done by an unseen GM trained on either in-domain or out-of-domain datasets. Furthermore, face manipulation may involve modifying facial attributes unseen in training (*e.g.*train on hair color modification yet test on gender modification). MaLP incorporates three modules that focus on encryption, detection, and localization. The encryption module selects and adds the template from the template set to the real images. These encrypted images are further processed by localization and detection modules to perform the respective tasks.

Designing a proactive manipulation localization approach comes with several challenges. First, it is not straightforward to formulate constraints for learning the template *unsupervisedly*. Second, calculating a fakeness map at the same resolution as the input image is computationally expensive if the decision for each pixel has to be made. Prior works [36, 65] either down-sample the images or use a patch-wise approach, both of which result in inaccurate low-resolution fakeness maps. Lastly, the templates should be generalizable to localize modified regions from unseen GMs.

We design a two-branch architecture consisting of a shallow CNN network and a transformer to optimize the template during training. While the former leverages local-level features due to its shallow depth, the latter focuses on global-level features to better capture the affinity of the far-apart regions. The joint training of both networks enables the MaLP to learn a better template, having embedded the information of both levels. During inference, the CNN network alone is sufficient to estimate the fakeness map with a higher inference efficiency. Compared to prior passive works [141, 65], MaLP improves the generalization performance on unseen GMs. We also demonstrate that MaLP can be used as a discriminator for fine-tuning conventional GMs to improve the quality of GM-generated images.

In summary, we make the following contributions.

- We are the first to propose a proactive scheme for image manipulation localization, applicable to both face and generic images.
- Our novel two-branch architecture uses both local and global level features to learn a set

Table 3.1 Comparison of our approach with prior works on manipulation localization and proactive schemes. We show the generalization ability of all works across different facial attribute modifications, unseen GMs trained on datasets with the same domain (in-domain) and different domains (out-domain). [Keys: Attr.: Attributes, Imp.: Improving, L.: Localization, D.: Detection].

Work	Scheme	Task	Template	Generalization			Imp. GM
				Attr.	In-domain	Out-domain	
[325]	Proactive	Tag	Fix	✔	✔	✖	✖
[272]	Proactive	Disrupt	Learn	✔	✖	✖	✖
[267]	Proactive	Disrupt	Learn	✔	✔	✖	✖
[356]	Proactive	Disrupt	Learn	✔	✖	✖	✖
[6]	Proactive	D.	Learn	✖	✔	✔	✖
[225]	Passive	L. + D.	-	✖	✖	✖	✖
[287]	Passive	L. + D.	-	✖	✖	✖	✖
[180]	Passive	L. + D.	-	✖	✔	✖	✖
[65]	Passive	L. + D.	-	✔	✔	✖	✖
[36]	Passive	L. + D.	-	✖	✔	✖	✖
[141]	Passive	L. + D.	-	✔	✔	✖	✖
MaLP	Proactive	L. + D.	Learn	✔	✔	✔	✔

of templates in an unsupervised manner. The framework is guided by constraints based on template recovery, fakeness maps classification, and high cosine similarity between predicted and ground-truth fakeness maps.

- MaLP can be used as a plug-and-play discriminator module to fine-tune the generative model to improve the quality of the generated images.

- Our method outperforms State-of-The-Art (SoTA) methods in manipulation localization and detection. Furthermore, our method generalizes well to GMs and modified attributes unseen in training. To facilitate the research of localization, we develop a benchmark for evaluating the generalization of manipulation localization, on images where the train and test GMs are different.

3.2 Related Work

Manipulation Localization. Prior works tackle manipulation localization by adopting a passive scheme. Some of them focus on forgery attacks like removal, copy-move, and splicing using multi-task learning [225]. Songsri-in *et al.* [287] leverage facial landmarks [54] for manipulation localization. Li *et al.* [180] estimate the blended boundary for forged face-swap images. [65]

uses an attention mechanism to leverage the relationship between pixels and [36] uses a patch-based classifier to estimate modified regions. Recently, Huang *et al*. [141] utilize gray-scale maps as ground truth for manipulation localization and leverage face parsing with an attention mechanism for prediction. The passive methods discussed above suffer from the generalization issue [225, 54, 287, 65, 36, 141] and estimate a low-resolution fakeness map [65] which is less accurate for the localization purpose. MaLP generalizes better to modified attributes and GMs unseen in training.

Proactive Scheme. Recently, proactive schemes are developed for various tasks. Wang *et al*. [325] leverage the recovery of embedded one-hot encoding messages to perform deepfake tagging. A small perturbation is added onto the images by Segalis *et al*. [272] to disrupt the output of a GM. The same task is performed by Ruiz *et al*. [267] and Yeh *et al*. [356], both adding adversarial noise onto the input images. Asnani *et al*. [6] propose a framework based on adding a learnable template to input images for generalized manipulation detection. Unlike prior works, which focus on binary detection, deepfake disruption, or tagging, our work emphasizes on manipulation localization. We show the comparison of our approach with prior works in Tab. 3.1.

Manipulation Detection. The advancement in manipulation detection keeps reaching new heights. Prior works propose to combat deepfakes by exploiting frequency domain patterns [330], up-sampling artifacts [371], model parsing [8, 353], hand-crafted features [222], lip motions [265], unified detector [70] and self-attention [65]. Recent methods use self-blended images [279], hierarchical localization features [116], real-time deviations [97], and self-supervised learning with adversarial training [40]. Finally, methods based on contrastive learning [345] and proactive scheme [6] have explicitly focused on generalized manipulation detection across unknown GMs.

3.3 Proposed Approach

3.3.1 Problem Formulation

Passive Manipulation Localization Let $\boldsymbol{I}^R$ be a set of real images that are manipulated by a GM G to output the set of manipulated images $G(\boldsymbol{I}^R)$. Prior passive works perform manipulation

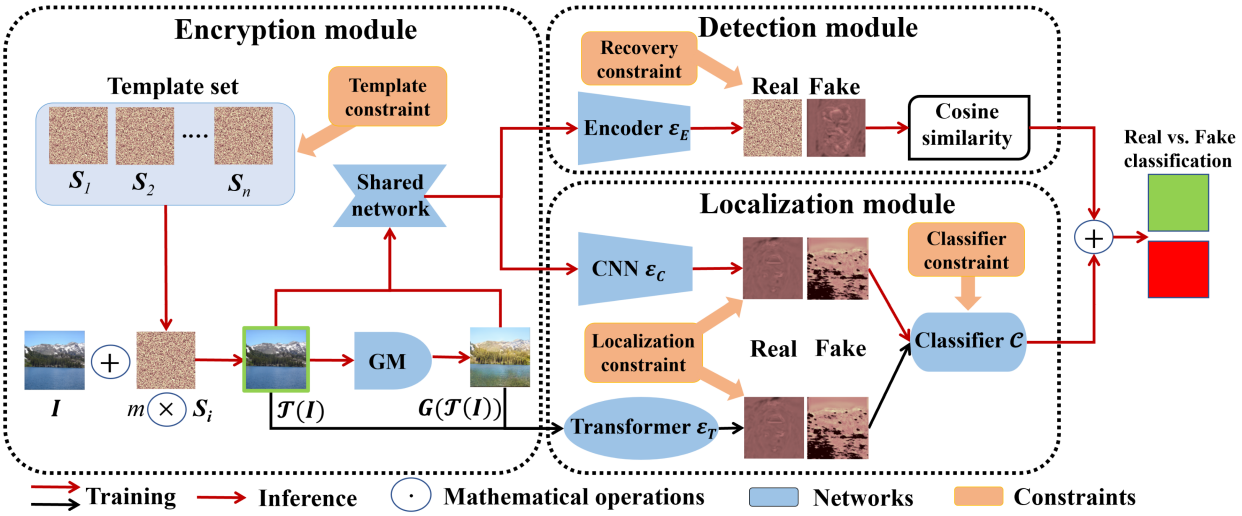


Figure 3.2 **The overview of MaLP.** It includes three modules: encryption, localization, and detection. We randomly select a template from the template set and add it to the real image as encryption. The GM is used in inference mode to manipulate the encrypted image. The detection module recovers the added template for binary detection. The localization module uses a two-branch architecture to estimate the fakeness map. Lastly, we apply the classifier to the fakeness map to better distinguish them from each other. Best viewed in color.

localization by estimating the fakeness map $\boldsymbol{M}_{pred}$ with the following objective:

$$\min_{\theta_{\mathcal{E}}} \left\{ \sum_{j} \left(\left\| \mathcal{E}(G(\boldsymbol{I}_j^R); \theta_{\mathcal{E}}) - \boldsymbol{M}_{GT} \right\|_2 \right) \right\}, \tag{3.1}$$

where $\mathcal{E}$ denotes the passive framework with parameters $\theta_{\mathcal{E}}$ and $\boldsymbol{M}_{GT}$ is the ground-truth fakeness map.

To represent the fakeness map, some prior methods [287, 65, 180] choose a binary map by applying a threshold on the difference between the real and manipulated images. This is undesirable as the threshold selection is highly subjective and sensitive, leading to inaccurate fakeness maps. Therefore, we adopt the continuous gray-scale map for calculating the ground-truth fakeness maps [141], formulated as:

$$\boldsymbol{M}_{GT} = Gray(|\boldsymbol{I}^R - G(\boldsymbol{I}^R)|)/255, \tag{3.2}$$

where $Gray(.)$ converts the image to gray-scale.

Proactive Scheme Asnani *et al.* [6] define adding the template as a transformation $\mathcal{T}$ applied to images $\boldsymbol{I}^R$, resulting in the encrypted images $\mathcal{T}(\boldsymbol{I}^R)$. The added template acts as a signature of the

defender and is learned during the training, aiming to improve the performance of the task at hand, *e.g.*detection, disruption, and tagging. Motivated by [6] that uses multiple templates, we have a set of n orthogonal templates $\mathcal{S} = \{\boldsymbol{S}_1, \boldsymbol{S}_2, ...\boldsymbol{S}_n\}$ where $\boldsymbol{S}_i \in \mathbb{R}^{128\times128}$, for a real image $\boldsymbol{I}_j^R \in \boldsymbol{I}^R$, transformation $\mathcal{T}$ is defined as:

$$\mathcal{T}(\boldsymbol{I}_j^R; \boldsymbol{S}_i) = \boldsymbol{I}_j^R + \boldsymbol{S}_i, \text{ where } i \in \{1, 2, ..., n\}. \tag{3.3}$$

The templates are optimized such that adding them to the real images wouldn't result in a noticeable visual difference, yet helps manipulation localization.

Proactive Manipulation Localization. Unlike the passive schemes [141, 65, 225, 180], we learn an optimal template set to help manipulation localization. For the encrypted images $\mathcal{T}(\boldsymbol{I}^R)$, we formulate the estimation of the fakeness map as:

$$\min_{\theta_{\mathcal{E}_P}, \boldsymbol{S}_i} \left\{ \sum_j \left(\left\| \mathcal{E}_P(G(\mathcal{T}(\boldsymbol{I}_j^R; \boldsymbol{S}_i)); \theta_{\mathcal{E}_P}) - \boldsymbol{M}_{GT} \right\|_2 \right) \right\}. \tag{3.4}$$

where $\mathcal{E}_P$ is the proactive framework with parameters $\theta_{\mathcal{E}_P}$.

However, as the output of the GM has changed from images in set $G(\boldsymbol{I}^R)$ to images in set $G(\mathcal{T}(\boldsymbol{I}^R))$, in our proactive approach, the calculation of the ground-truth fakeness map shall be changed from Eq. (3.2) to the follows:

$$\boldsymbol{M}_{GT} = Gray(|\boldsymbol{I}^R - G(\mathcal{T}(\boldsymbol{I}^R))|)/255. \tag{3.5}$$

3.3.2 Manipulation Localization

MaLP consists of three modules: encryption, localization, and detection. The encryption module is used to encrypt the real images. The localization module estimates the fakeness map using a two-branch architecture. The detection module performs binary detection for the encrypted and manipulated images by recovering the template and using the classifier in the localization module. All three modules, as detailed next, are trained in an end-to-end manner.

3.3.2.1 Encryption Module

Following the procedure in [6], we add a randomly selected learnable template from the template set to a real image. We control the strength of the added template using a hyperparameter m, which

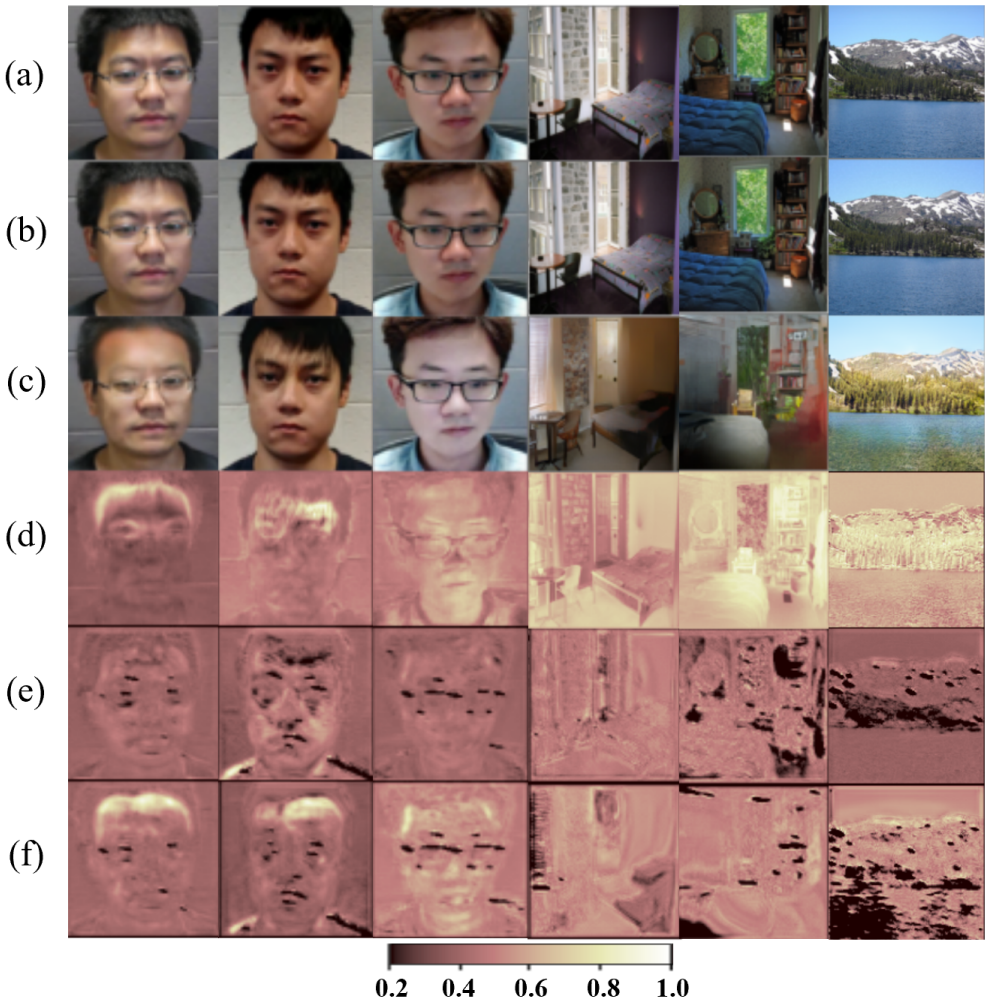


Figure 3.3 Visualization of fakeness maps for faces and generic images showing generalization across unseen attribute modifications and GMs: (a) real image, (b) encrypted image, (c) manipulated image, (d) $\boldsymbol{M}_{GT}$, (e) predicted fakeness map for encrypted images, and (f) predicted fakeness map for manipulated images. The first column shows the manipulation of (seen GM, seen attribute modification) *i.e.*(STGAN, bald). Following two columns show the manipulation of (seen GM, unseen attribute modification) *i.e.*(STGAN, [bangs, pale skin]. The fourth and fifth columns show manipulation of unseen GM, GauGAN for non-face images. The last column shows manipulation by unseen GM, DRIT. We see that the fakeness map of manipulated images is more bright and similar to $\boldsymbol{M}_{GT}$, while the real fakeness map is more close to zero. We use the cmap as “pink" to better visualize the fakeness map. All face images come from SiWM-v2 data [115].

prevents the degradation of the image quality. The encryption process is summarised below:

$$\mathcal{T}(\boldsymbol{I}_j^R) = \boldsymbol{I}_j^R + m \times \boldsymbol{S}_i \text{ where } i = Rand(1, 2, ..., n). \tag{3.6}$$

We select the value of m as 30% for our framework.

We optimize the template set by focusing on properties like low magnitude, orthogonality, and

high-frequency content [6]. The properties are applied as constraints as follows.

$$J_T = \lambda_1 \times \sum_{i=1}^{n} ||\boldsymbol{S}_i||_2 + \lambda_2 \times \sum_{\substack{i,j=1 \\ i \neq j}}^{n} \mathrm{CS}(\boldsymbol{S}_i, \boldsymbol{S}_j) + \lambda_3 \times ||\mathcal{L}(\mathfrak{F}(\boldsymbol{S}))||_2, \tag{3.7}$$

where CS is the cosine similarity, $\mathcal{L}$ is the low-pass filter, $\mathfrak{F}$ is the fourier transform, λ_1, λ_2, λ_3 are weights for losses of low magnitude, orthogonality and high-frequency content, respectively.

3.3.2.2 Localization Module

To design the localization module, we consider two desired properties: a larger receptive field for fakeness map estimation and high inference efficiency. A network with a large receptive field will consider far-apart regions in learning features for localization. Yet, large receptive fields normally come from deeper networks, implying slower inference.

In light of these properties, we design a two-branch architecture consisting of a shallow CNN network $\mathcal{E}_C$ and a ViT transformer [79] $\mathcal{E}_T$ (see Fig. 3.2). The intuition is to have one shallow branch to capture local features, and one deeper branch to capture global features. While training with both branches helps to learn better templates, in inference we only use the shallow branch for a higher efficiency. Specifically, the shallow CNN network has 10 layers which is efficient in inference but can only capture the local features due to small receptive fields. To capture global information, we adopt the ViT transformer. With the self-attention between the image patches, the transformer can estimate the fakeness map considering the far-apart regions.

Both the CNN and transformer are trained jointly to estimate a better template set, resembling the concept of the ensemble of networks. We empirically show that training both networks simultaneously results in higher performance than training either network separately. As the shallow CNN network is much faster in inference than the transformer, we use the transformer only in training to optimize the templates and switch off the transformer branch in inference.

To estimate the fakeness map, we leverage the supervision of the ground-truth fakeness map in Eq. (3.5). For fake images, we maximize the cosine similarity (CS) and structural similarity index measure (SS) between the predicted and ground-truth fakeness map. However, the fakeness map should be a zero image for encrypted images. Therefore, we apply an L_2 loss [141] to minimize the predicted map to zero for encrypted images. To maximize the difference between the two fakeness

maps, we further minimize the cosine similarity between the predicted map from encrypted images and $\boldsymbol{M}_{GT}$. The localization loss is defined as:

$$J_L = \begin{cases} \Big\{\lambda_4 \times ||\mathcal{E}_{C/T}(\boldsymbol{I})||_2^2 + & \text{if } \boldsymbol{I} \in \mathcal{T}(\boldsymbol{I}^R) \\ \lambda_5 \times \mathrm{CS}(\mathcal{E}_{C/T}(\boldsymbol{I}), \boldsymbol{M}_{GT})\Big\} & \\ \Big\{\lambda_6 \times (1 - \mathrm{CS}(\mathcal{E}_{C/T}(\boldsymbol{I}), \boldsymbol{M}_{GT})) + & \text{if } \boldsymbol{I} \in G(\mathcal{T}(\boldsymbol{I}^R)) \\ \lambda_7 \times (1 - SS(\mathcal{E}_{C/T}(\boldsymbol{I}), \boldsymbol{M}_{GT}))\Big\}. & \end{cases} \tag{3.8}$$

Finally, we have a classifier to make a binary decision of real *vs*.fake using the fakeness maps. This classifier is included in the framework to aid the detection module for binary detection of the input images, which will be discussed in Sec. 3.3.2.3. Another reason to have the classifier is to make the fakeness maps from encrypted and fake images to be distinguishable. We find that this design allows our training to converge much faster.

3.3.2.3 Detection Module

To leverage the added template for manipulation detection, we perform template recovery using encoder $\mathcal{E}_E$. We follow the procedure in [6] to recover the added template from the encrypted images by maximizing the cosine similarity between $\boldsymbol{S}$ and $\boldsymbol{S}_R$. However, for manipulated images, we minimize the cosine similarity between the recovered template ($\boldsymbol{S}_R$) and all the templates in the template set $\mathcal{S}$.

$$J_R = \begin{cases} \lambda_8 \times (1 - \mathrm{CS}(\boldsymbol{S}, \boldsymbol{S}_R)) & \text{if } x \in \mathcal{T}(\boldsymbol{I}^R) \\ \lambda_9 \times (\sum_{i=1}^{n} (\mathrm{CS}(\boldsymbol{S}_i, \boldsymbol{S}_R))) & \text{if } x \in G(\mathcal{T}(\boldsymbol{I}^R)). \end{cases} \tag{3.9}$$

Further, we leverage our estimated fakeness map to help manipulation detection. As discussed in the previous section, we apply a classifier C to perform binary classification of the predicted fakeness map for the encrypted and fake images. The logits of the classifier are further combined with the cosine similarity of the recovered template. The averaged logits are back-propagated using the binary cross-entropy constraint. This not only improves the performance of manipulation detection but also helps manipulation localization. Therefore, we apply the binary cross entropy

Table 3.2 Manipulation localization comparison with prior works.

Method	Localization			Detection		
	CS ↑	PSNR ↑	SSIM ↑	Accuracy ↑	EER ↓	AUC ↑
[65]	0.6230	6.214	0.2178	0.9975	**0.0050**	0.9975
[141]	0.8831	22.890	**0.7876**	0.9945	0.0077	0.9998
MaLP	**0.9394**	**23.020**	0.7312	**0.9991**	0.0072	**1.0**

loss on the averaged logits as follows:

$$J_C = \lambda_{10} \times -\sum_j \Big\{ y_j . \log\Big[\frac{C(\boldsymbol{X}_j) + CS(\boldsymbol{S}_R, \boldsymbol{S})}{2}\Big] - (1 - y_j) . \log\Big[1 - \frac{C(\boldsymbol{X}_j) + CS(\boldsymbol{S}_R, \boldsymbol{S})}{2}\Big]\Big\}, \tag{3.10}$$

where y_j is the class label, $\boldsymbol{S}$ and $\boldsymbol{S}_R$ are the added and recovered template respectively.

Our framework is trained in an end-to-end manner with the overall loss function as follows:

$$J = J_T + J_R + J_C + J_L. \tag{3.11}$$

3.3.3 MaLP as A Discriminator

One application of MaLP is to leverage our proposed localization module as a discriminator for improving the quality of the manipulated images. MaLP performs binary classification by estimating a fakeness map, which can be used as an objective. This results in output images being resilient to manipulation localization, thereby lowering the performance of our framework.

We use MaLP as a plug-and-play discriminator to improve image generation quality through fine-tuning pretrained GMs. The generation quality and manipulation localization will compete head-to-head, resulting in a better quality of the manipulated images. We define the fine-tuning objective for the GM as follows:

$$\min_{\theta_G} \max_{\theta_{MaLP}, \boldsymbol{S}_i} \Big\{ \sum_j \Big(\mathbb{E}\big[log(\mathcal{E}_{MaLP}(\mathcal{T}(\boldsymbol{I}_j^R)); \theta_{MaLP})\big] + \mathbb{E}\big[1 - log(\mathcal{E}_{MaLP}(G(\mathcal{T}(\boldsymbol{I}_j^R; \boldsymbol{S}_i); \theta_G); \theta_{MaLP}))\big]\Big)\Big\}. \tag{3.12}$$

where $\mathcal{E}_{MaLP}$ is our framework with θ_{MaLP} parameters.

Table 3.3 Comparison of localization performance across unseen GMs and attribute modifications. We train on STGAN bald/smile attribute modification and test on AttGAN/StyleGAN.

Method	Cosine similarity ↑(AttGAN)			Cosine similarity ↑ (StyleGAN)		
	Bald	Black Hair	Eyeglasses	Smile	Age	Gender
[141]	0.8141	0.6932	0.6950	0.6176	0.3141	0.6470
MaLP	**0.8201**	**0.7940**	**0.8557**	**0.8159**	**0.8255**	**0.8016**

3.4 Experiments

3.4.1 Experimental Setup

Settings Following the settings in [141], we use STGAN [194] to manipulate images from CelebA [199] dataset and train on bald facial attribute modification. In order to evaluate the generalization of image manipulation localization, we construct a new benchmark that consists of 200 real images of 22 different GMs on various data domains. The real images are chosen from the dataset on which the GM is trained on. The list of GMs, datasets and implementation details are provided in the supplementary.

Evaluation Metrics We use cosine similarity (CS), peak signal-to-noise ratio (PSNR), and structural similarity index measure (SSIM) as adopted by [141] to evaluate manipulation localization since the GT is a continuous map. For binary detection, we use the area under the curve (AUC), equal error rate (EER), and accuracy score [141].

3.4.2 Comparison with Baselines

We compare our results with [141] and [65] for manipulation localization. The results are shown in Tab. 3.2. MaLP has higher cosine similarity and similar PSNR for localization compared to [141]. However, we observe a dip in SSIM. This might be because of the degradation caused by adding our template to the real images and then performing the manipulation. The learned template helps localize the manipulated regions better, as demonstrated by cosine similarity, but the degradation affects SSIM and PSNR. We also compare the performance of real *vs*.fake binary detection. As expected, our proposed proactive approach outperforms the passive methods with a perfect AUC and near-perfect accuracy. We also show visual examples of fakeness maps for images modified by unseen GMs in Fig. 3.3. MaLP is able to estimate the fakeness map for unseen

Table 3.4 Benchmark for manipulation localization across 22 different unseen GMs, showing cosine similarity between ground-truth and predicted fakeness maps. We compare our proactive *vs*.passive baselines [36, 127, 65] approach to highlight the generalization ability of our MaLP. We scale the images to 128^2 for "sc." and keep the resolution as is for "no sc.".

GM	SEAN [387]	StarGAN [52]	CycleGAN [385]	GauGAN [238]	Con_Enc. [240]	StarGAN2 [53]	ALAE [245]	BiGAN [386]	AuGAN [385]	GANim [248]	DRGAN [304]	ILVR [50]
Resolution	256^2	128^2	256^2	256^2	128^2	256^2	256^2	256^2	340^2	128^2	128^2	256^2
ResNet50 [127]	0.8614	0.7513	0.6715	0.7615	**0.8639**	0.8196	0.6766	0.6514	0.6639	0.6871	**0.8029**	0.7018
[36]	0.7514	0.7111	0.7981	0.8016	0.7894	0.7026	0.7156	0.7217	0.7516	0.7612	0.7115	0.7851
[65]	0.7961	0.7887	0.8014	0.8256	0.8541	0.7034	0.7549	0.7805	0.7232	0.8457	0.7239	0.7854
MaLP (sc.)	**0.9376**	**0.8718**	0.9128	**0.9251**	0.8546	**0.8836**	**0.9192**	0.9181	0.8894	**0.9625**	0.7512	0.8003
MaLP (no sc.)	0.9258	**0.8718**	**0.9245**	0.9125	0.8546	0.8785	0.9141	**0.9229**	**0.9149**	**0.9625**	**0.7512**	**0.8359**

GM	DRIT [177]	Pix2Pix [144]	CounGAN [232]	DualGAN [357]	ESRGAN [333]	UNIT [195]	MUNIT [139]	ColGAN [223]	GDWCT [49]	RePaint [201]	Average
Resolution	256^2	256^2	128^2	256^2	1024^2	512×931	256×512	128^2	128^2	256^2	-
ResNet50 [127]	0.7486	0.6719	0.7293	0.7365	**0.8703**	0.7083	0.6601	0.7596	0.8350	0.6512	0.7401
[36]	0.7871	0.7769	0.8146	0.7569	0.8168	0.8064	0.6788	0.7610	0.8691	0.7516	0.7645
[65]	0.8120	0.7781	0.8559	0.7721	0.8241	0.8086	0.7097	0.7874	0.8879	0.7696	0.7903
MaLP (sc.)	0.8867	**0.8915**	**0.9326**	**0.8872**	0.8348	0.8214	0.7565	**0.8096**	**0.9384**	0.8102	0.8725
MaLP (no sc.)	**0.9084**	0.8714	**0.9326**	0.8432	**0.8743**	0.8391	**0.7860**	**0.8096**	**0.9384**	**0.8290**	**0.8773**

modifications and GMs across face/generic image datasets.

3.4.3 Generalization

Across Attribute Modifications Following the settings in [141], we evaluate the performance of MaLP across unseen attribute modifications. Specifically, we train MaLP using STGAN with the bald/smile attribute modification and test it on unseen attribute modifications with unseen GMs: AttGAN/StyleGAN. As shown in Tab. 3.3, MaLP is more generalizable to all unseen attribute modifications. Furthermore, AttGAN shares the high-level architecture with STGAN but not with StyleGAN. We observe a significant increase in localization performance for StyleGAN compared to AttGAN. This shows that, unlike our MaLP, passive works perform much worse if the test GM doesn't share any similarity with the training GM.

Across GMs Although [141] tries to show generalization across unseen GMs; it is limited by the GMs within the same domain of the dataset used in training. We propose a benchmark to evaluate the generalization performance for future manipulation localization works that consists of 22 different GMs in various domains. We select GMs that are publicly released and can perform partial manipulation.

As no open-source code base is available for [141], we train a passive approach using a ResNet50 [127] network to estimate the fakeness map as the baseline for comparison. Further, we compare our approach with [36, 65]. Although [36, 65] estimate a fakeness map, it has at least 5× lower resolution compared to input images due to their patch-based methodology. For a fair comparison, we rescale their predicted fakeness maps to the resolution of $\boldsymbol{M}_{GT}$. We compare the

Table 3.5 FID score comparison for the application of our approach as a discriminator for improving the generation quality of the GM.

State	Fine-tune	StarGAN FID ↓
Before	–	60.49
After	G	**51.91**
	$G + MaLP$	52.07

cosine similarity in Tab. 3.4. MaLP is able to outperform all the baselines for almost all GMs, which proves the effectiveness of the proactive scheme.

We also evaluate the performance of $\mathcal{E}_C$ for high-resolution images. For encryption, we upsample the 128×128 template to the original resolution of images and evaluate $\mathcal{E}_C$ on these higher resolution encrypted images. We observe similar performance of $\mathcal{E}_C$ for higher resolution images in Tab. 3.4, proving the versatility of $\mathcal{E}_C$ to image sizes.

3.4.4 Improving Quality of GMs

We fine-tune the GM into fooling our framework to generate a fakeness map as a zero image. This process results in better-quality images. Initially, we train MaLP with the pretrained GM so that it can perform manipulation localization. Next, to fine-tune the GM, we adopt two strategies. First, we freeze MaLP and fine-tune the GM only. Second, we fine-tune both the GM and the MaLP but update the MaLP with a lower learning rate. The result for fine-tuning StarGAN is shown in Tab. 3.5. We observe that for both strategies, MaLP reduces the FID score of StarGAN. We also show some visual examples in Fig. 3.4. We see that the images are of better quality after fine-tuning, and many artifacts in the images manipulated by the pretrained model are removed.

3.4.5 Other Comparisons

Binary Detection We compare with prior proactive and passive approaches for binary manipulation detection [6, 330, 222, 371]. We adopt the evaluation protocol in [6] to test on images manipulated by CycleGAN, StarGAN, and GauGAN. We are able to perform similar to [6] as shown in Tab. 3.6. We have better average precision than passive schemes and generalize well to GMs unseen in training. We also conduct experiments to see whether localization can help binary detection to improve the performance, as mentioned in Sec. 3.3.2.3. The combined predictions' results are better than just using the detection module as shown in Tab. 3.6. This is intuitive as the localization

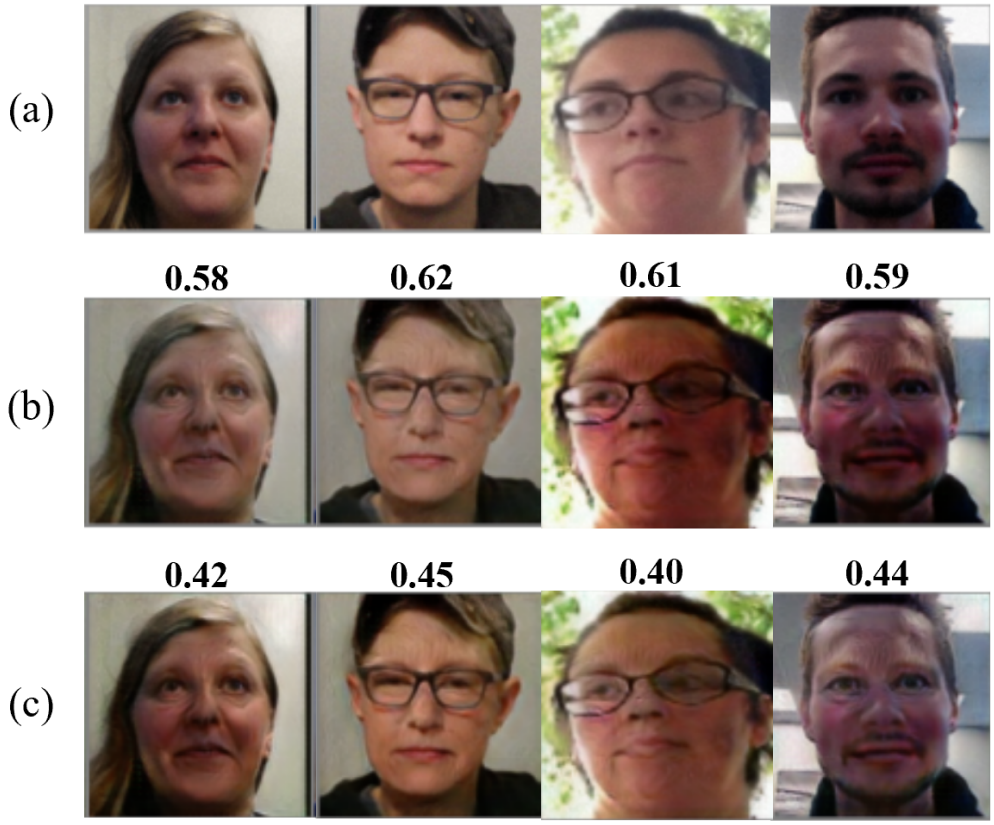


Figure 3.4 Visualization of (a) encrypted images, (b) manipulated images before fine-tuning, and (c) manipulated images after fine-tuning. The generation quality has improved after we fine-tune the GM using our framework as a discriminator. The artifacts in the images have been reduced, and the face skin color is less pale and more realistic. We also specify the cosine similarity of the predicted fakeness map and $\boldsymbol{M}_{GT}$. The GM is able to decrease the performance of our framework after fine-tuning. All face images come from SiWM-v2 data [115].

module provides extra information, thereby increasing the performance.

Inference Speed We compare the inference speed of our MaLP against prior work. [141] uses Deeplabv3-ResNet101 model from PyTorch [239]. In our generalization benchmark shown in Sec. 3.4.3, we use the ResNet50 model for training the passive baseline. The inference speed per image on an NVIDIA K80 GPU for Deeplabv3-ResNet101, ResNet50, and MaLP are 75.61, 52.66, and 29.26 ms, respectively. MaLP takes less than half the inference time compared to [141] due to our shallow CNN network.

Adversarial Attack Our framework can be considered as an adversarial attack on real images to aid manipulation localization. Therefore, it is vital to contrast the performance between our approach and classic adversarial attacks. For this purpose, we perform experiments that make use of

Table 3.6 Comparison with prior binary detection works. [Keys: D.M.: Detection module, L.M.: Localization module].

Method	Train GM	Set size	Test GM Average precision (%)↑		
			CycleGAN	StarGAN	GauGAN
Nataraj *et al*. [222]	CycleGAN	-	**100**	88.20	56.20
Zhang *et al*. [371]	AutoGAN	-	**100**	**100**	61.00
Wang *et al*. [330]	ProGAN	-	84.00	**100**	67.00
Asnani *et al*. [6]	STGAN	1	94.00	**100**	69.50
MaLP (D.M.)	STGAN	1	94.10	**100**	69.61
MaLP (D.M. + L.M.)	STGAN	1	94.30	**100**	**72.16**

Table 3.7 Comparison with adversarial attack methods.

Method	Scheme	Cosine similarity↑		
		Bald	Black Hair	Eyeglasses
Huang *et al*. [141]	Passive	0.8141	0.6932	0.6950
PGD [207]	Proactive	0.8051	0.7514	0.8358
FGSM [109]	Proactive	0.8111	0.7882	0.8512
CW [34]	Proactive	0.8014	**0.8344**	0.8405
MaLP	Proactive	**0.8201**	0.7940	**0.8557**

adversarial attacks, namely PGD [207], CW [34], and FGSM [109] to guide the learning of the added template. We evaluate on unseen GM AttGAN for unseen attribute modifications. We show the performance comparison in Tab. 3.7. MaLP has higher cosine similarity across some unseen facial attribute modifications compared to adversarial attacks. This can be explained as the adversarial attack methods being over-fitted to training parameters (data, target network *etc*.). Therefore, if the testing data is changed with unseen attribute modifications by GMs, the performance of adversarial attacks degrades. Further, these attacks are analogous to our MaLP as a proactive scheme which, in general, have better performance than passive works.

Model Robustness Against Degradations It is necessary to test the robustness of our proposed approach against various types of real-world image editing degradations. We evaluate our method on degradations applied during testing as adopted by [141], which include JPEG compression, blurring, adding noise, and low resolution. The results are shown in Fig. 3.5. Our proposed MaLP is more robust to real-world degradations than passive schemes.

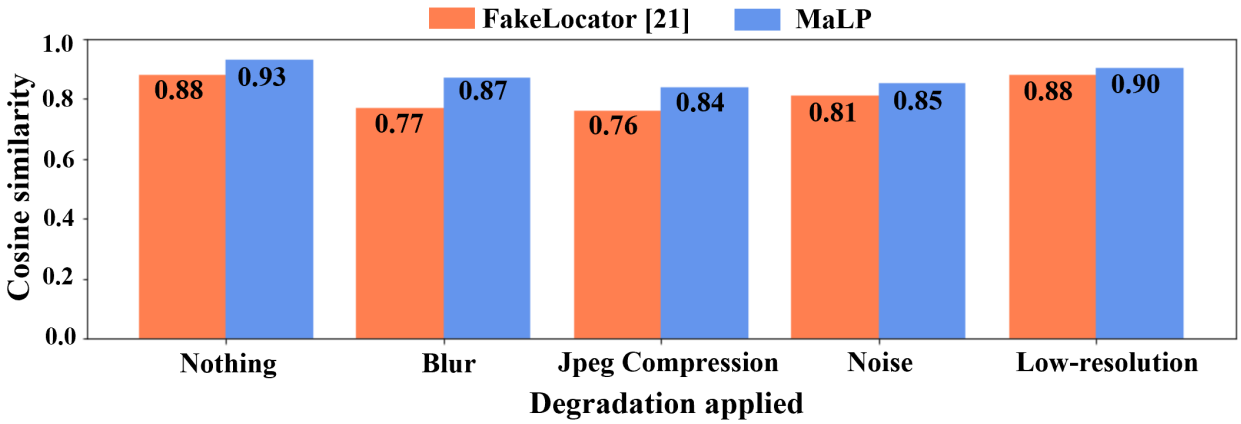


Figure 3.5 Comparison of our approach's robustness against common image editing degradations.

3.4.6 Ablations

Two-branch Architecture As described in Sec. 3.3.2.2, MaLP adopts a two-branch architecture to predict the fakeness map using the local-level and global-level features, which are estimated by a shallow CNN and a transformer. We ablate by training each branch separately to show the effectiveness of combining them. As shown in Tab. 3.8, if the individual network is trained separately, the performance is lower than the two-branch architecture. Next, to show the efficacy of the transformer, we use a ResNet50 network in place of the transformer to predict the fakeness map. We observe that the performance is even worse than using only the transformer. ResNet50 lacks the added advantage of self-attention in the transformer, which estimates the global-level features much better than a CNN network.

Constraints MaLP leverages different constraints to estimate the fakeness map using an optimized template. We perform an ablation by removing each constraint separately, showing the importance of every constraint. Tab. 3.9 shows the cosine similarity for localization and accuracy for detection. Removing either the classifier or recovery constraint results in lower detection performance. This is expected as we leverage logits from both C and $\mathcal{E}_E$, and removing the constraint for one network will hurt the logits of the other network. Furthermore, removing the template constraint results in a decrease in performance. Although the gap is small, the template is not properly optimized to have lower magnitude and high-frequency content.

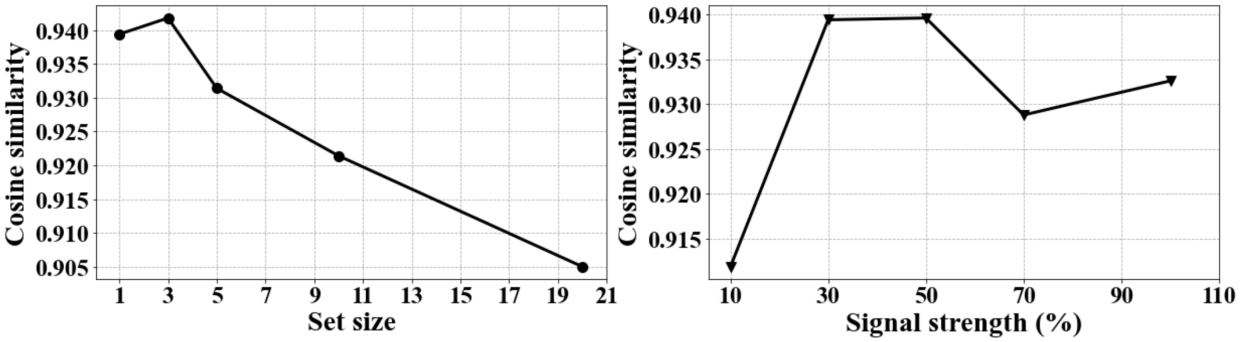
Removing the localization constraint and just applying a L_2 loss for supervising fakeness maps

Table 3.8 Ablation of two-branch architecture. CNN is a shallow network with 10 layers. Training each branch separately has worse localization results than combining them.

Network trained	Cosine similarity ↑	Accuracy ↑
CNN only	0.8961	0.9801
Transformer only	0.8848	0.9856
CNN + ResNet50	0.8647	0.9512
CNN + Transformer	**0.9394**	**0.9981**

Table 3.9 Ablation of constraints used in training our framework.

Constraint removed	Cosine similarity ↑	Accuracy ↑
Classifier constraint J_C	0.9319	0.9814
Template constraint J_T	0.9143	0.9803
Localization constraint J_L	0.8814	0.9539
Recovery constraintJ_R	0.9206	0.9780
Fixed template	0.8887	0.9514
Nothing (MaLP)	**0.9394**	**0.9991**



(a) (b)

Figure 3.6 Ablation study on hyperparameters used in our framework: set size and signal strength.

result in a significant performance drop for both localization and detection, showing the necessity of this constraint. Finally, we show the importance of a learnable template by not optimizing it during the training of MaLP. This hurts the performance a lot, similar to removing the localization constraint. Both these observations prove that our localization constraint and learnable template are important components of MaLP.

Template Set Size We perform an ablation to vary the size of the template set $\mathcal{S}$. Having multiple templates will improve security if an attacker tries to reverse engineer the template from encrypted images. The results are shown in Fig. 3.6 (a). The cosine similarity takes a dip when the set size is increased. We also observe the inter-template cosine similarity, which remains constant at a high

value of around 0.74 for all templates. This is against the findings of [6]. Localization is a more challenging task than binary detection. Therefore, it is less likely to find different templates for our MaLP in the given feature space compared to [6].

Signal Strength We vary the template strength hyperparameter m to find its impact on the performance. As shown in Fig. 3.6 (b), the cosine similarity increases as we increase the strength of the added template. However, this comes with the lower visual quality of the encrypted images if the template strength is increased. The performance doesn't vary much after $m = 30\%$, which we use for MaLP.

3.5 Conclusion

This paper focuses on manipulation localization using a proactive scheme (MaLP). We propose to improve the generalization of manipulation localization across unseen GM and facial attribute modifications. We add an optimal template onto the real images and estimate the fakeness map via a two-branch architecture using local and global-level features. MaLP outperforms prior works with much stronger generalization capabilities, as demonstrated by our proposed evaluation benchmark with 22 different GMs in various domains. We show an application of MaLP in fine-tuning GMs to improve generation quality.

Limitations First, the number of publicly available GMs is limited. More thorough testing on many different GMs might give more insights into the problem of generalizable manipulation localization. Second, we show that our MaLP can be used to fine-tune the GMs to improve image generation quality. However, it is based on the pretrained GM. Using our method to train a GM from scratch can be an interesting direction to explore in the future.

CHAPTER 4

PROBED: PROACTIVE OBJECT DETECTION WRAPPER

Previous research in $2D$ object detection focuses on various tasks, including detecting objects in generic and camouflaged images. These works are regarded as passive works for object detection as they take the input image as is. However, convergence to global minima is not guaranteed to be optimal in neural networks; therefore, we argue that the trained weights in the object detector are not optimal. To rectify this problem, we propose a wrapper based on proactive schemes, PrObeD, which enhances the performance of these object detectors by learning a signal. PrObeD consists of an encoder-decoder architecture, where the encoder network generates an image-dependent signal termed templates to encrypt the input images, and the decoder recovers this template from the encrypted images. We propose that learning the optimum template results in an object detector with an improved detection performance. The template acts as a mask to the input images to highlight semantics useful for the object detector. Finetuning the object detector with these encrypted images enhances the detection performance for both generic and camouflaged. Our experiments on MS-COCO, CAMO, COD10K, and NC4K datasets show improvement over different detectors after applying PrObeD[1].

4.1 Introduction

Generic $2D$ object detection (GOD) has improved from earlier traditional detectors [312, 313, 64, 87] to the deep-learning-based object detectors [260, 254, 43, 32, 117, 128]. Advancements in deep-learning-based methods underwent many architectural change over recent years, including one-stage [254, 256, 22, 255, 196, 189], two-stage [103, 102, 260], CNN-based [102, 254, 256, 22, 73, 91, 98, 63], transformer-based [32, 388], and diffusion-based [43] methods. All these methods aim to predict the $2D$ bounding box of the objects in the images and their category class.

Another emerging area related to generic object detection is camouflaged object detection [82, 81, 149, 178, 120, 122, 121] (COD). COD aims to detect and segment objects blended with the

[1] *Vishal Asnani*, Abhinav Kumar, Suya You, and Xiaoming Liu. "PrObeD: proactive object detection wrapper." Advances in Neural Information Processing Systems 36, 2024.

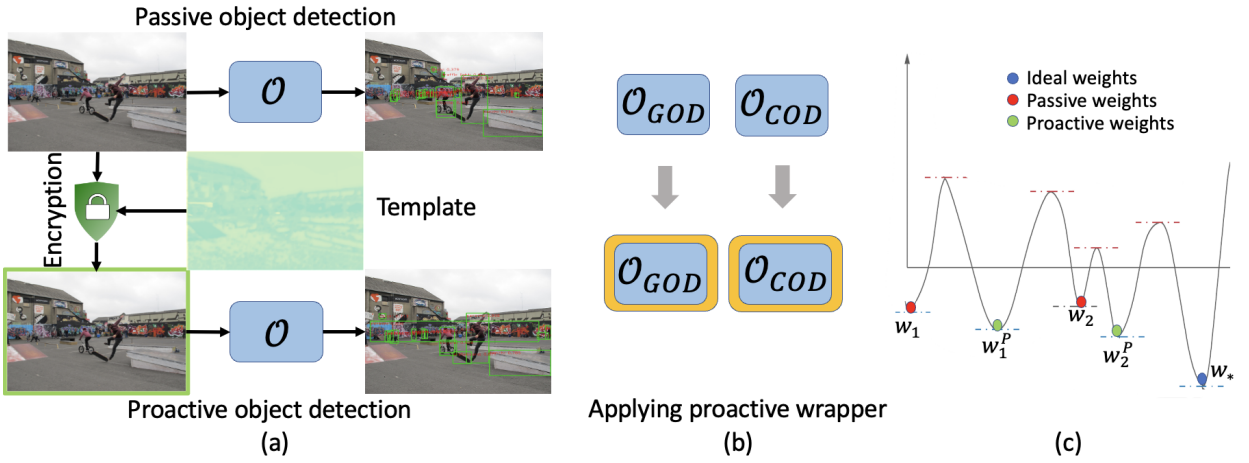


Figure 4.1 **(a) Passive *vs*. Proactive object detection**. A learnable template encrypts the input images, which are further used to train the object detector. **(b)** PrObeD serves as a wrapper on both generic and camouflaged object detectors, enhancing the detection performance. **(c)** For the linear regression model under additive noise and other assumptions, the converged weights of the proactive detector are closer to the optimal weights as compared to the converged weights of the passive detector. See Sec. 4.3.2 for details and proof.

background [82, 81] via object-level mask supervision. Applications of COD include medical [83, 193], surveillance [46] and autonomous driving [346]. Early COD detectors exploit hand-crafted features [275, 236] and optical flow [135], while current methods are deep-learning-based. These methods utilize attention [292, 38], joint learning [178], image gradient [149], and transformers [208, 348].

All these methods take input images as is for the detection task and hence are called passive methods. However, there is a line of research on proactive methods for a wide range of vision tasks such as disruption [267, 272], tagging [325], manipulation detection [6], and localization [7]. Proactive methods use signals, called templates, to encrypt the input images and pass the encrypted images as the input to the network. These are trained in an end-to-end manner by using either a fixed [325] or learnable template [267, 272, 7, 6] to improve the performance. A major advantage of proactive schemes is that such methods generalize better on unseen data/models [6, 7]. Motivated by this, we propose a plug-and-play Proactive Object Detection wrapper, PrObeD, to improve GOD and COD detectors.

Designing PrObeD as a proactive scheme involves several challenges and key factors. First, the proactive wrapper needs to be a plug-and-play module that can be applied to both GOD and COD

detectors. Secondly, the encryption process should be intuitive to benefit the object detection task. *e.g.*, an ideal template for detection should highlight the foreground objects in the input image. Lastly, the choice of supervision to estimate the template for encryption is hard to formulate.

Previous proactive methods [6, 7] use learnable but image-independent templates for manipulation and localization tasks. However, the object detection task is scene-specific; therefore, the ideal template should be image-dependent. Based on this key insight, we propose a novel plug-and-play proactive wrapper in which we apply object detectors to enhance detection performance. The PrObeD wrapper utilizes an encoder network to learn an image-dependent template. The learned template encrypts the input images by applying a transformation, defined as an element-wise multiplication between the template and the input image. The decoder network recovers the templates from the encrypted images. We utilize regression losses for supervision and leverage the ground-truth object map to guide the learning process, thereby imparting valuable object semantics to be integrated into the template. We then fine-tune the proactive wrapper with the GOD and COD detectors to improve their detection performance. Extensive experiments on MS-COCO, CAMO, COD10K, and NC4K datasets show that PrObeD improves the detection performance for both GOD and COD detectors.

In summary, the contributions of this work include:

- We propose a novel proactive approach *PrObeD* for the object detection task. To the best of our knowledge, this is the first work to develop a proactive approach to $2D$ object detection.
- We mathematically prove that the proactive method results in a better-converged model than the passive detector under assumptions and, consequently, a better object detector.
- PrObeD wraps around both GOD and COD detectors and improves detection performance on MS-COCO, CAMO, COD10K, and NC4K datasets

4.2 Related works

Proactive Schemes. Earlier works adopt to add signals like perturbation [272], adversarial noise [267], and one-hot encoding [325] messages while focusing on tasks like disruption [272, 267]

and deepfake tagging [325]. Asnani *et al*. [6] propose to learn an optimized template for binary detection by unseen generative models. Recently, MaLP [7] adds the learnable template to perform generalized manipulation localization for unknown generative models. Unlike these works, PrObeD uses image-dependent templates and is a plug-and-play wrapper for a different task of object detection.

Generic Object Detection Detection of generic objects, instead of specific object categories such as pedestrians [25], apples [55], and others [10, 170, 169], has been a long-standing objective of computer vision. RCNN [103, 104] employs the extraction of object proposals. He *et al*. [125] propose a spatial pooling layer to extract a fixed-length representation of all the objects. Modifications of RCNN [102, 260, 185, 367] increase the inference speed. Feature pyramid network [188] detects objects with a wide variety of scales. The above methods are mostly two-stage, so inference is an issue. Single-stage detectors like YOLO [254, 256, 22, 255, 316], SSD [196], HRNet [321] and RetinaNet [189] increase the speed and simplicity of the framework compared to the two-stage detector. Recently, transformer-based methods [32, 388] use a global-scale receptive field. Chen *et al*. [43] use diffusion models to denoise noisy boxes at every forward step. PrObeD functions as a wrapper around the pre-existing object detector, facilitating its transformation into an enhanced object detector. The comparison of PrObeD with prior works is summarized in Tab. 4.1.

Camouflaged Object Detection Early COD works rely on hand-crafted features like co-occurrence matrices [275], $3D$ convexity [236], optical flow [135], covariance matrix [150], and multivariate calibration components [259]. Later on, [292, 38] incorporate an attention-based cross-level fusion of multi-scale features to recover contextual information. Mei *et al*. [215] take motivation by predators to identify camouflaged objects using a position and focus ideology. SINet [82] uses a search and identification module to perform localization. SINET-v2[81] uses group-reversal attention to extract the camouflaged maps. [154] explores uncertainty maps and [389] utilizes cube-like architecture to integrate multi-layer features. ANet [176], LSR [204], and JCSOD [178] employ joint learning with different tasks to improve COD. Lately, [208, 348, 48] apply a transformer-based architecture for difficult-aware learning, uncertainty modeling, and temporal consistency. Zhai *et*

Table 4.1 Comparison of PrObeD with prior works.

Method	Proactive	Task	Template		COD	GOD	Plug-Play
			Number	Type			
Faster R-CNN [260]	×	Object Detection	-	-	×	✓	×
YOLO [254]	×	Object Detection	-	-	×	✓	×
DeTR [32]	×	Object Detection	-	-	×	✓	×
DGNet [149]	×	Object Detection	-	-	✓	×	×
SINet-v2 [81]	×	Object Detection	-	-	✓	×	×
JCSOD [178]	×	Object Detection	-	-	✓	×	×
OGAN [272]	✓	Disrupt	1	Learnable	-	-	×
Ruiz *et al*. [267]	✓	Disrupt	1	Learnable	-	-	×
Yeh *et al*. [356]	✓	Disrupt	1	Learnable	-	-	×
FakeTagger [325]	✓	Tagging	≥ 1	Fixed, Id-dependent	-	-	×
Asnani *et al*. [6]	✓	Manipulation Detection	≥ 1	Learnable set, Image-independent	-	-	✓
MaLP [7]	✓	Manipulation Localization	≥ 1	Learnable set, Image-independent	-	-	✓
PrObeD (Ours)	✓	Object Detection	≥ 1	Learnable, Image-dependent	✓	✓	✓

al. [368] use a graph learning model to disentangle input into different features for localization. DGNet [149] uses image gradients to exploit intensity changes in the camouflaged object from the background. Unlike these methods, PrObeD uses proactive methods to improve camouflaged object detection.

4.3 Proposed Approach

Our method originates from understanding what makes proactive schemes effective. We first overview the two detection problems: GOD and COD in Sec. 4.3.1. We next derive Lemma 1, where we show that the proactive schemes with the multiplicative transformation of images are better than passive schemes by comparing the deviation of trained network weights from the optimal. Based on this result, we derive that Average Precision (AP) from the proactive model is better than AP from the passive model in Theorem 1. At last, we present our proactive scheme-based wrapper, PrObeD, in Sec. 4.3.3, which builds upon the Theorem 1 to improve generic 2D objects and camouflaged detection.

4.3.1 Background

4.3.1.1 Passive Object Detection

Although generic $2D$ object detection and camouflage detection are similar problems, they have different objective functions. Therefore, we treat them as two different problems and define their objectives separately.

Generic 2D Object Detection. Let $\boldsymbol{I}_j$ be the set of input images given to the generic 2D object detector O with trainable parameters θ. Most of these detectors output two sets of predictions per image: (1) bounding box coordinates, $O(\boldsymbol{I}_j)_1 = \hat{T} \in \mathbb{R}^4$, (2) class logits, $O(\boldsymbol{I}_j)_2 = \hat{C} \in \mathbb{R}^C$, where N is the number of foreground object categories. If the ground-truth bounding box coordinates are T_j, and the ground-truth category label is C, the objective function of such detector is:

$$\min_{\theta} \left\{ \sum_{j} \left(||O(\boldsymbol{I}_j; \theta)_1 - T_j||_2 \right) - \sum_{j} \sum_{i=1}^{N} \left(C_j^i \cdot \log(O(\boldsymbol{I}_j; \theta)_2) \right) \right\}. \tag{4.1}$$

Camouflaged Object Detection. Let $\boldsymbol{I}_j$ be the input image set given to the camouflaged object detector O with trainable parameters θ, and $\boldsymbol{G}_j$ be the ground-truth segmentation map. Prior passive works predict a segmentation map with the following objective:

$$\min_{\theta} \left\{ \sum_{j} \left(\left\| O(\boldsymbol{I}_j; \theta) - \boldsymbol{G}_j \right\|_2 \right) \right\}. \tag{4.2}$$

4.3.1.2 Proactive Object Detection

Proactive schemes [7, 6] encrypt the input images with the template to aid manipulation detection/localization. Such schemes take an input image $\boldsymbol{I}_j \in \mathbb{R}^{H \times W \times 3}$ and learns a template $\boldsymbol{S}_j \in \mathbb{R}^{H \times W}$. PrObeD uses image-dependent templates to improve object detection. Given an input image $\boldsymbol{I}_j \in \mathbb{R}^{H \times W \times 3}$, PrObeD learns to output a template $\boldsymbol{S}_j \in \mathbb{R}^{H \times W}$, which can be used by a transformation $\mathcal{T}$ resulting in encrypted images $\mathcal{T}(\boldsymbol{I}_j)$. PrObeD uses element-wise multiplication as the transformation $\mathcal{T}$, which is defined as:

$$\mathcal{T}(\boldsymbol{I}_j) = \mathcal{T}(\boldsymbol{I}_j; \boldsymbol{S}_j) = \boldsymbol{I}_j \odot \boldsymbol{S}_j. \tag{4.3}$$

4.3.2 Mathematical Analysis of Passive and Proactive Detectors

PrObeD optimizes the template to improve the performance of the object detector. We argue that this template helps arrive at a better global minima representing the optimal parameters θ. We now define the following lemma to support our argument:

Lemma 1 ***Converged weights of proactive and passive detectors.*** *Consider a linear regression model that regresses an input image $\boldsymbol{I}_j$ under an additive noise setup to obtain the 2D coordinates. Assume the noise under consideration e is a normal random variable $\mathcal{N}(0, \sigma^2)$. Let $\boldsymbol{w}$ and $\boldsymbol{w}^*$*

denote the trained weights of the pretrained linear regression model and the optimal weights of the linear regression model. Also, assume SGD optimizes the model parameters with decreasing step size s such that the steps are square summable i.e., $\mathcal{S} = \lim_{t\to\infty} \sum_{k=1}^{t} s_k^2$ *exist, and the noise is independent of the image. Then, there exists a template* $\boldsymbol{S}_j \in [0,1]$ *for the image* $\boldsymbol{I}_j$ *such that the multiplicative transformation of images as the input results in a trained weight* $\boldsymbol{w}'$ *closer to the optimal weight than the originally trained weight* $\boldsymbol{w}$*. In other words,*

$$\mathbb{E}(||\boldsymbol{w}' - \boldsymbol{w}^*||_2) < \mathbb{E}(||\boldsymbol{w} - \boldsymbol{w}^*||_2). \tag{4.4}$$

The proof of Lemma 1 is in supplementary. We use the variance of the gradient of the encrypted images to arrive at this lemma. We next use Lemma 1 to derive the following theorem:

Theorem 1 ***AP comparison of proactive and passive detectors.*** *Consider a linear regression model that regresses an input image* $\boldsymbol{I}_j$ *under an additive noise setup to obtain the 2D coordinates. Assume the noise under consideration e is a normal random variable* $\mathcal{N}(0, \sigma^2)$*. Let* $\boldsymbol{w}$ *and* $\boldsymbol{w}^*$ *denote the trained weights of the pretrained linear regression model and the optimal weights of the linear regression model. Also, assume SGD optimizes the model parameters with decreasing step size s such that the steps are square summable i.e.,* $\mathcal{S} = \lim_{t\to\infty} \sum_{k=1}^{t} s_k^2$ *exist, and the noise is independent of the image. Then, the AP of the proactive detector is better than the AP of the passive detector.*

The proof of Theorem 1 is in the supplementary. We use the Lemma 1 and the non-decreasing nature of AP w.r.t. IoU to arrive at this theorem. Next, we adapt the objectives of Eqs. (4.1) and (4.2) to incorporate the proactive methods as follows:

$$\min_{\theta, \boldsymbol{S}_j} \left\{ \sum_j \left(||\mathcal{O}(\mathcal{T}(\boldsymbol{I}_j; \boldsymbol{S}_j); \theta)_1 - T_j||_2 \right) - \sum_j \sum_{i=1}^{N} \left(C_j^i \cdot \log(\mathcal{O}(\mathcal{T}(\boldsymbol{I}_j; \boldsymbol{S}_j); \theta)_2) \right) \right\}, \tag{4.5}$$

$$\min_{\theta, \boldsymbol{S}_j} \left\{ \sum_j \left(\left\| \mathcal{O}(\mathcal{T}(\boldsymbol{I}_j; \boldsymbol{S}_j); \theta) - \boldsymbol{G}_j \right\|_2 \right) \right\}. \tag{4.6}$$

4.3.3 PrObeD

Our proposed approach comprises of three stages: template generation, template recovery, and detector fine-tuning. First, we use an encoder network to generate an image-dependent template for

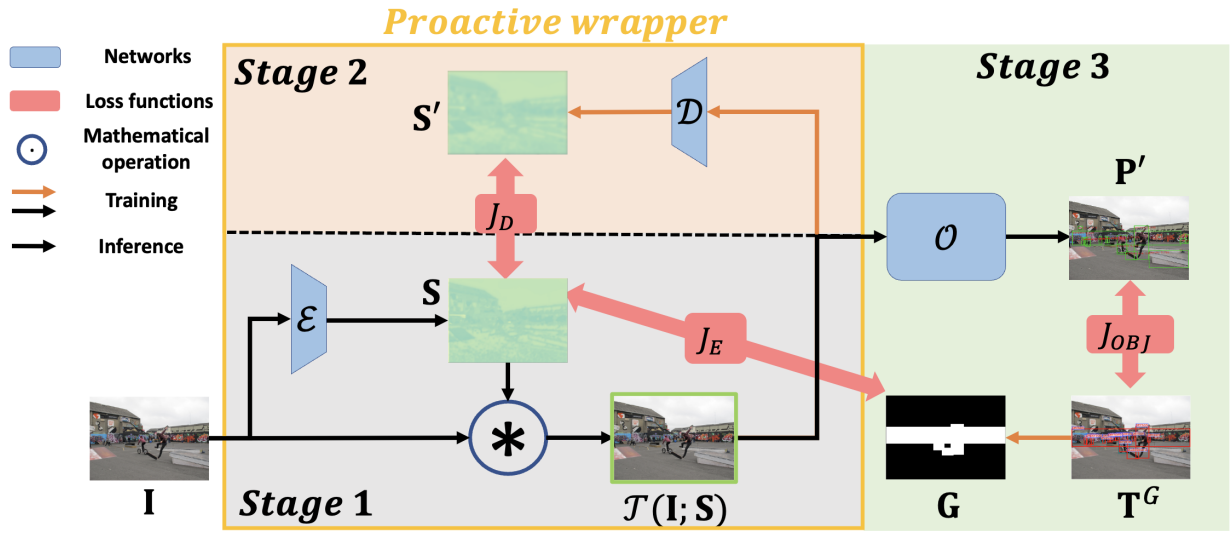


Figure 4.2 **Overview of PrObeD.** PrObeD consists of three stages: (1) template generation, (2) template recovery, and (3) detector fine-tuning. The templates are generated by encoder network $\mathcal{E}$ to encrypt the input images. The decoder network $\mathcal{D}$ is used to recover the template from the encrypted images. Finally, the encrypted images are used to fine-tune the object detector to perform detection. We train all the stages in an end-to-end manner. However, for inference, we only use stages 1 and 3. Best viewed in color.

image encryption. This encrypted image is further used to recover the template through a decoder network. Finally, the object detector is fine-tuned using the encrypted images. All three stages are trained in an end-to-end fashion. While all the stages are used for training PrObeD, we specifically use only stages 1 and 3 for inference. We will now describe each stage in detail.

4.3.3.1 Proactive Wrapper

Our proposed approach consists of three stages, as shown in Fig. 4.2. However, only the first two stages are part of our proposed proactive wrapper, which can be applied to object detector to improve its performance.

Stage 1: Template Generation. Prior works learn a set of templates [7, 6] in their proactive schemes. This set of templates is enough to perform the respective downstream tasks as the generative model manipulates the template, which is easy to capture with a set of learnable templates. However, for object detection tasks, every image has unique object characteristics such as size, appearance, and color that can vary significantly. This variability present in the images may exceed the descriptive capacity of a finite set of templates, thereby necessitating the use of image-

specific templates to accurately represent the range of object features present in each image. In other words, a fixed set of templates may not be sufficiently flexible to capture the diversity of visual features across the given set of input images, thus demanding more adaptable, image-dependent templates.

Motivated by the above argument, we propose to generate the template $\boldsymbol{S}_j$ for every image using an encoder network. We hypothesize that highlighting the area of the key foreground objects would be beneficial for object detection. Therefore, for GOD, we use the ground-truth bounding boxes T^G to generate the pseudo ground-truth segmentation map. Specifically, for any image $\boldsymbol{I}_j$, if the bounding box coordinates are $T_j^G = \{x_1, x_2, y_1, y_2\}$, we define the pseudo ground-truth segmentation map as:

$$\forall m \in [0, H], n \in [0, W], \text{ we have}$$

$$\boldsymbol{G}_j(m, n) = 1 \text{ if } x_1 \leq m \leq x_2 \text{ and } y_1 \leq n \leq y_2, \text{ otherwise } 0$$

However, for COD, the dataset already has the ground-truth segmentation map $\boldsymbol{G}_j$, which we use as the supervision for the encoder to output the templates with semantic information of the image to be restricted only in the region of interest for the detector. For both GOD and COD, we minimize the cosine similarity (Cos) between $\boldsymbol{S}_j$ and $\boldsymbol{G}_j$ as the supervision for the encoder network. The encoder loss J_E is as follows:

$$J_E = 1 - \text{Cos}(\boldsymbol{S}_j, \boldsymbol{G}_j) = 1 - \text{Cos}(\mathcal{E}(\boldsymbol{I}_j), \boldsymbol{G}_j). \tag{4.7}$$

This generated template acts as a mask for the input image to highlight the object region of interest for the detector. We use this template with the transformation $\mathcal{T}$ to encrypt the input image as $\mathcal{T}(\boldsymbol{I}_j; \boldsymbol{S}_j) = \boldsymbol{I}_j \odot \boldsymbol{S}_j$. As we start from the pretrained model of object detector O, we initialize the bias of the last layer of the encoder as 0 so that for the first few iterations, $\boldsymbol{S}_j \approx \mathbf{1}$. This is to ensure that the distribution of $\boldsymbol{I}_j$ and $\mathcal{T}(\boldsymbol{I}_j; \boldsymbol{S}_j)$ remains similar for the first few iterations, and O doesn't encounter a sudden change in its input distribution.

Stage 2: Template Recovery. So far, we have discussed the generation of template $\boldsymbol{S}_j$ using $\mathcal{E}$, which will be used as a mask to encrypt the input image. The encrypted images are used for two

purposes: (1) recovery of templates and (2) fine-tuning of the object detector. The main intuition of recovering the templates is from the prior works on image steganalysis [258, 257] and proactive schemes [7, 6]. Motivated by these works, we draw the following insight: *"To properly learn the optimal template and embed it onto the input images, it is beneficial to recover the template from encrypted images."*

To perform recovery, we exploit an encoder-decoder approach. Using this approach leverages the strengths of the encoder network $\mathcal{E}$ for feature extraction, capturing the most useful salient details, and the decoder network $\mathcal{D}$ for information recovery, allowing for efficient and effective encryption and decryption of the template. We also empirically show that not using the decoder to recover the templates harms the object detection performance.

To supervise $\mathcal{D}$ in recovering $\boldsymbol{S}_j$ from $\mathcal{T}(\boldsymbol{I}_j; \boldsymbol{S}_j)$, we propose to maximize the cosine similarity between the recovered template, $\boldsymbol{S}_j^{'}$ and $\boldsymbol{S}_j$. The decoder loss is as follows:

$$J_D = 1 - \text{Cos}(\boldsymbol{S}_j^{'}, \boldsymbol{S}_j) = 1 - \text{Cos}(\mathcal{D}(\mathcal{T}(\boldsymbol{I}_j; \boldsymbol{S}_j)), \boldsymbol{S}_j). \tag{4.8}$$

Stage 3: Detector Fine-tuning. Due to our encryption, the distribution of the images input to the pretrained O changes. Thus, we fine-tune O on the encrypted images $\mathcal{T}(\boldsymbol{I}_j; \boldsymbol{S})$. As proposed in Theorem 1, given the encrypted images $\mathcal{T}(\boldsymbol{I}_j; \boldsymbol{S})$, we use the pretrained detector O with parameters θ to arrive at a better local minima. Therefore, the general objective of GOD and COD in Eq. (4.5) and Eq. (4.6) change to as follows:

$$\min_{\theta, \theta_\mathcal{E}, \theta_\mathcal{D}} \left\{ \sum_j \left(||O(\mathcal{T}(\boldsymbol{I}_j; \mathcal{E}(\boldsymbol{I}_j; \theta_\mathcal{E})); \theta, \theta_\mathcal{D})_1 - T_j||_2 - \sum_{i=1}^{N} \left(C_j^i . \log(O(\mathcal{T}(\boldsymbol{I}_j; \mathcal{E}(\boldsymbol{I}_j; \theta_\mathcal{E})); \theta, \theta_\mathcal{D})_2) \right) \right) \right\}, \tag{4.9}$$

$$\min_{\theta, \theta_\mathcal{E}, \theta_\mathcal{D}} \left\{ \sum_j \left(\left\| O(\mathcal{T}(\boldsymbol{I}_j; \mathcal{E}(\boldsymbol{I}_j; \theta_\mathcal{E})); \theta, \theta_\mathcal{D}) - \boldsymbol{G}_j \right\|_2 \right) \right\}. \tag{4.10}$$

We use the detector-specific loss function J_{OBJ} of O along with the encoder and decoder loss in Eq. (4.7) and Eq. (4.8) to train all the three stages. The overall loss function J to train PrObeD is as follows:

$$J = \lambda_{OBJ} J_{OBJ} + \lambda_E J_E + \lambda_D J_D. \tag{4.11}$$

Table 4.2 **GOD results** on MS-COCO val split. PrObeD improves the performance of all GOD at all thresholds and across all categories.

Method	AP ↑	AP_{50} ↑	AP_{75} ↑	AP_S ↑	AP_M ↑	AP_L ↑
Faster R-CNN [260]	19.3	42.5	16.9	1.8	17.9	39.3
Faster R-CNN [260]+PrObeD	**31.7**	**52.6**	**33.3**	**11.0**	**35.5**	**51.1**
Faster R-CNN + FPN [188]	37.3	58.0	40.6	21.4	41.0	48.4
Faster R-CNN + FPN [188] + Seg. Mask [124]	38.2	60.3	41.7	22.1	43.2	**51.2**
Faster R-CNN + FPN [188] + PrObeD	**38.5**	**60.4**	**41.9**	**22.5**	**43.4**	49.8
Sparse R-CNN [291]	37.6	55.6	40.2	20.5	39.6	52.9
Sparse R-CNN [291]+ PrObeD	**39.2**	**57.5**	**41.5**	**21.7**	**40.1**	**53.6**
YOLOv5 [254]	48.9	67.6	53.1	31.8	54.4	62.3
YOLOv5 [254]+ PrObeD	**49.4**	**67.9**	**53.5**	**32.0**	**55.1**	**62.6**
DeTR [32]	41.9	62.3	44.1	20.3	45.8	61.0
DeTR [32]+ PrObeD	**42.1**	**62.6**	**44.4**	**20.4**	**46.0**	**61.3**

Table 4.3 **COD results** on CAMO, COD10K and NC4K datasets. PrObeD outperforms DGNet on all datasets and metrics.

Method	CAMO				COD10K				NC4K			
	E_m↑	S_m↑	wF_β↑	MAE↓	E_m↑	S_m↑	wF_β↑	MAE↓	E_m↑	S_m↑	wF_β↑	MAE↓
DGNet[149]	0.859	0.791	0.681	0.079	0.833	0.776	0.603	0.046	0.876	0.815	0.710	0.059
+ PrObeD	**0.871**	**0.797**	**0.702**	**0.071**	**0.869**	**0.803**	**0.661**	**0.037**	**0.900**	**0.838**	**0.755**	**0.049**

4.4 Experiments

We apply PrObeD for two categories of object detectors: GOD and COD.

GOD Baselines. For GOD, we apply PrObeD on four detectors with varied architectures: two-stage, one-stage, and transformer-based detectors, namely, Faster R-CNN [260], YOLO [254], Sparse R-CNN, and DeTR [32]. We use these works as baselines for three reasons: (1) varied architecture types, (2) their increased prevalence in the community, and (3) varied timelines (from earlier to recent detectors). We use the PyTorch [239] code of the respective detectors for our GOD experiments and use the corresponding GODs as our baseline. For YOLOv5 and DeTR, we use the official repositories released by the authors; for Faster R-CNN, we use the public repository "Faster R-CNN.pytorch". For other GOD detectors, we use Detectron2 library as the pre-trained detector. We use the ResNet101 backbone for Faster R-CNN, Sparse R-CNN and DeTR, and CSPDarknet53 for YOLOv5.

COD Baselines. For COD, we apply PrObeD on the current SoTA camouflage detector DGNet [149] and use DGNet as our baseline. For all object detectors, we use the pretrained model released by

the authors and fine-tune them with PrObeD. Please see the supplementary for more details.

Datasets. Our experiments use the MS-COCO 2017 [190] dataset for GOD, while we use CAMO [176], COD10K [81], and NC4K [204] datasets for COD. We use the following splits of these datasets:

- MS-COCO 2017 Val Split [190]: It includes 118,287 images for training and $5K$ for testing.
- COD10K Val Split [81]: It includes 4,046 camouflaged images for training and 2,026 for testing.
- CAMO Val Split [176]: It includes $1K$ camouflaged images for training and 250 for testing.
- NC4K Val [204]: It includes 4,121 NC4K images. We use it for generalization testing as in [149].

Evaluation Metrics. We use mean average precision average at multiple thresholds in $[0.5, 0.95]$ (AP) for GOD as in [190]. We also report results at threshold of 0.5 (AP_{50}), threshold of 0.75 (AP_{75}) and at different object sizes: small (AP_S), medium (AP_M), and large (AP_L). For COD, we use E-measure E_m, S-measure S_m, weighted F1 score wF_β and mean absolute error MAE as [149].

4.4.1 GOD Results

Quantitative Results. Tab. 4.2 shows the results of applying PrObeD on GOD networks. PrObeD improves the average precision of all three detectors. The performance gain is significant for Faster R-CNN. As Faster R-CNN is an older detector, it was at a worse minima to start with. PrObeD improves the convergence weight of Faster R-CNN by a significant margin, thereby improving the performance. We further experiment with two variations of Faster R-CNN, namely, Faster R-CNN + FPN and Sparse-RCNN. We observe an increase in the performance of both detectors. PrObeD also improves newer detectors like YOLOv5 and DeTR, although the gains are smaller compared to Faster R-CNN. We believe this happens because the newer detectors leave little room for improvement due to which PrObeD improves the performance slightly. We next compare PrObeD with a work that leverage segmentation map as a mask for object detection. We compare

Table 4.4 Performance comparison with proactive works. MaLP [7] has a significantly deteriorated performance than PrObeD.

Method	CAMO				COD10K				NC4K			
	$E_m\uparrow$	$S_m\uparrow$	$wF_\beta\uparrow$	MAE$\downarrow$	$E_m\uparrow$	$S_m\uparrow$	$wF_\beta\uparrow$	MAE$\downarrow$	$E_m\uparrow$	$S_m\uparrow$	$wF_\beta\uparrow$	MAE$\downarrow$
MaLP [7]	0.474	0.514	0.218	0.254	0.491	0.520	0.150	0.202	0.503	0.548	0.228	0.222
PrObeD	**0.871**	**0.797**	**0.702**	**0.071**	**0.869**	**0.803**	**0.661**	**0.037**	**0.900**	**0.838**	**0.755**	**0.049**

Table 4.5 **Ablation studies** of PrObeD using Faster R-CNN GOD on MS-COCO 2017 dataset. Removing the encoder/decoder network or adding the template results in degraded performance.

Changed	From→To	AP$\uparrow$	$AP_{50}\uparrow$	$AP_{75}\uparrow$	$AP_S\uparrow$	$AP_M\uparrow$	$AP_L\uparrow$
Template	Image Dependent→Fixed	17.6	37.9	15.1	1.3	15.4	39.5
	Image Dependent→Universal	19.4	42.6	17.1	1.9	18.0	39.4
Decoder	Yes→No	25.2	46.1	26.2	5.3	26.6	24.1
Transformation	Multiply→Add	19.2	42.3	20.1	1.7	17.9	39.1
PrObeD	-	**31.7**	**52.6**	**33.3**	**11.0**	**35.5**	**51.1**

our performance with Mask R-CNN [124], which uses an image segmentation branch to help with object detection. Tab. 4.2 shows that the gains using Mask R-CNN are lower than using our proactive wrapper.

Qualitative Results. Fig. 4.3 shows qualitative results for the MS-COCO 2017 dataset. PrObeD clearly improves the performance of pretrained Faster R-CNN for three types of errors: Missed predictions, false negatives, and localization errors. PrObeD has a lower number of missed predictions, fewer false positives, and better bounding box localization. We also visualize the generated and recovered templates. We see that the template has object semantics of the input images. When the template is multiplied with the input image, it highlights the foreground objects, thereby making the task of object detector easier.

Error Analysis. We show the error analysis [23] for GOD section 4 of the supplementary. We observe that all GOD detectors make mistakes mainly due to five types of errors: classification, localization, duplicate detection, background detection, and missed detection. The main reason for the degraded performance is the errors in which the foreground-background boundary is missed. These errors include localization, background detection, and missed detection. Our proactive wrapper significantly corrects these errors, as the template has object semantics, which, when multiplied with the input image, highlights the foreground objects, consequently simplifying the task of object detection.

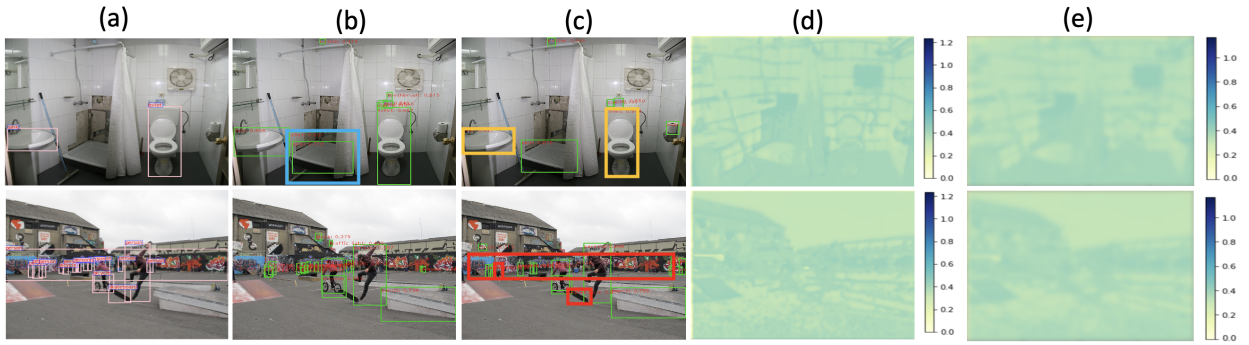


Figure 4.3 **Qualitative GOD Results** on MS-COCO 2017 dataset. (a) ground-truth annotations, (b) Faster R-CNN [260] predictions, (c) Faster R-CNN [260]+ PrObeD predictions, (d) generated template, and (e) recovered template. We highlight the objects responsible for improvement in (c) as compared to (b). The yellow box represents better localization, the blue box represents false positives, and the red box represents missed predictions. PrObeD improves on all these errors made by (b).

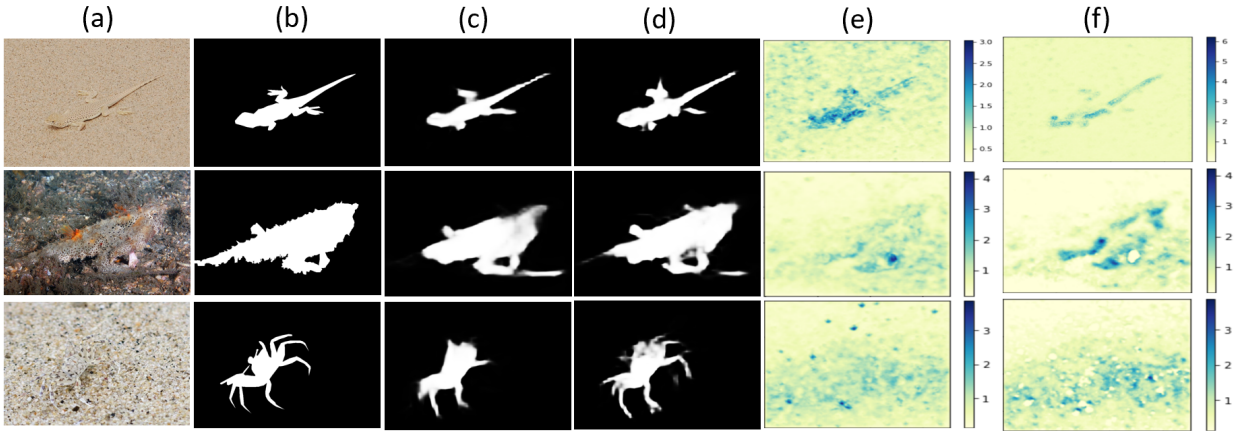


Figure 4.4 **Qualitative COD Results** on CAMO, COD10K, and NC4K datasets from top to bottom, after applying PrObeD. (a) input images, (b) ground-truth camouflaged map, (c) DGNet[149] predictions, (d) DGNet[149]+ PrObeD predictions, (e) generated PrObeD template, and (f) recovered PrObeD template. PrObeD template has the semantics of the camouflaged object, which aids DGNet in detection.

4.4.2 COD Results

Quantitative Results. Tab. 4.3 shows the result of applying PrObeD to DGNet [149] on three different datasets. PrObeD, when applied on top of DGNet, outperforms DGNet on all four metrics for all datasets. The biggest gain appears in COD10K and NC4K datasets. This is impressive as these datasets have more diverse testing images than CAMO. As NC4K is only a testing set, the higher performance of PrObeD demonstrates its superior generalizability as compared to DGNet [149]. This result agrees with the observation in [6, 7], where proactive-based approaches

Table 4.6 **Ablation of training iterations** on Faster R-CNN. YOLOv5, and DeTR for more iterations similar to after applying PrObeD. We also report the inference time for all the detectors before and after applying PrObeD. Training object detectors proactively with PrObeD results in more performance gain compared to training passively for more iterations. PrObeD adds an overhead cost on top of the inference cost of detectors.

Method	Iterations	AP ↑	AP_{50} ↑	AP_{75} ↑	AP_S ↑	AP_M ↑	AP_L ↑	Time (ms)
Faster R-CNN [260]	1×	19.3	42.5	16.9	1.8	17.9	39.3	161.1
Faster R-CNN [260]	2×	20.1	46.6	21.5	3.3	20.3	41.2	
Faster R-CNN [260] + PrObeD	2×	**31.7**	**52.6**	**33.3**	**11.0**	**35.5**	**51.1**	175.3 (↑ 8.7%)
YOLOv5 [254]	1×	48.9	67.6	53.1	31.8	54.4	62.3	48.5
YOLOv5 [254]	2×	48.8	67.7	53.0	31.8	54.7	62.4	
YOLOv5 [254] + PrObeD	2×	**49.4**	**67.9**	**53.5**	**32.0**	**55.1**	**62.6**	62.7 (↑ 29.1%)
DeTR [32]	1×	41.9	62.3	44.1	20.3	45.8	61.0	194.2
DeTR [32]	2×	41.9	62.4	44.0	20.1	45.9	61.1	
DeTR [32] + PrObeD	2×	**42.1**	**62.6**	**44.4**	**20.4**	**46.0**	**61.3**	208.4 (↑ 7.2%)

exhibit improved generalization on manipulation detection and localization tasks.

Qualitative Results. Fig. 4.4 visualizes the predicted camouflaged map for DGNet before and after applying PrObeD on testing samples of all three datasets. PrObeD improves the predicted camouflaged map, with less blurriness along the boundaries and better localization of the camouflaged object. As observed before for GOD, the generated and recovered template has the semantics of the camouflaged objects, which after multiplication intensifies the foreground object, resulting in better segmentation by DGNet.

4.4.3 Ablation Study

Comparison with Proactive Works. The prior proactive works perform a different task of image manipulation detection and localization. Therefore, these works are not directly comparable to our proposed proactive wrapper, which performs a different task of object detection as described in Tab. 4.1. However, manipulation localization and COD both involve a prediction of a localization map, segmentation, and fakeness map, respectively. This inspires us to experiment with MaLP [7] for the task of COD. We train the localization module of MaLP supervised with the COD datasets. The results are shown in Tab. 4.4. We see that MaLP is not able to perform well for all three datasets. MaLP is designed for estimating universal templates rather than templates tailored to specific images. It shows the significance of image-specific templates in object detection. While MaLP's design with image-independent templates is effective for localizing image manipulation,

applying it to object detection has a negative impact on performance.

Framework Design. PrObeD consists of blocks to improve the object detector. Tab. 4.5 ablates different versions of PrObeD to highlight the importance of each block in our design. PrObeD utilizes an encoder network $\mathcal{E}$ to learn image-dependent templates aiding the detector. We remove the encoder $\mathcal{E}$ from our network, replacing it with a fixed template. We observe that the performance deteriorates by a large margin. Next, we make this template learnable as proposed in PrObeD, but only a single template would be used for all the input images. This choice also results in worse performance, highlighting that image-dependent templates are necessary for object detection. Finally, we remove the decoder network $\mathcal{D}$, which is used to recover the template from the encrypted images. Although this results in a better performance than the pretrained Faster R-CNN, we observe a drop as compared to PrObeD. Therefore, as discussed in Sec. 4.3.3, the recovery of templates is indeed a necessary and beneficial step for boosting the performance of the proactive schemes.

Encryption Process. PrObeD includes an encryption process as described in Eq. (4.3), which involves multiplying the template with the input image. This process makes the template act as a mask, highlighting the foreground for better detection. However, prior proactive works [7, 6] consider adding templates to achieve better results. Thus, we ablate by changing the encryption process to template addition. Tab. 4.5 shows that template addition degrades performance by a significant margin w.r.t. our multiplication scheme. This shows that encryption is a key step in formulating proactive schemes, and the same encryption process may not work for all tasks.

More Training Time. We perform an ablation to show that the performance gain of the detector is due to our proactive wrapper instead of training for more iterations of the pretrained object detector. Results in Tab. 4.6 show that although more training iterations for the detector has a performance gain, it's not enough to get the significant margin in performance as achieved by PrObeD. This shows that extra training can help, but only up to a certain extent.

Inference Time. We evaluate the overhead computational cost after applying PrObeD on different object detectors are shown in Tab. 4.6, averaged across $1,000$ images, on a NVIDIA $V100$ GPU.

Our encoder network has 17 layers, which adds extra cost for inference. For detectors with bulky architectures like Faster R-CNN (ResNet101) and DeTR (transformer), the overhead computational cost is quite small, 8.7% and 7.2%, respectively. This additional cost is minor compared to the performance gain of detectors, especially Faster R-CNN. For a lighter detector like YOLOv5, our overhead computational cost increases to 29.1%. So, there is a trade-off of applying PrObeD to different detectors with varied architectures. PrObeD is more beneficial to bulky detectors like two-staged/transformer-based as compared to one-stage detectors.

4.5 Conclusion

We mathematically prove that the proactive method results in a better-converged model than the passive detector under assumptions and, consequently, a better 2D object detector. Based on this finding, we propose a proactive scheme wrapper, PrObeD, which enhances the performance of camouflaged and generic object detectors. The wrapper outputs an image-dependent template using an encoder network, which encrypts the input images. These encrypted images are then used to fine-tune the object detector. Extensive experiments on MS-COCO, CAMO, COD10K, and NC4K datasets show that PrObeD improves the overall object detection performance for both GOD and COD detectors.

Limitations. Our proposed scheme has the following limitations. First, PrObeD does not provide a significant gain for recent object detectors such as YOLO and DeTR. Second, the proactive wrapper should be thoroughly tested on other object detectors to show the generalizability of PrObeD. Finally, we only experiment with simple multiplication and addition as the encryption scheme. A more sophisticated encryption process might further improve the object detectors' performance. We leave these for our future avenues.

CHAPTER 5

PROMARK: PROACTIVE DIFFUSION WATERMARKING FOR CAUSAL ATTRIBUTION

Generative AI (GenAI) is transforming creative workflows through the capability to synthesize and manipulate images via high-level prompts. Yet creatives are not well supported to receive recognition or reward for the use of their content in GenAI training. To this end, we propose ProMark, a causal attribution technique to attribute a synthetically generated image to its training data concepts like objects, motifs, templates, artists, or styles. The concept information is proactively embedded into the input training images using imperceptible watermarks, and the diffusion models (unconditional or conditional) are trained to retain the corresponding watermarks in generated images. We show that we can embed as many as 2^{16} unique watermarks into the training data, and each training image can contain more than one watermark. ProMark can maintain image quality whilst outperforming correlation-based attribution. Finally, several qualitative examples are presented, providing the confidence that the presence of the watermark conveys a causative relationship between training data and synthetic images[1].

5.1 Introduction

GenAI is able to create high-fidelity synthetic images spanning diverse concepts, largely due to advances in diffusion models, *e.g*. DDPM [132], DDIM [216], LDM [264]. GenAI models, particularly diffusion models, have been shown to closely adopt and sometimes directly memorize the style and the content of different training images – defined as "concepts" in the training data [33, 172]. This leads to concerns from creatives whose work has been used to train GenAI. Concerns focus upon the lack of a means for attribution, *e.g*.recognition or citation, of synthetic images to the training data used to create them and extend even to calls for a compensation mechanism (financial, reputational, or otherwise) for GenAI's derivative use of concepts in training images contributed by creatives.

[1] *Vishal Asnani*, John Collomosse, Tu Bui, Xiaoming Liu, and Shruti Agarwal. "ProMark: Proactive Diffusion Watermarking for Causal Attribution." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024

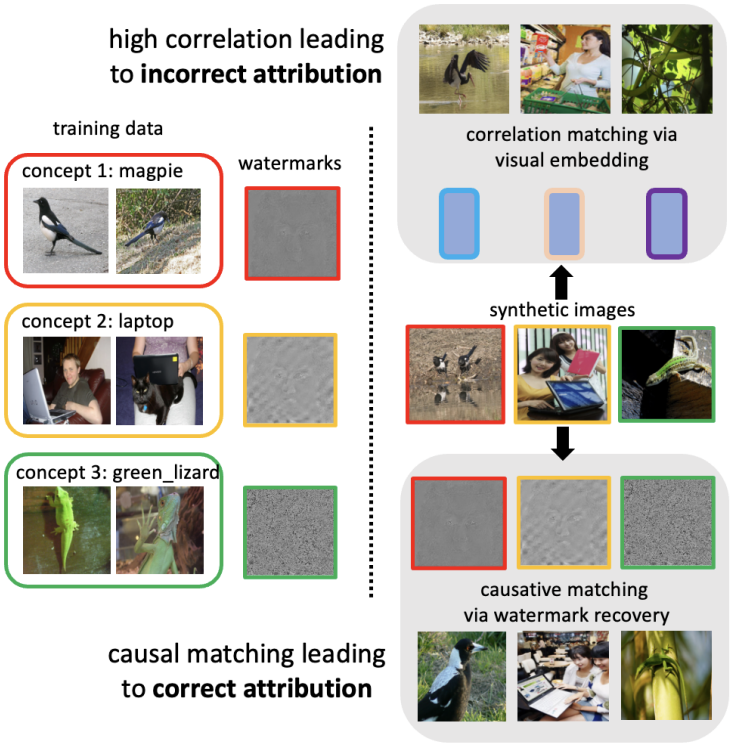


Figure 5.1 **Causative *vs*.correlation-based matching for concept attribution.** ProMark identifies the training data most responsible for a synthetic image ('attribution'). Correlation-based matching doesn't always perform the data attribution properly. We propose ProMark, which is a proactive approach involving adding watermarks to training data and recovering them from the synthetic image to perform attribution in a causative way.

We refer to this problem as *concept attribution* – the ability to attribute generated images to the training concept/s which have most directly influenced their creation. Several passive techniques have recently been proposed to solve the attribution problem [11, 269, 328]. These approaches use visual correlation between the generated image and the training images for attribution. Whilst they vary in their method and rationale for learning the similarity embedding – all use some forms of contrastive training to learn a metric space for visual correlation.

We argue that although correlation can provide visually intuitive results, a measure of similarity is not a causative answer to whether certain training data is responsible for the generation of an image or not. Further, correlation-based techniques can identify close matches with images that were not even present in the training data.

Keeping this in mind, we explore an intriguing field of research which is developing around proactive watermarking methodologies [356, 267, 325, 7], that employ signals, termed *templates*

to encrypt input images before feeding them into the network. These works have integrated and subsequently retrieved templates to bolster the performance of the problem at hand. Inspired by these works, we introduce ProMark, a proactive watermarking-based approach for GenAI models to perform concept attribution in a causative way. The technical contributions of ProMark are three-fold:

1. Causal vs. Correlation-based Attribution. ProMark performs causal attribution of synthetic images to the predefined concepts in the training images that influenced the generation. Unlike prior works that visually correlate synthetic images with training data, we make no assumption that visual similarity approximates causation. ProMark ties watermarks to training images and scans for the watermarks in the generated images, enabling us to demonstrate rather than approximate/imply causation. This provides confidence in grounding downstream decisions such as legal attribution or payments to creators.

2. Multiple Orthogonal Attributions. We propose to use orthogonal invisible watermarks to proactively embed attribution information into the input training data and add a BCE loss during the training of diffusion models to retain the corresponding watermarks in the generated images. We show that ProMark causatively attributes as many as 2^{16} unique training-data concepts like objects, scenes, templates, motifs, and style, where the generated images can simultaneously express one or two orthogonal concepts.

3. Flexible Attributions. ProMark can be used for training conditional or unconditional diffusion models and even finetuning a pre-trained model for only a few iterations. We show that ProMark's causative approach achieves higher accuracy than correlation-based attribution over five diverse datasets (Sec. 5.4.1): Adobe Stock, ImageNet, LSUN, Wikiart, and BAM while preserving synthetic image quality due to the imperceptibility of the watermarks.

Fig. 5.1 presents our scenario, where synthetic image(s) are attributed back to the most influential GenAI training images. Correlation-based techniques [11, 328] try to match the high-level image structure or style. Here, the green-lizard synthetic image is matched to a generic green image without a lizard [11]. With ProMark's causative approach, the presence of the green-lizard watermark in

Table 5.1 **Comparison of ProMark with prior works**. Uniquely, we perform causative attribution using proactive watermarking to attribute multiple concepts. [Keys: emb.: embedding, obj.: object, own.: ownership, sem.: semantic, sty.: style, wat.: watermark].

Method	Scheme type	Task	Match type	# Class	Multiple attribution	Attribution type
[269]	passive	attribution	emb.	-	×	sty.
[11]	passive	attribution	emb.	-	×	obj.
[328]	passive	attribution	emb.	693	×	sty., obj.
[90]	passive	detect	wat.	2	×	-
[198]	passive	detect	wat.	2	×	-
[62]	passive	detect	wat.	2	×	-
[325]	proactive	detect	wat.	2	-	-
[6]	proactive	detect	wat.	2	-	-
[7]	proactive	localization	wat.	2	-	-
[5]	proactive	obj. detect	-	90	-	-
ProMark	proactive	attribution	wat.	2^{16}	✓	sty., obj. own., sem.

the synthetic image will correctly indicate the influence of the similarly watermarked concept group of lizard training images.

5.2 Related Works

Passive Concept Attribution. Concept attribution differs from model [28] or camera [37] attribution in that the task is to determine the responsible training data for a given generation. Existing concept attribution techniques are passive – they do not actively modify the GenAI model or training data but instead, measure the visual similarity (under some definition) of synthetic images and training data to quantify attribution for each training image. EKILA [11] proposes patch-based perceptual hashing (visual fingerprinting [224, 19]) to match the style of the query patches to the training data for attribution. Wang *et al*. [328] finetune semantic embeddings like CLIP, DINO, *etc*.for the attribution task. Both [11] and [328] explore ALADIN [269] for style attribution. ALADIN is a feature representation for fine-grained style similarity learned using a weakly supervised approach.

All these works are regarded as passive approaches as they take the image as an attribute by correlating between generated and training image styles. Instead, our approach is a proactive scheme that adds a watermark to training images and performs attribution in a causal manner (Tab. 5.1).

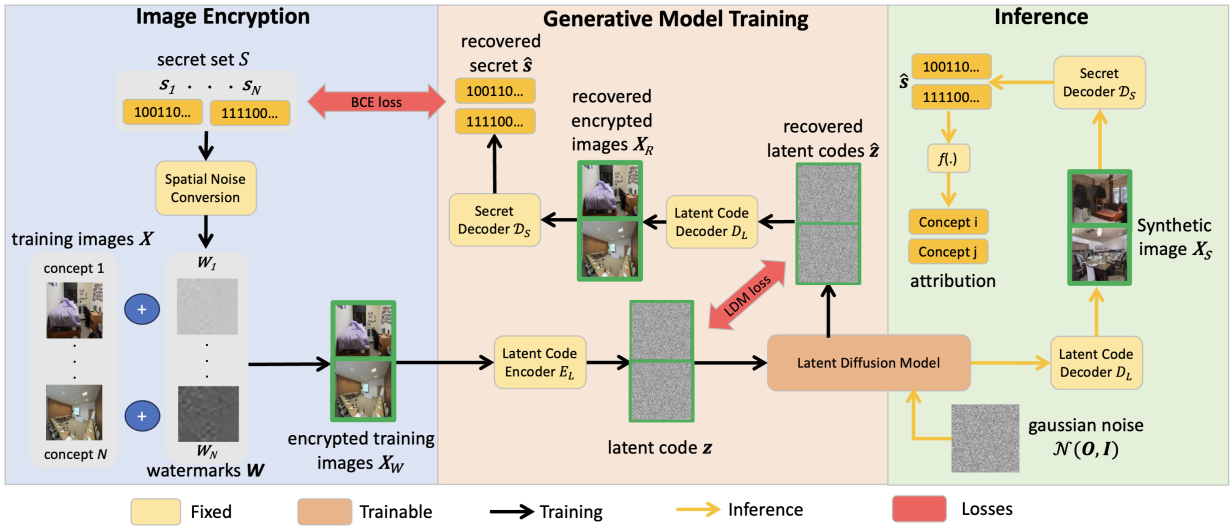


Figure 5.2 **Overview of ProMark**. We show the training and inference procedure for our proposed method. Our training pipeline involves two stages, image encryption and generative model training. We convert the bit-sequences to spatial watermarks ($\boldsymbol{W}$), which are then added to the corresponding concept images ($\boldsymbol{X}$) to make them encrypted ($\boldsymbol{X_W}$). The generative model is then trained with the encrypted images using the LDM supervision. During training, we recover the added watermark using the secret decoder ($\mathcal{D}_S$) and apply the BCE supervision to perform attribution. To sample newly generated images, we use a Gaussian noise and recover the bit-sequences using the secret decoder to attribute them to different concepts. Best viewed in color.

Proactive Schemes. Proactive schemes involve adding a signal/perturbation onto the input images to benefit different tasks like deepfake tagging [325], deepfake detection [6], manipulation localization [7], object detection [5], *etc*.. Some works [356, 267] disrupt the output of the generative models by adding perturbations to the training data. Alexandre *et al*. [270] tackles the problem of training dataset attribution by using fixed signals for every data type. These prior works successfully demonstrate the use of watermarks to classify the content of the AI-generated images proactively. We extend the idea of proactive watermarking to perform the task of causal attribution of AI-generated images to influential training data concepts. Watermarking has not been used to trace attribution in GenAI before.

Watermarking of GenAI Models. It is an active research to watermark AI-generated images for the purpose of privacy protection. Fernandez1 *et al*. [90] fine-tune the LDM's decoder to condition on a bit sequence, embedding it in images for AI-generated image detection. Kirchenbauer *et al*. [165] propose a watermarking method for language models by pre-selecting random tokens

and subtly influencing their use during word generation. Zhao *et al*. [380] use a watermarking scheme for text-to-image diffusion models, while Liu *et al*. [198] verify watermarks by pre-defined prompts. [62, 241] add a watermark for detecting copyright infringement. Asnani *et al*. [8] reverse engineer a fingerprint left by various GenAI models to further use it for recovering the network and training parameters of these models [354, 8]. Finally, Cao *et al*. [31] adds an invisible watermark for protecting diffusion models which are used to generate audio modality. Most of these works have used watermarking for protecting diffusion models, which enables them to add just one watermark onto the data. In contrast, we propose to add multiple watermarks to the training data and to a single image, which is a more challenging task than embedding a universal watermark.

5.3 Method

5.3.1 Background

Diffusion Models. Diffusion models learn a data distribution $p(\boldsymbol{X})$, where $\boldsymbol{X} \in \mathbb{R}^{h \times w \times 3}$ is the input image. They do this by iteratively reducing the noise in a variable that initially follows a normal distribution. This can be viewed as learning the reverse steps of a fixed Markov Chain with a length of T. Recently, LDM [264] is proposed to convert images to their latent representation for faster training in a lower dimensional space than the pixel space. The image is converted to and from the latent space by a pretrained autoencoder consisting of an encoder $\boldsymbol{z} = \mathcal{E}_L(\boldsymbol{X})$ and a decoder $\boldsymbol{X}_R = \mathcal{D}_L(\boldsymbol{z})$, where $\boldsymbol{z}$ is the latent code and $\boldsymbol{X}_R$ is the reconstructed image. The trainable denoising module of the LDM is $\epsilon_\theta(\boldsymbol{z}_t, t); t = 1...T$, where ϵ_θ is trained to predict the denoised latent code $\hat{\boldsymbol{z}}$ from its noised version $\boldsymbol{z}_t$. This objective function can be defined as follows:

$$L_{LDM} = \mathbb{E}_{\mathcal{E}_L(\boldsymbol{X}),\epsilon\sim\mathcal{N}(0,1),t}\left[||\epsilon - \epsilon_\theta(\boldsymbol{z}_t, t)||_2^2\right], \tag{5.1}$$

where ϵ is the noise added at step t.

Image Encryption. Proactive works [6, 7, 5] have shown performance gain on various tasks by proactively transforming the input training images $\boldsymbol{X}$ with a watermark, resulting in an encrypted image. This watermark is either fixed or learned, depending on the task at hand. Similar to prior

proactive works, our image encryption is of the form:

$$\boldsymbol{X}_W = \mathcal{T}(\boldsymbol{X}; \boldsymbol{W}) = \boldsymbol{X} + m \times R(\boldsymbol{W}, h, w), \tag{5.2}$$

where $\mathcal{T}$ is the transformation, $\boldsymbol{W}$ is the spatial watermark, $\boldsymbol{X}_W$ is the encrypted image, m is the watermark strength, and $R(.)$ resizes $\boldsymbol{W}$ to the input resolution (h, w).

We use the state-of-the-art watermarking technique RoSteALS [27] to compute the spatial watermarks for encryption due to its robustness to image transformation and generalization (the watermark is independent of content of the input image). RoSteALS is designed to embed a secret of length b-bits into an image using robust and imperceptible watermarking. It comprises of a secret encoder $\mathcal{E}_S(\boldsymbol{s})$, which converts the bit-secret $\boldsymbol{s} \in \{0, 1\}^b$ into a latent code offset $\boldsymbol{z}_o$. It is then added to the latent code of an autoencoder $\boldsymbol{z}_w = \boldsymbol{z} + \boldsymbol{z}_o$. This modified latent code $\boldsymbol{z}_w$ is then used to reconstruct a watermarked image via autoencoder decoder. Finally, a secret decoder, denoted by $\mathcal{D}_S(X_W)$, takes the watermarked images as input and predicts the bit-sequence $\hat{\boldsymbol{s}}$.

5.3.2 Problem Definition

Let $C = \{c_1, c_2, \ldots, c_N\}$ be a set of N distinct concepts within a dataset that is used for training a GenAI model for image synthesis. The problem of concept attribution can be formulated as follows:

Given a synthetic image $\boldsymbol{X}_S$ generated by a GenAI model, the objective of concept attribution is to accurately associate $\boldsymbol{X}_S$ to a concept $c_i \in C$ that significantly influenced the generation of $\boldsymbol{X}_S$.

We aim to find a mapping $f : \boldsymbol{X}_S \rightarrow c_i$ such that

$$c_i^* = \arg\max_{c_i \in C} f(\boldsymbol{X}_S, c_i), \tag{5.3}$$

where c_i^* represents the concept most strongly attributed to image $\boldsymbol{X}_S$.

5.3.3 Overview

The pipeline of ProMark is shown in Fig. 5.2. The principle is simple: if a specific watermark unique to a training concept can be detected from a generated image, it indicates that the generative model relies on that concept in the generation process. Thus, ProMark involves two steps: training data encryption via watermarks and generative model training with watermarked images.

To watermark the training data, the dataset is first divided into N groups, where each group corresponds to a unique concept that needs attribution. These concepts can be semantic (*e.g.*objects, scenes, motifs or stock image templates) or abstract (*e.g.*stylistic, ownership info). Each training image in a group is encoded with a unique watermark without significantly altering the image's perceptibility. Once the training images are watermarked, they are used to train the generative model. As the model trains, it learns to generate images based on the encrypted training images. Ideally, the generated images would have traces of watermarks corresponding to concepts they're derived from.

During inference, ProMark conforms to whether a generated image is derived from a particular training concept by identifying the unique watermark of that concept within the image. Through the careful use of unique watermarks, we can trace back and causally attribute generated images to their origin in the training dataset.

5.3.4 Training

During training, our algorithm is composed of two stages: image encryption and generative model training. We now describe each of these stages in detail.

Image Encryption. The training data is first divided into N concepts, and images in each partition are encrypted using a fixed spatial watermark $\boldsymbol{W}_j \in \mathbb{R}^{h \times w}$ ($j \in 0, 1, 2, ..., N$). Each noise $\boldsymbol{W}_j$ is a b-dim bit-sequence (secret) $\boldsymbol{s}_j = \{p_{j1}, p_{j2}, ..., p_{jb}\}$ where $p_{ji} \in \{0, 1\}$.

In order to compute the watermark $\boldsymbol{W}_j$ from the bit-sequence $\boldsymbol{s}_j$, we encrypt 100 random images with $\boldsymbol{s}_j$ using pretrained RoSteALS secret encoder $\mathcal{E}_S(.)$ which takes $b = 160$ length secret as input. From these encrypted images, we obtain 100 noise residuals by subtracting the encrypted images from the originals, which are averaged to compute the watermark $\boldsymbol{W}_j$ as:

$$\boldsymbol{W}_j = \frac{1}{100} \sum_{i=1}^{100} (\boldsymbol{X}_i - \mathcal{E}_S(\boldsymbol{X}_i, \boldsymbol{s}_j)). \tag{5.4}$$

The above process is defined as spatial noise conversion in Fig. 5.2. The averaging of noise residuals across different images reduces the image content in the watermark and makes the watermark independent of any specific image. Additionally, the generated watermarks are orthogonal due to

different bits for all $\boldsymbol{s}_j$, ensuring distinguishability from each other. With the generated watermarks, each training image is encrypted using Eq. (5.2) with one of the N watermarks that correspond to the concept the image belongs to.

Generative Model Training. Using the encrypted data, we train the LDM's denoising module $\epsilon_\theta(.)$ using the objective function (Eq. (5.1)), where $\boldsymbol{z}_t$ is the noised version of:

$$\boldsymbol{z} = \mathcal{E}_L(\boldsymbol{X}_{W_j}) = \mathcal{E}_L(\mathcal{T}(\boldsymbol{X}; \boldsymbol{W}_j)), \tag{5.5}$$

i.e., the input latent codes $\boldsymbol{z}$ are generated using the encrypted images $\boldsymbol{X}_{W_j}$ for $j \in \{0, 1, 2...., N\}$.

However, we found that only using LDM loss is insufficient to successfully learn the connection between the conceptual content and its associated watermark. This gap in learning presents a significant hurdle, as the primary aim is to trace back generated images to their respective training concepts via the watermark. To tackle this, an auxiliary supervision is introduced to LDM's training,

$$L_{BCE}(\boldsymbol{s}_j, \hat{\boldsymbol{s}}) = -\frac{1}{b}\sum_{i=1}^{b}[p_{ji}\log(\hat{p}_i) + (1 - p_{ji})\log(1 - \hat{p}_i)], \tag{5.6}$$

where $L_{BCE}(.)$ is the binary cross-entropy (BCE) between the actual bit-sequence $\boldsymbol{s}_j$ associated with watermark $\boldsymbol{W}_j$ and the predicted bit-sequence $\hat{\boldsymbol{s}}$. The denoised latent code $\hat{\boldsymbol{z}}$ is then decoded using the autoencoder $\mathcal{D}_L(.)$, and the embedded secret $\hat{\boldsymbol{s}}$ is predicted by the secret decoder $\mathcal{D}_S(.)$ as:

$$\hat{\boldsymbol{s}} = \mathcal{D}_S(\mathcal{D}_L(\hat{\boldsymbol{z}})). \tag{5.7}$$

By employing BCE, the model is guided to minimize the difference between the predicted watermark and the embedded watermark, hence improving the model's ability to recognize and associate watermarks with their respective concepts. Finally, our objective is to minimize the loss function $L_{attr} = L_{LDM} + \alpha L_{BCE}$ during training, where α is set to 2 for our experiments.

5.3.5 Inference

After the LDM learns to associate the watermarks with concepts, we use random Gaussian noise to sample the newly generated images from the model. While the diffusion model creates these new images, it also embeds a watermark within them. Each watermark maps to a distinctive

orthogonal bit-sequence associated with a specific training concept, serving as a covert signature for attribution.

To attribute the generated images and ascertain which training concept influenced them, we predict the secret embedded by the LDM in the generated images (see Eq. (5.7)). Given a predicted binary bit-sequence, $\hat{\boldsymbol{s}} = \{\hat{p_1}, \hat{p_2}, ..., \hat{p_b}\}$ and all the input bit-sequences $\boldsymbol{s}_j$ for $j \in 0, 1, 2..., N$, we define the attribution function, f, in Eq. (5.3) as:

$$f(\hat{\boldsymbol{s}}, \boldsymbol{s}_j) = \sum_{k=1}^{b} [\hat{p_k} = p_{jk}], \tag{5.8}$$

where $[\hat{p_k} = p_{ik}]$ acts as an indicator function, returning 1 if the condition is true, *i.e.*, the bits are identical, and 0 otherwise. Consequently, we assign the predicted bit sequence to the concept whose bit sequence it most closely mirrors — that is, the concept j^* for which $f(\hat{\boldsymbol{s}}, \boldsymbol{s}_{j^*})$ is maximized:

$$j^* = \arg \max_{j \in \{1,2,...,N\}} f(\hat{\boldsymbol{s}}, \boldsymbol{s}_j). \tag{5.9}$$

In other words, the concept whose watermark is most closely aligned with the generated image's watermark is deemed to be the influencing source behind the generated image.

5.3.6 Multiple Watermarks

In prior image attribution works, an image is usually attributed to a single concept (*e.g.*image content or image style). However, in real-world scenarios, an image may encapsulate multiple concepts. This observation brings forth a pertinent question: "Is it possible to use multiple watermarks for multi-concept attribution within a single image?"

In this paper, we propose a novel approach to perform multi-concept attribution by embedding multiple watermarks into a single image. In our preliminary experiments, we restrict our focus to the addition of two watermarks. To achieve this, we divide the image into two halves and resize each watermark to fit the respective halves. This ensures that each half of the image carries distinct watermark information pertaining to a specific concept.

For the input RGB image $\boldsymbol{X}$, $\{\boldsymbol{W}_i, \boldsymbol{W}_j\}$ are the watermarks for two secrets $\{\boldsymbol{s}_i, \boldsymbol{s}_j\}$, we formulate

the new transformation $\mathcal{T}$ as:

$$\begin{aligned}\mathcal{T}(\boldsymbol{X}; \boldsymbol{W}_i, \boldsymbol{W}_j) &= \Big\{ \boldsymbol{X}_{left}, \boldsymbol{X}_{right} \Big\} \\ &= \Big\{ (\boldsymbol{X}(:, 0 : \frac{w}{2}, :) + R(\boldsymbol{W}_i, h, \frac{w}{2})), \\ &\quad (\boldsymbol{X}(:, \frac{w}{2} : w, :) + R(\boldsymbol{W}_j, h, \frac{w}{2}) \Big\},\end{aligned}$$

where $\{.\}$ is the horizontal concatenation. The loss function uses the two predicted secrets ($\hat{\boldsymbol{s}}_1$ and $\hat{\boldsymbol{s}}_2$) from the two halves of the generated image, defined as:

$$L_{attr} = L_{LDM} + \alpha(L_{BCE}(\boldsymbol{s}_i, \hat{\boldsymbol{s}}_1) + L_{BCE}(\boldsymbol{s}_j, \hat{\boldsymbol{s}}_2)).$$

5.4 Experiments

5.4.1 Unconditional Diffusion Model

In this section, we train multiple versions of unconditional diffusion models [264] to demonstrate that ProMark can be used to attribute a variety of concepts in the training data. In each case, the model is trained starting from random initialization of LDM weights. Described next are details of the datasets and evaluation protocols.

Datasets We use 5 datasets spanning attribution categories like image templates, scenes, objects, styles, and artists. For each dataset, we consider the dataset classes as our attribution categories. For each class in a dataset, we use 90% images for training, and 10% for evaluation, unless specified otherwise.

1. Stock: We collect images from Adobe Stock, comprising of near-duplicate image clusters like templates, symbols, icons, *etc*.. An example image from some clusters is shown in the supplementary. We use 100 such clusters, each with $2K$ images.

2. LSUN: The LSUN dataset [363] comprises 10 scene categories, such as bedrooms and kitchens. It's commonly used for scene classification, training generative models like GANs, and anomaly detection. Same as the Stock dataset, we use $2K$ images per class.

3. Wiki-S: The WikiArt dataset [296] is a collection of fine art images spanning various styles and artists. We use the 28 style classes with 580 average images per class.

Table 5.2 Comparison with prior works for unconditional diffusion model on various datasets. [Keys: str.: strength].

Method	Str. (%)	Attribution Accuracy (%) ↑				
		Stock	LSUN	Wiki-A	Wiki-S	ImageNet
ALADIN [269]	-	99.86	46.27	48.95	33.25	9.25
CLIP [251]	-	75.67	87.13	77.58	60.84	60.12
F-CLIP [328]	-	78.49	87.39	77.23	60.43	62.83
SSCD [246]	-	99.63	73.26	69.51	50.37	37.32
EKILA [11]	-	99.37	70.60	51.23	37.06	38.00
ProMark	30	100	95.12	97.45	98.12	83.06
	100	**100**	**100**	**100**	**100**	**91.07**

4. Wiki-A: From the WikiArt dataset [296] we also use the 23 artist classes with 2, 112 average images per class.

5. ImageNet: We use the ImageNet dataset [71] which comprises of 1 million images across $1K$ classes. For this dataset, we use the standard validation set with $50K$ for evaluation and the remaining images for training.

Evaluation Protocol For all datasets, the concept attribution performance is tested on the held-out data as follows. For a held-out image, we first encrypt it with the concept's watermark. Then using the latent code of the encrypted image, we noise it till a randomly assigned timestamp and apply our trained diffusion model to reverse back to the initial timestamp with the estimated noise. The denoised latent code is then decoded using the autoencoder $\mathcal{D}_L(.)$, and the embedded secret is predicted using the secret decoder $\mathcal{D}_S(.)$. Using Eq. (5.9), we compute the predicted concept and calculate the accuracy using the ground-truth concept.

Results Shown in Tab. 5.2 is the attribution accuracy of ProMark at two watermark strengths *i.e.* 100% and 30% which is set by variable m in Eq. (5.2). ProMark outperforms prior works, achieving near-perfect accuracy on all the datasets when the watermark strength is 100%. However, the watermark introduces visual artifacts [27] if the watermark strength is full. Therefore, we decrease the watermark strength to 30% before adding it to the training data (see Sec. 5.4.5 for ablation on watermark strength). Even though our performance drops at a lower watermark strength, we still outperform the prior works. This shows that our causal approach can be used to attribute a

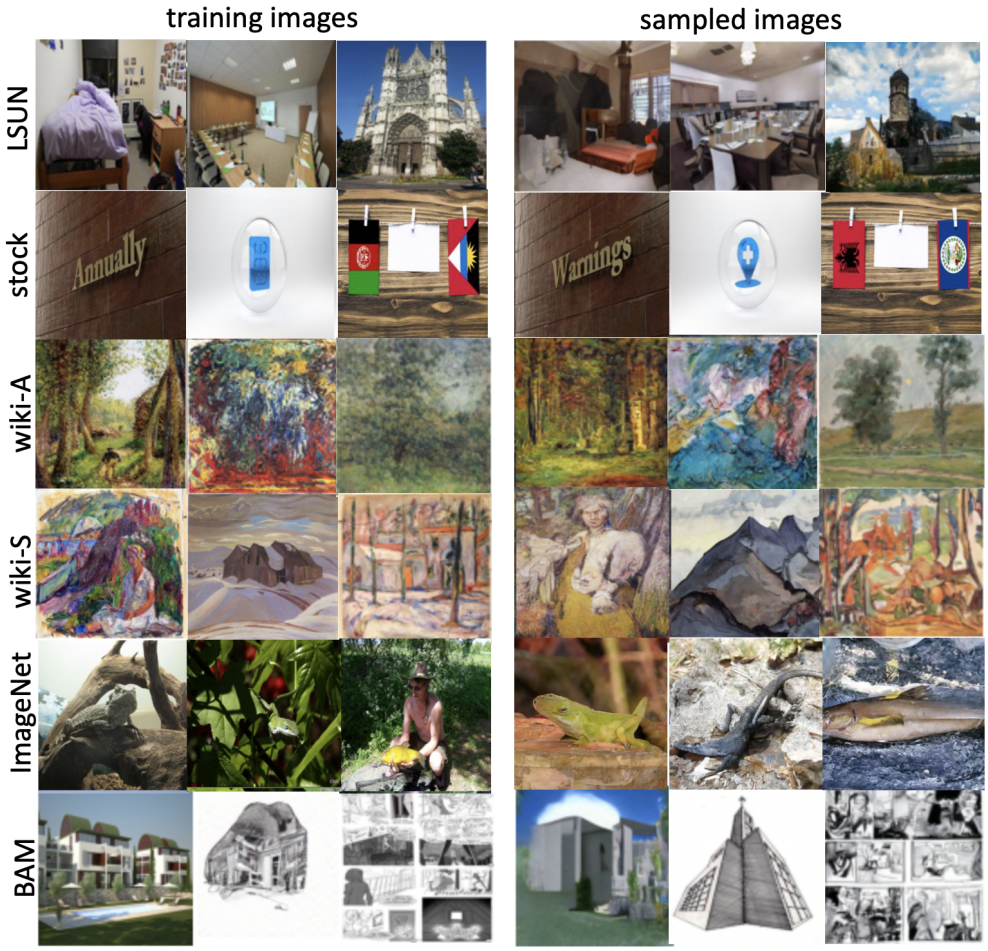


Figure 5.3 **Example training and newly sampled images of different datasets for the corresponding classes**. We observe a similar content in the inference image compared with the training image of the predicted class.

variety of concepts in the training data with an accuracy higher than the prior passive approaches.

Fig. 5.3 (rows 1-5) shows the qualitative examples of the newly sampled images from each of the trained models. For each model, we sample the images using random Gaussian noise until we have images for every concept. The concept for each image is predicted using the secret embedded in the generated images. Shown in each row of Fig. 5.3 are three training images (columns 1-3) and three sampled images from the corresponding concepts (columns 4-6). This shows that ProMark makes the diffusion model embed the corresponding watermark for the class of the generated image, thereby demonstrating the usefulness of our approach.

Shown in Fig. 5.4 are the nearest images retrieved using the embedding-based methods (row (2)-(6)) for the query images from the ImageNet (row (1)). For each image retrieval, we highlight

Figure 5.4 **Visual results of prior embedding-based works.** We show the image of the closest matched embedding for each method on ImageNet. We highlight images green for correct attribution, otherwise red. Embedding-based works do not always attribute to the correct concept.

the correct/incorrect attribution using a green/red box. As we can see, the correlation-based prior techniques rely on visual similarity between the query and the retrieved images, ignoring the concept. However, for each query image, ProMark predicts the correct concept corresponding to the query image (Fig. 5.3).

Table 5.3 Multi-concept attribution comparison with baselines.

Method	Strength (%)	Attribution Accuracy (%) ↑		
		Media	Content	Combined
ALADIN [269]	-	42.16	41.25	34.97
CLIP [251]	-	46.71	45.12	42.36
F-CLIP [328]	-	52.12	51.56	46.23
SSCD [246]	-	47.06	46.09	40.61
EKILA [11]	-	43.72	43.58	37.09
ProMark (single)	30	-	-	**97.73**
ProMark (multi)	30	91.33	89.21	84.66
	100	**95.61**	**93.31**	90.12

5.4.2 Multiple Watermarks

We evaluate the effectiveness of ProMark for multi-concept attribution. As before, an unconditional diffusion model is trained starting from random initialization, and each image in the training data is encrypted with two watermarks as outlined in Sec. 5.3.6.

Dataset For this experiment, we use the BAM dataset [337], comprising contemporary artwork sourced from Behance, a platform hosting millions of portfolios by professionals and artists. This dataset uniquely categorizes each image into two label types: media and content. It encompasses 7 distinct labels for media and 9 for content, culminating in a diverse set of 63 label pairs, with 4,593 average images in these label pairs. For each class pair, we use 90% data for training and 10% for held-out evaluation.

Results The same evaluation is performed as described in Sec. 5.4.1, except the accuracy is now computed for two concepts instead of one. Shown in Tab. 5.3 is the attribution accuracy for the two concepts individually and simultaneously. To benchmark the effectiveness of ProMark, we also compare against baselines, where ProMark outperforms baselines, achieving a combined attribution accuracy of 90.12% as compared to 46.61% for F-CLIP [328]. We believe our findings substantiate that ProMark can be extended to a scenario where the generated images are composed of several unique concepts from the training images. For ablation, we train ProMark with 7×8 classes, with each pair of media and content as an individual concept. ProMark is able to achieve 97.73% attribution accuracy for single-concept, higher than the performance achieved for multi-concept case *i.e.*90.12%. However, single concept approach is not scalable when the number of concepts

Table 5.4 Comparison with different baselines for the conditional model trained on ImageNet dataset.

Method	Strength (%)	Attribution Accuracy (%) ↑	
		Held-out data	New images
ALADIN [269]	-	9.25	0.18
CLIP [251]	-	60.12	41.01
F-CLIP [328]	-	62.83	50.19
SSCD [246]	-	37.32	30.10
EKILA [11]	-	38.00	29.06
ProMark	30	91.24	87.30
	100	**95.60**	**90.13**

in an image increases, as the number of watermarks would grow exponentially (7×8 *vs*.$7 + 8$). Therefore, transitioning to a multi-concept scenario is more appropriate for real-world scenarios, where scalability and practicality are crucial.

In the final row of Fig. 5.3, we present qualitative examples of newly sampled images from the model trained on the BAM dataset. Observations indicate that these sampled images successfully adopt both media and content corresponding to training images of the same concept. This provides empirical evidence of ProMark's effectiveness in facilitating multi-concept attribution.

5.4.3 Number of Concepts

AI models leverage large-scale image datasets [264, 20, 132, 216], encompassing a broad spectrum of concepts. This diversity necessitates concept attribution methods that can maintain high performance across numerous concepts. In this context, we test ProMark with an exponentially increasing number of concepts. Our dataset comprises Adobe Stock images with near duplicate image templates (used as concepts). As we escalate the number of concepts, we concurrently reduce the per-concept image count, only 24 images per concept for 2^{16} concepts, see the red curve of Fig. 5.5 (a) for image count. This is done to obtain balanced image distribution and also to challenge ProMark's robustness.

The outcomes, depicted in Fig. 5.5(a) red curve, indicate an anticipated decline in ProMark's efficacy in line with the increase in the number of concepts, reducing from 100% attribution accuracy for 10 concepts (chance accuracy 10%) to 82% for 2^{16} concepts (chance accuracy $1.5e$-3%). This reduction in attribution accuracy is correlated with the reduction in bit-secret

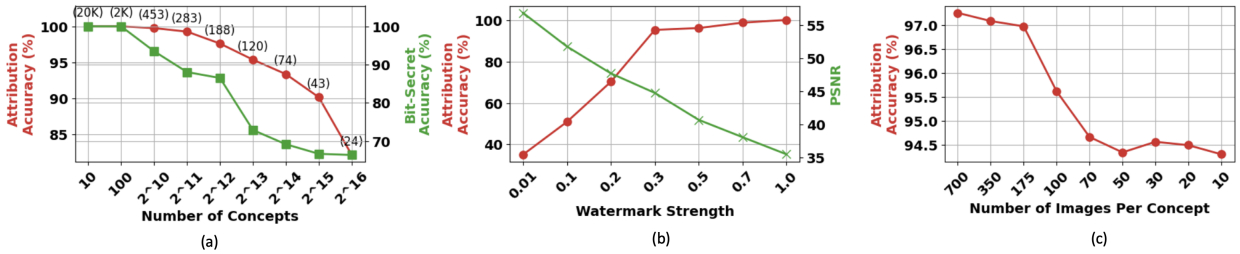


Figure 5.5 **Ablation experiments**: We show the results for ablating multiple parameters of ProMark. (a) Number of concepts, (b) watermark strength, and (c) number of images per concept.

accuracy (green curve) for every predicted secret, indicating poor watermark recovery due to the increased confusion between the watermarks. Notwithstanding the increased difficulty, ProMark demonstrates commendable performance, underscoring its potential in real-world applications.

5.4.4 Conditional Diffusion Model

As the diffusion models are usually trained with conditions to guide generation, we also evaluate using the conditional LDM model [264]. For this, we fine-tune a model pretrained of the ImageNet dataset (see Sec. 5.4.1), where the 1000 ImageNet classes are used as model conditions and also as the 1000 concepts.

Evaluation Protocol In addition to the evaluation on the held-out data (see Sec. 5.4.1), we also perform the quantitative evaluation on the newly sampled images as follows. We use the labels of the ImageNet dataset as conditions to sample $10K$ images (10 images per label). Using these labels as the ground-truth concept for a newly sampled image, we compute the accuracy of the concept predicted by the embedded watermark in the generated images.

Results The accuracies for held-out and newly sampled images are shown in Tab. 5.4. The performance on the held-out dataset for the conditional model improves compared to the unconditional models as the label conditions provide improved supervision for correct watermarks. ProMark also outperforms prior embedding-based works by a large margin on both held-out and newly sampled images. The attribution accuracy on the new images, however, is less than the held-out data. We hypothesize that it is because newly sampled images may contain more than one concept and can be more confusing to attribute. The high accuracy, even for newly sampled images, suggests that ProMark exhibits higher generalizability to unseen synthetic images.

5.4.5 Ablation Study

For the ablation experiment, we use Stock dataset with a varying number of concepts, and we train unconditional LDM models from random initialization.

Strength of Watermark. The hyperparameter m in Eq. (5.2) modulates the intensity of the watermark applied to the training images, ensuring encrypted images retain high quality. We systematically alter m to examine its impact on the LDM's performance and the Peak Signal-to-Noise Ratio (PSNR) of the output images with reference to the held-out encrypted images. Fig. 5.5(b) shows that attribution accuracy improves with increased m, plateauing beyond a threshold of 0.5. The discernible compromise in image quality, as evidenced by the inverse relationship between intensity and PSNR, can be attributed to the use of fixed watermarks obtained using RoSteALS [27], which is originally optimized for robustness. In light of this, we select an optimal watermark strength of 0.3, which balances between performance and PSNR. We measured the FID between original and newly sampled images from a pretrained ImageNet conditional model (trained without watermark) and ProMark model (trained with watermark), which is 13.28 and 17.63 respectively. This small increment shows negligible quality loss in the generated images due to ProMark.

Number of Images Per Concept. To ascertain the optimal number of images required per concept for effective watermark learning, we ablate by fixing the number of concepts to 500 and varying the number of images used to train the LDM. Fig. 5.5(c) reveals that performance drops by 2.5% when image count per concept is reduced from 700 to 10. Remarkably, the general efficacy of ProMark remains consistently high, suggesting a low sensitivity to the image count per concept. These results demonstrate that ProMark can successfully learn watermarks with as few as 10 images per concept, highlighting its efficiency and potential for applications with limited data availability.

Framework Design. ProMark employs BCE loss to instruct the LDM model in the accurate embedding of bit-sequence watermarks within generated images. The attribution performance degrades to 2% when BCE loss is not used as compared to 100% in Tab. 5.2. This shows that removing BCE loss significantly impairs the LDM's performance, underscoring the necessity of

this supervision in helping LDM embed watermarks effectively.

Also, ProMark incorporates a secret decoder to retrieve secret bit-sequence from synthesized images, rendering the process contingent upon the pretrained secret decoder. In contrast, prior works [6, 7, 5] recover watermarks by training a dedicated decoder with the main model in an end-to-end fashion. To ablate this alternative approach, we train a standard decoder along with LDM by optimizing for the cosine similarity between the embedded and extracted watermarks. We see a degradation in performance from 100% to 80.56%, indicating that the pretrained secret decoder is a better choice for our approach. This is due to the increased complexity of predicting watermarks of resolution 256^2 as compared to 160-bit sequence from the encrypted images.

5.5 Conclusion

We introduce a novel proactive watermarking-based approach, ProMark, for causal attribution. We use predefined training concepts like styles, scenes, objects, motifs, *etc*.. to attribute the influence of training data on generated images. We show ProMark's is effective across various datasets and model types, maintaining image quality while providing more accurate attribution on a large number of concepts. Our approach can also be extended to multi-concept attribution by embedding multiple watermarks onto the image. Finally, for each experiment, our approach achieves a higher attribution accuracy than the prior passive approaches. Such attribution offers opportunities to recognize and reward creative contributions to generative AI, underpinning new models for value creation in the future creative economy [57].

Limitations. In evaluating ProMark, we note a trade-off between image quality and attribution accuracy, which may need us to learn watermarks for attribution task. Our model is currently trained with predefined concepts and further research is needed on training paradigm when new concepts are introduced. While we use orthogonal watermarks for varied concepts like motifs and styles, this may not accurately reflect the interrelated nature of some concepts, suggesting another opportunity for future research. Finally, our results are specific to the LDM, and extending this approach to other GenAI models could provide a better understanding of ProMark's effectiveness.

CHAPTER 6

CUSTOMMARK: CUSTOMIZATION OF DIFFUSION MODELS FOR PROACTIVE ATTRIBUTION

Generative AI (GenAI) presents challenges in attributing synthesized content to its original training data, particularly for artists whose styles are replicated by these models. We introduce CustomMark, a novel technique for customizing pre-trained text-to-image GenAI models to enable attribution. With CustomMark, text prompts can be modified to embed a watermark in generated images, linking them to training concepts such as an artist's style, specific objects, or the GenAI model itself. Our approach supports sequential customization, allowing new concepts to be attributed efficiently and scalably without retraining from scratch. We demonstrate that CustomMark can robustly watermark hundreds of individual concepts and support multiple attributions within a single image while preserving high visual quality of the generation[1].

6.1 Introduction

Given GenAI's potential to democratize creativity, ethical concerns have emerged among artists regarding the unauthorized use of their works. Many seek recognition or compensation for the derivative use of their styles in generated images [263]. In the past, such creative recognition has relied on collaborations between technology, legal frameworks, and artistic practices [24]. GenAI currently lacks such mechanisms, leading to artist discontent and prompting adversarial strategies like "Glaze" [276], "Anti-DreamBooth" [309], and others [381, 96, 88] to protect their works.

To address this discontent, it is needed that GenAI models provide attribution when generated images are derived from artists' works in training data. Such attribution could potentially unlock new revenue streams in the creator economy, rewarding creative opt-in to GenAI training [57]. A decentralized framework to compensate creators based on visual similarities between generated and training images was proposed in [11]. Several similarity embeddings have been explored [269, 11, 328] to determine the subset of training images that influenced the generation. While intuitive, these visual correlation-based attribution methods [269, 11, 328] often fail to provide definitive

[1] *Vishal Asnani*, John Collomosse, Xiaoming Liu, and Shruti Agarwal. "CustomMark: Customization of Diffusion Models for Proactive Attribution." In review, 2025.

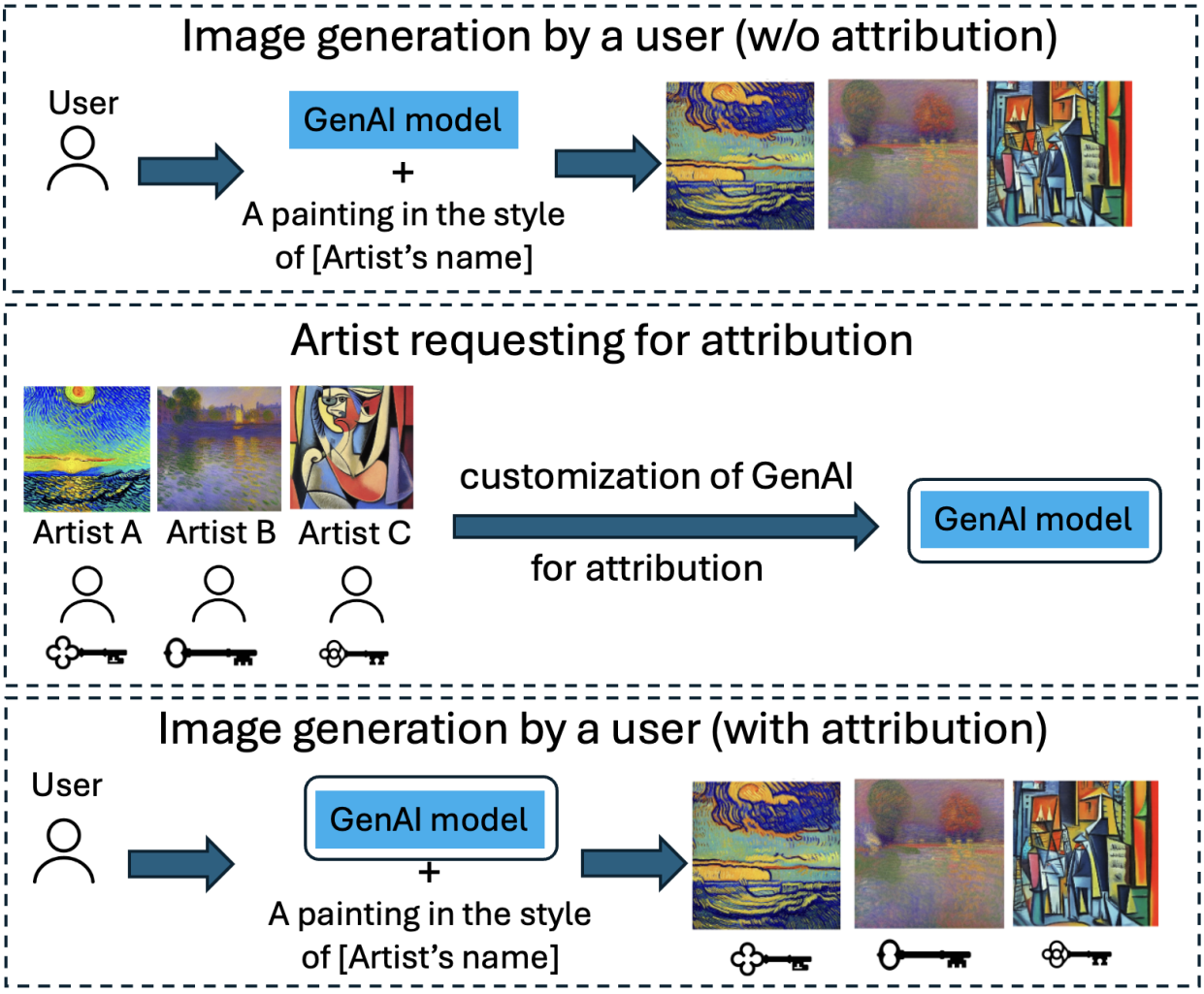


Figure 6.1 **Overview of concept attribution by GenAI models.** (a) A user generates images of various artists' style using artists' tokens in the prompt (w/o attribution). (b) Artists request to the companies to provide attribution for their work. Using CustomMark, companies customize their models to enable attribution only for the artists that have requested the same. (c) A user generates the images using the improved GenAI model with artists' specific watermark for attribution to the artists.

explanations and can also incorrectly attribute works not present in the training set.

Alternative approaches attempt to establish direct causal relationships using techniques like proactive watermarking [4] or influence estimation via data removal [329]. However, these methods require modifications to training data or inference paradigms, making them computationally heavy.

In response, we propose CustomMark, an efficient technique for attribution in pre-trained GenAI models. Similar to [4], we use concept-specific watermarking but without requiring predefined concepts before training. CustomMark enables selective attribution of specific concepts in a pre-

trained model, supporting sequential learning for newly emerging seen or unseen concepts. This approach avoids exhaustive retraining and allows attribution only for relevant concepts.

As shown in Fig. 6.1, we focus on attribution in text-to-image Latent Diffusion Models (LDMs), where attributable concepts appear in prompts, such as "A painting in the style of V*" or "An image of V*." If the owner of concept V* requests attribution, CustomMark embeds a concept-specific watermark into generated images while preserving visual quality. Unlike [4], which attributes to a subset of training images, CustomMark directly attributes the concept itself. The watermark remains robust against non-editorial modifications, ensuring traceability to the original concept and the GenAI model as the image circulates online. Since CustomMark embeds watermarks in a concept-specific manner without requiring exhaustive retraining, it effectively functions as a form of model customization.

Current customization methods [148, 171, 366, 219, 375, 268, 89, 351, 75, 278] struggle to scale across many distinct concepts, often compromising generation quality. To address this, we propose a novel architecture that customizes pretrained LDMs for large-scale watermarking. Building on [88], we use a concept encoder to map a bit-secret to token-embedding perturbations but find it insufficient for scalability. Thus, we introduce a mapper network that perturbs input Gaussian noise, we fine-tune the LDM's attention layers, and leverage CSD [286] loss for faster training and improved image quality. CustomMark enables fine-tuned LDMs to generate watermarked images aligned with text prompts while embedding corresponding watermarks. Its sequential learning capability allows new attributions with just 10% additional finetuning, preserving visual quality while protecting artist styles. Our contributions are:

1. An efficient, scalable technique to customize LDMs for imperceptibly watermarking single or multiple seen/unseen concepts in a generated image, enabling robust concept attribution in pre-trained text-to-image LDMs.

2. Sequential attribution capability, allowing fine-tuning for new concepts dynamically without retraining the model, ensuring selective attribution of relevant seen and unseen concepts.

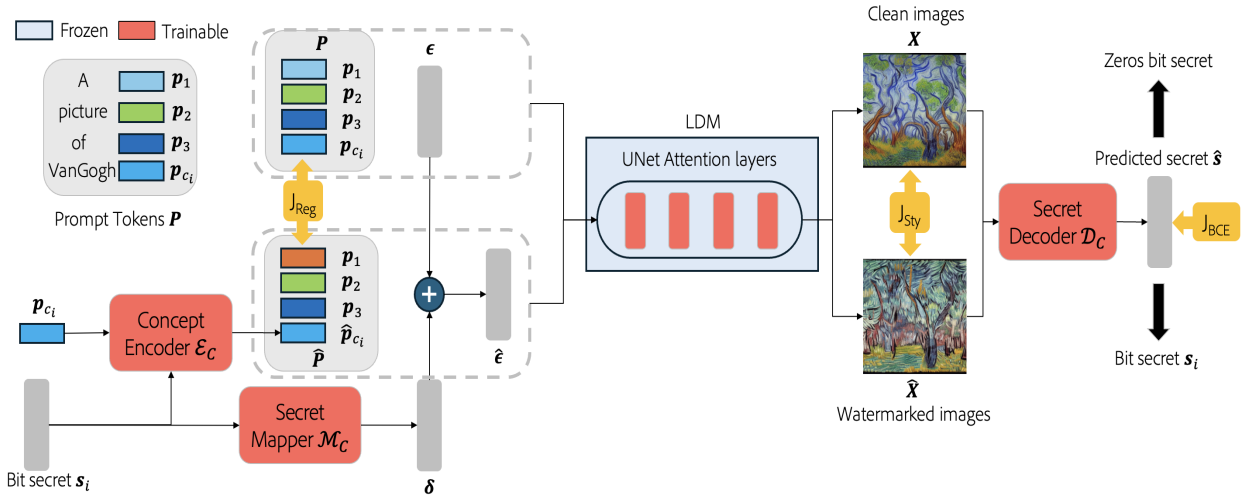


Figure 6.2 **Overview of CustomMark.** Illustrating the training workflow for CustomMark. A concept token $\boldsymbol{p}_{c_i}$ is encoded through the Concept Encoder $\mathcal{E}_C$ to generate a modified prompt $\hat{\boldsymbol{p}}_{c_i}$ with embedded watermark information. The Secret Mapper $\mathcal{M}_C$ maps a bit secret $\boldsymbol{s}_i$ to perturb the concept token, producing δ, which is added to the Gaussian noise ϵ. The LDM using the prompt tokens and pertubed Gaussian noise, producing watermarked images $\hat{\boldsymbol{X}}$ that carry the bit secret in visual form. During inference, the Secret Decoder $\mathcal{D}_C$ extracts the bit secret from watermarked image $\hat{\boldsymbol{X}}$ and the clean image $\boldsymbol{X}$ to extract the bit secret. CustomMark is guided by various constraints, namely regularization loss J_{Reg} to make the artist token embedding similar, style loss J_{Sty} to maintain style consistency between clean and watermarked images, and the bit secret loss J_{BCE} to predict the added bit secret. Best viewed in color.

.

3. Demonstration that diffusion models can attribute 100s of artists' styles and 1000 ImageNet classes while maintaining high visual quality of watermarked concepts.

6.2 Related Works

Proactive Schemes. Proactive methods enhance various tasks by embedding perturbations into input images, providing benefits to deepfake tagging [325], detection of manipulated content [6], localization of manipulations [7], object detection [5, 88], and concept attribution [4]. Some approaches focus on altering the training data to disrupt the output of generative models [356, 267]. Meanwhile, Alexandre *et al*. [270] introduce a fixed signal method to enable attribution of training datasets. Recently, a survey by Asnani *et al*. [9] discusses various proactive approaches, encryption schemes, learning process, and their applications, such as vision model defense [298, 342], LLM defense [221, 341, 378], privacy protection [299, 343, 383, 252], improving GenAI models [285, 163, 205, 237, 173, 373], 3D domain [134, 151, 374, 305, 359, 146], *etc*.. In CustomMark, we use

proactive techniques to do concept attribution in an efficient and scalable manner, with a focus on practical application to the real-world scenarios.

IP Protection and Concept Attribution. For IP protection of AI-generated models and content, watermarking techniques embed signals into outputs via model fine-tuning [90], prompt verification [380, 198], and token-level adjustments [165]. Tools like DiffusionShield [62] and detection watermarking [241] prevent misuse, while on the other hand latent fingerprinting [8] and audio watermarking [31] extend protection across media. Additional model security is provided by DeepSigns [67], DeepMarks [39], and network embedding [320], as well as deep spatial encryption [369], backdoor triggers [1], and dynamic defenses like DAWN [293].

Concept attribution identifies which training data influenced a generated output, distinct from model [28] or camera attribution [37]. Traditional methods passively assess visual similarities between generated and training images using predefined criteria. For instance, Wang *et al*. [328] propose Attribution by Customization (AbC), modifying embeddings like CLIP and DINO with customized diffusion models. Style-specific attribution methods such as ALADIN [269] and EKILA [11] employ perceptual hashing for patch-based matching. MONTRAGE [26] monitors weight updates to attribute pre-trained concepts, while Asnani *et al*. [4] embed concept-specific watermarks in training images for direct attribution. In contrast, we introduce a proactive watermarking technique that requires no training data modifications and enables selective, sequential attribution after training.

GenAI Customization. Advances in GenAI customization leverage techniques like Video Motion Customization [148], Custom Diffusion [171], and CustomNet [366] to adapt models to specific concepts and motions, while approaches like Modular Customization [247] and CIDM [77] enhance scalability and prevent catastrophic forgetting. Efficiency-focused methods [75] and LoRA-Composer [351] optimize customization with minimal parameter adjustments, while AquaLoRA [89] provides watermarking for unauthorized use protection, and textual inversion [219, 375, 268] enables precise text-based editing. Privacy-oriented anti-customization [317] offers additional security by adapting adversarial strategies. We propose a proactive concept attribution technique

using model customization, which hasn't been explored before.

6.3 Method

6.3.1 Background

Prompts and Cross-Attention Mechanism in Diffusion Model. In text-to-image LDMs [264], prompts and cross-attention mechanisms work together to guide image generation. A prompt is processed by a text encoder, and converted into a *text embedding*. This embedding conditions the sampling process by capturing the prompt's meaning. Instead of merely producing random images, the cross-attention mechanism allows the model to "attend" to specific parts of the text embedding, guiding the diffusion process to align the output with the input prompt. For key $\boldsymbol{K}$, query $\boldsymbol{Q}$ and value $\boldsymbol{V}$, the scaled dot-product attention is given by:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}. \tag{6.1}$$

Further, multi-head cross attention with respective weight matrices $\boldsymbol{W}_i^*$s, is utilized to improve generation quality by processing the prompt with multiple attention heads:

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Concat}(\text{H}_1, \ldots, \text{H}_h)\boldsymbol{W}, \tag{6.2}$$

$$\text{H}_i = \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V). \tag{6.3}$$

As the multi-head cross-attention in Eq. (6.3) is the main component to establish a relationship between prompts and the generated image, in CustomMark, we only fine-tune $\boldsymbol{W}_i^*$s. This significantly reduces training time while enhancing critical associations between the concept and its watermarked image.

Concept Attribution. ProMark [4] defines the concept attribution as finding the closest concept in the training dataset for a given generated image. For this purpose, ProMark divides the entire dataset into different concepts and trains with each concept being watermarked. However, this is impractical for the real world as it difficult to retrain the GenAI models on the entire watermarked data. Therefore, we re-define the problem of Concept Attribution as follows.

Let C represent a set of N distinct concepts within the training dataset of a GenAI model. Out of the N concepts, let $\hat{C} = \{c_1, c_2, \ldots, c_M\}$ be the M concepts that need attribution, whose token

embeddings are represented as $\boldsymbol{P}_c = \{\boldsymbol{p}_{c_1}, \boldsymbol{p}_{c_2}, \ldots, \boldsymbol{p}_{c_M}\}$. Given a synthetic image $\boldsymbol{X}$ generated by a GenAI model using $\boldsymbol{p}_{c_i} \in \boldsymbol{P}_c$, along with other prompt token embeddings, forming an input prompt $\boldsymbol{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{c_i}, \ldots, \boldsymbol{p}_n\}$, the objective of concept attribution is to map $\boldsymbol{X}$ to its corresponding concept c_i. Specifically, we find a mapping function f such that $c_i = f(\boldsymbol{X})$.

6.3.2 CustomMark

Overview. To add attribution capabilities to a pre-trained LDM, CustomMark perturbs the inputs to the LDM and fine-tunes its attention weights. The input token embedding $\boldsymbol{p}_{c_i}$ and the input Gaussian noise $\boldsymbol{\epsilon}$ are perturbed by the concept encoder $\mathcal{E}_C$ and the secret mapper $\mathcal{M}_C$ networks, that encode a concept specific bit-secret into the respective inputs. This results in the perturbed embedding $\hat{\boldsymbol{p}}_{c_i}$ and the perturbed Gaussian noise $\hat{\boldsymbol{\epsilon}}$, which are fed into the LDM to sample new images. The synthesized images are then fed to the secret decoder $\mathcal{D}_C$ that outputs the corresponding bit-secret. During training, only the attention weights in Eq. (6.2), and Eq. (6.3) of the LDM are fine-tuned. The framework is guided by several constraints which allows for the generation of images with embedded secrets and also maintain the original artistic style. We will now present our method in details.

Embedding Encryption. In CustomMark we perturb all the concepts in $\boldsymbol{P}_c$ using a single concept encoder $\mathcal{E}_C$. For i^{th} concept, the concept token embedding $\boldsymbol{p}_{c_i}$ is encrypted using $\mathcal{E}_C$ as:

$$\hat{\boldsymbol{p}}_{c_i} = \mathcal{E}_C(\boldsymbol{p}_{c_i}, \boldsymbol{s}_i), \tag{6.4}$$

where $\boldsymbol{s}_i$ is the concept specific bit-secret of length l, i.e. $\boldsymbol{s}_i = \{b_{i1}, b_{i2}, ..., b_{il}\}$ where $b_{ij} \in \{0, 1\}$.

After encryption, the original embedding is replaced by the encrypted text embedding, resulting in encrypted prompt token embeddings $\hat{\boldsymbol{P}} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \hat{\boldsymbol{p}}_{c_i}, \ldots, \boldsymbol{p}_n\}$. To obtain the watermarked image, $\hat{\boldsymbol{P}}$ is fed to the LDM in place of the original token embeddings $\boldsymbol{P}$. Following the architecture of [88], we apply a regularization mean squared error (MSE) loss between $\boldsymbol{P}$ and $\hat{\boldsymbol{P}}$ at initial iterations, so that the encoder $\mathcal{E}_C$ has a good starting point to preserve the style, and support secret

learning. The regularization loss is:

$$J_{Reg} = ||\hat{\boldsymbol{P}} - \boldsymbol{P}||_2^2. \tag{6.5}$$

Secret Learning. We will now discuss the learning of LDM to generate watermarked images given the encrypted token embeddings $\hat{\boldsymbol{P}}$. In addition to $\mathcal{E}_C$, we use a mapper network $\mathcal{M}_C$ to further accelerate the secret learning. Using i^{th} bit-secret $\boldsymbol{s}_i$, we estimate a perturbation $\boldsymbol{\delta} = \mathcal{M}_C(\boldsymbol{s}_i)$ which is added to the initially sampled Gaussian noise $\boldsymbol{\epsilon}$ for image generation. Therefore, the perturbed $\boldsymbol{\epsilon}$ is given by:

$$\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon} + \alpha \times \mathcal{M}_C(\boldsymbol{s}_i), \tag{6.6}$$

where α controls the magnitude of $\boldsymbol{\delta}$. The perturbed Gaussian noise $\hat{\boldsymbol{\epsilon}}$ along with $\hat{\boldsymbol{P}}$ is given as input to the LDM to sample an image. Finally, to avoid the complexity of LDM training, we only finetune the attention layers of the LDM, while fixing other layers.

During training, we create both clean and watermarked images, $\boldsymbol{X}$ and $\hat{\boldsymbol{X}}$, using the inputs ($\boldsymbol{P}$, $\boldsymbol{\epsilon}$) and ($\hat{\boldsymbol{P}}$, $\hat{\boldsymbol{\epsilon}}$). The style descriptors $\boldsymbol{d}$ and $\hat{\boldsymbol{d}}$ from images $\boldsymbol{X}$ and $\hat{\boldsymbol{X}}$ are extracted using the pretrained Contrastive Style Descriptors (CSD) [286] model. CSD contain concise and effective style information, while being invariant to semantic content and capable of disentangling multiple styles. We maximize the cosine similarity between two descriptors, which ensures that the watermarked images matches the style of original concept. To further support style matching, we apply a MSE loss between the two images, in addition to the CSD loss. Therefore, our style loss is given by:

$$J_{Sty} = 1 - \cos(\hat{\boldsymbol{d}}, \boldsymbol{d}) + ||\boldsymbol{X} - \hat{\boldsymbol{X}}||_2^2. \tag{6.7}$$

$\boldsymbol{X}$ and $\hat{\boldsymbol{X}}$ are further fed to a secret decoder $\mathcal{D}_C$, which estimates the bit secret in given images. The decoder shall output a zeros secret for $\boldsymbol{X}$, and the secret $\boldsymbol{s}_i$ for $\hat{\boldsymbol{X}}$. To train $\mathcal{D}_C$, we use a binary cross-entropy (BCE) loss between the ground truth bit-sequence $\boldsymbol{s}_i$ and the predicted one $\hat{\boldsymbol{s}}_i$:

$$J_{BCE}(\boldsymbol{s}_i, \hat{\boldsymbol{s}}_i) = -\frac{1}{l}\sum_{j=1}^{l}[b_j \log(\hat{b}_j) + (1 - b_j)\log(1 - \hat{b}_j)]. \tag{6.8}$$

Therefore, CustomMark is trained in an end-to-end manner to minimize the objective $L_{attr} = L_{Sty} + L_{BCE} + \beta L_{Reg}$ during training, where $\beta = 10$ for our experiments.

During inference, if the random Gaussian noise and the input prompt are perturbed, the diffusion model embeds a watermark within the generated image. This watermark can be decoded using $\mathcal{D}_C$ to the concept specific bit-secret, functioning as hidden signatures for attribution.

Concept Attribution in Inference. To attribute the generated images, we extract the bit secret embedded by the LDM using $\mathcal{D}_C$. Using this predicted bit-secret $\hat{\boldsymbol{s}} = \mathcal{D}_C(\hat{\boldsymbol{X}})$ and the bit-secret $\boldsymbol{s}_i$ corresponding to the concept c_i, we define the attribution mapping function f as:

$$f(\hat{\boldsymbol{X}}) = \underset{i \in [1,M]}{\operatorname{argmax}}\, g(\mathcal{D}_C(\hat{\boldsymbol{X}}), \boldsymbol{s}_i), \tag{6.9}$$

where,

$$g(\mathcal{D}_C(\hat{\boldsymbol{X}}), \boldsymbol{s}_i) = g(\hat{\boldsymbol{s}}, \boldsymbol{s}_i) = \sum_{k=1}^{l} [\hat{b}_k = b_{ik}], \tag{6.10}$$

and $[\hat{b}_k = b_{jk}]$ is an indicator function that returns 1 if the bits match, and 0 otherwise. Thus, using the predicted bit-sequence we assign the generated images to the concept whose bit-sequence matches the best, *i.e.*, the i^{th} concept that maximizes $g(\hat{\boldsymbol{s}}, \boldsymbol{s}_i)$.

6.3.3 Sequential Learning

In real-world scenarios, the number of concepts requiring attribution is not always fixed. The set of concepts can change frequently, making it impractical to retrain the attribution model from scratch each time new concepts are introduced. To address this challenge, we propose the idea of sequential learning with CustomMark.

For example, if CustomMark is initially trained on M concepts, denoted as $\hat{C} = \{c_1, c_2, \ldots, c_M\}$, and a new concept c_{M+1} needs to be attributed, the model can be fine-tuned on the expanded set $\hat{C} \cup c_{M+1}$, starting from the model pretrained on $\hat{C}$. This approach allows the model to adapt to new concepts without requiring a predefined set during initial training. Our experiments demonstrate that learning new concepts in this manner requires only about 10% additional iterations, making it significantly more efficient than retraining CustomMark from scratch.

6.3.4 Multi-Concept Learning

In real-world text-to-image generation, multiple concepts are often combined within a single prompt, such as *"a painting of a dog in the style of Van Gogh."* To enable concept attribution in such

Figure 6.3 Comparison with ProMark [4] on ImageNet. ProMark produces low-quality images with bubble-like artifacts from its encryption, whereas CustomMark enables LDMs to generate high-quality images that closely match the original training concepts.

cases, CustomMark extends its attribution mechanism to handle multiple concepts simultaneously.

Given two concepts, c_i and c_j, from the attributed set $\hat{C}$, their respective token embeddings $\boldsymbol{p}_{c_i}$ and $\boldsymbol{p}_{c_j}$ are perturbed using the concept encoder $\mathcal{E}_C$. This results in the perturbed embeddings:

$$\hat{\boldsymbol{p}}_{c_i} = \mathcal{E}_C(\boldsymbol{p}_{c_i}, \boldsymbol{s}_i), \quad \hat{\boldsymbol{p}}_{c_j} = \mathcal{E}_C(\boldsymbol{p}_{c_j}, \boldsymbol{s}_j). \tag{6.11}$$

The perturbed prompt embeddings $\hat{\boldsymbol{P}} = \{\boldsymbol{p}_1, \ldots, \hat{\boldsymbol{p}}_{c_i}, \ldots, \hat{\boldsymbol{p}}_{c_j}, \ldots, \boldsymbol{p}_n\}$ are then used in the LDM to generate a watermarked image $\hat{\boldsymbol{X}}$. During decoding, the secret decoder $\mathcal{D}_C$ is designed to recover the **concatenated secret** associated with both concepts:

$$\hat{\boldsymbol{s}} = \mathcal{D}_C(\hat{\boldsymbol{X}}) = [\boldsymbol{s}_i; \boldsymbol{s}_j]. \tag{6.12}$$

The concatenation ensures that both concept-specific secrets are extracted from the generated image, thereby enabling attribution for multiple concepts simultaneously. The attribution function

f is then applied independently for each concept:

$$f(\hat{\boldsymbol{X}}) = \underset{i,j\in[1,M]}{\operatorname{argmax}}\, g(\mathcal{D}_C(\hat{\boldsymbol{X}}), [\boldsymbol{s}_i; \boldsymbol{s}_j]), \tag{6.13}$$

This approach ensures that CustomMark can reliably attribute both concepts in a multi-concept image, allowing for effective auditing of GenAI models even when multiple stylistic or semantic elements are present in the generated content.

6.4 Experiments

Implementation Details For training CustomMark, a predefined list of prompts are used per concept (see supp.). For concepts, we use 1,000 ImageNet [71] classes, 23 WikiArt [296] artists, and a custom 200 list of artists (see supp.). For text-to-image LDM, Stable Diffusion 1.5 is used. Unless stated, we use a bit-sequence of size 16. We evaluate CustomMark using four metrics: bit accuracy, attribution accuracy, CSD [286] score, and CLIP [328] score as described. For attribution assessment, bit accuracy: is the maximum percentage of bits matched between the predicted bit-secret and any of the concept-specific secret and attribution accuracy: is the percentage of times the predicted bit-secret matches with the correct concept-specific secret. For quality assessment, CSD score: is the cosine similarity between CSD descriptors, which assesses the style match between two images and CLIP score: is the cosine similarity between CLIP image embeddings. For all evaluation, we report average results on 100 generated and/or 100 clean images. For 10 concepts, CustomMark is trained for $20K$ iterations. All experiments are conducted on 8 A100 NVIDIA GPUs with a batch size of 8 per GPU.

6.4.1 Results

Comparison with Attribution Methods We evaluate various passive and proactive attribution methods on images generated by LDMs that are trained on the ImageNet and WikiArt datasets, containing 1000 and 23 classes, respectively. Here, each class is treated as a unique concept. For fair comparison, we generate 100 images per class for both ProMark [4] and CustomMark. Since ProMark and CustomMark embed different watermark, their accuracy is reported only on their respective 100 watermarked images. Whereas for passive methods, including ALADIN [269],

Table 6.1 Comparison with passive and proactive methods on images generated by conditional model trained on ImageNet and Wikiart dataset. CustomMark outperforms the passive methods on both datasets significantly. Both proactive methods have similar performance on ImageNet, but for Wikiart, CustomMark performs better than ProMark.

Method	**Type**	**Attribution Accuracy (%) ↑**	
		ImageNet	**Wikiart**
ALADIN [269]	Passive	5.55	18.58
CLIP [251]	Passive	42.61	52.60
AbC [328]	Passive	53.51	56.03
SSCD [246]	Passive	25.50	45.34
EKILA [11]	Passive	30.98	43.03
ProMark [4]	Proactive	**87.30**	87.19
CustomMark	Proactive	87.12	**89.25**

CLIP [251], AbC [328], SSCD [246], and EKILA [11], that rely on embeddings, the evaluation is done on images generated by both proactive models i.e. an average over a total of 200 generated images per concept. As shown in Tab. 6.1, the passive methods exhibit relatively low attribution accuracy.

In contrast, the proactive methods, ProMark and CustomMark, significantly outperform the passive methods, with much higher accuracy in both datasets. Although ProMark trains on an entirely watermarked dataset with all LDM parameters learnable in training, its performance is still comparable to CustomMark. Further, ProMark adversely impacts image quality, as shown in Fig. 6.3, where the generated ImageNet samples of ProMark are of lower quality and display visible artifacts. To quantify the quality, we calculate the FID score [131, 273] between the original ImageNet images (from a pretrained model without watermarks) and the watermarked images from each proactive model. The pretrained model achieves an FID score of 13.28. ProMark yields an FID score of 17.63, while CustomMark achieves an FID score of 14.73, indicating substantially better image quality. Thus, CustomMark not only maintains robust attribution performance but also generates higher-quality images than ProMark, making it a more effective solution for practical applications.

Comparison with Customization-Based Watermarking Methods We compare our method with [88], that also leverages textual token perturbations to guard personalized concepts. However, in [88], authors train new concept encoder-decoder pair for each personalization – an impractical

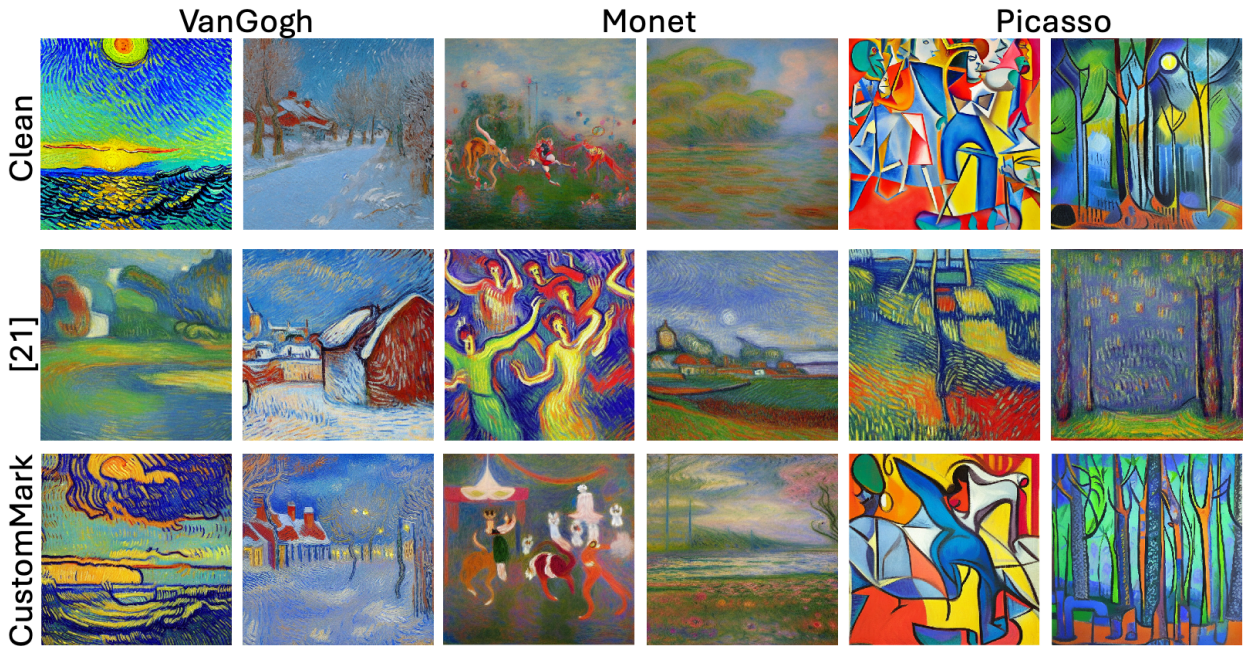


Figure 6.4 Attribution results of three concept artists: VanGogh, Monet and Picasso, sampled from LDM before and after applying the attribution capability of customization-based method [88] and CustomMark. [88] makes the LDM sample images far apart from the original style of artists, while CustomMark-watermarked images are much closer to the original style.

Table 6.2 Comparison with customization-based method by Feng *et al.* [88]. [KEYS: Acc.=Accuracy].

Method	**Bit Acc. (%)↑**	**Attribution Acc. (%) ↑**	**CLIP Score ↑**	**CSD Score ↑**
Feng *et al.* [88]	90.87	74.14	0.57	0.51
CustomMark	**99.29**	**94.29**	**0.81**	**0.77**

Figure 6.5 **Sequential learning of new concepts.** CustomMark starts with three initial concepts and incrementally learns new attributions without retraining from scratch. Each column displays clean and watermarked images, demonstrating CustomMark's efficiency in adapting to new styles with only about 10% extra training iterations per concept while maintaining high stylistic fidelity. We only show the concept used to create the image. A list of all the prompts used is given in supp.

solution in real world. For a fair comparison, we adapt [88] by training a single encoder-decoder pair for 3 artists' styles as concepts, namely VanGogh, Monet and Picasso. As shown in Tab. 6.2, CustomMark surpasses this baseline in all metrics, achieving higher watermark detection

accuracy (99.29), attribution accuracy (94.29), and generation quality (CSD score 0.81 and CLIP score 0.77). These results demonstrate the effectiveness of CustomMark for concept watermarking in GenAI.

Shown in Fig. 6.4 are some qualitative results for comparison. Unlike [88], which struggles to preserve individual artistic styles like brushstrokes and color palettes, CustomMark accurately captures each artist's unique nuances. For example, for Picasso (second row, last col), [88] generates Van Gogh style brushstrokes.

Sequential Learning In Fig. 6.5, we showcase CustomMark's sequential learning capability, where the model begins attribution with three concepts and subsequently integrates additional concepts one at a time. This setup reflects a dynamic, real-world setting where the need for concept attribution evolves over time as new styles are added. Instead of retraining the model from scratch for each new concept, CustomMark employs sequential learning to incrementally learn attributions for new concepts without erasing previously learned styles.

Starting with three initial concepts, CustomMark fine-tunes the model as new concepts are introduced, updating attribution while preserving distinct stylistic features. This is evident in the similarity between clean and watermarked images in each column, where CustomMark maintains high fidelity to the original style. With sequential learning, it attributes new concepts with only 10% additional iterations per concept, avoiding full retraining. These results demonstrate CustomMark's scalability and efficiency in preserving style-consistent, high-quality outputs for GenAI models.

Unseen Artists Watermarking We demonstrate CustomMark's ability to attribute both seen and unseen concepts using textual inversion. As shown in Fig. 6.4, known concepts are watermarked by perturbing their token embeddings. However, in real-world scenarios, generative models often encounter novel concepts outside the initial training set, requiring adaptability beyond predefined attributions.

To address this, we leverage textual inversion to derive token embeddings for unseen concepts. Once obtained, we apply watermark perturbations, enabling attribution without significant model retraining. Fig. 6.6 illustrates this by showing stylistic consistency between clean and watermarked

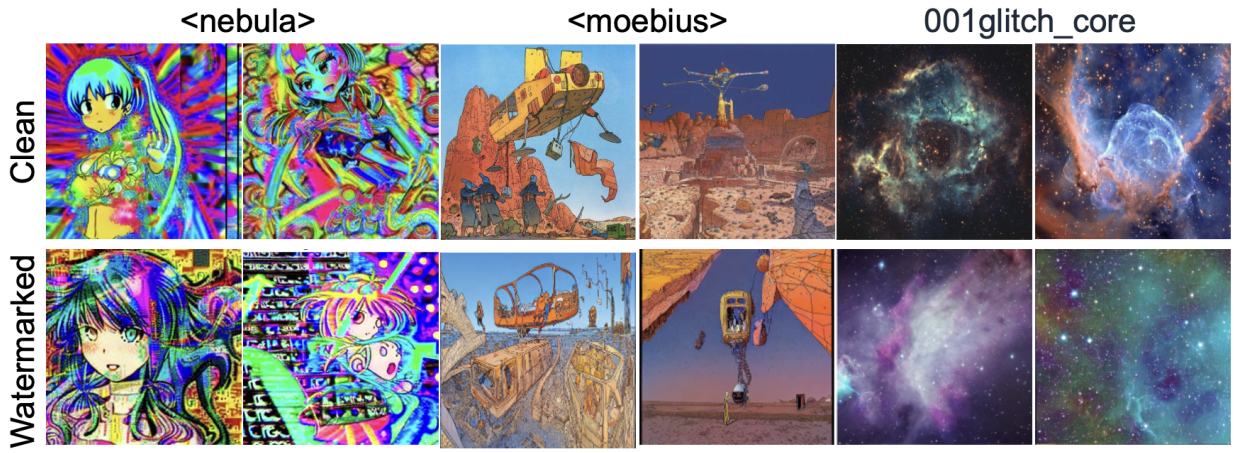


Figure 6.6 **Attribution of Unseen Concepts with CustomMark.** Shown is the CustomMark's ability to handle attribution for unseen concepts. The consistent style between clean and watermarked images across new styles demonstrates CustomMark's robustness in preserving artistic fidelity while achieving scalable attribution. We only show the concept used to create the image. A list of all the prompts used is given in supp.

images, preserving unique attributes of each new style. This demonstrates CustomMark's adaptability, allowing it to generalize to new styles while maintaining fidelity and stylistic integrity.

Multi-Concept Watermarking For this scenario, we take 20 concepts into consideration (10 objects, and 10 artists). Each concept is associated with an 8-bit secret. The decoder extracts a 16-bit secret for the generated image. The qualitative results in Fig. 6.7 demonstrate that CustomMark successfully embeds attribution signatures for both object (e.g., "dog," "tree") and style (e.g., "Van Gogh," "Picasso") concepts within a single image while preserving visual quality. Quantitatively, the attribution and bit accuracy evaluated on 100 clean and generated images is 89.14% and 95.47%, respectively.

6.4.2 Ablations

For all the ablation experiments, unless stated, we use a model trained for 10 concepts (see supp.).

Nearby Concepts and Clean Images.

CustomMark provides the flexibility to easily switch from the watermarked image generation to non-watermarked version, which we define as clean image generation. To do this, we use the non-perturbed original text tokens, while keeping the mapper network $\mathcal{M}_C$ and the fine-tuned

Figure 6.7 Attribution for multiple Concepts present in a single prompt with CustomMark.

Table 6.3 Ablation study for various style losses. [KEYS: Acc. Accuracy, Att. Attribution, Atte.: Attention].

Method	Bit Acc. (%)↑	Attribution Acc. (%) ↑	CLIP Score ↑	CSD Score ↑
CSD	98.6	88.15	0.65	0.73
CSD + L2 (latent)	99.12	90.94	0.70	0.67
CSD + L2 (image)	99.17	92.35	0.73	0.74
CSD + L2 + LDM atte.	**99.29**	**94.29**	**0.81**	**0.77**

attention weights of the model. An all-zero bit secret is used as an input to $\mathcal{M}_C$ and the secret decoder is expected to output the same for these clean images. We evaluate CustomMark's ability to generate clean images for 1) attributable concepts: that are fine-tuned with CustomMark and 2) nearby concepts: that are related to attributable concepts but not exactly the same. For example, if CustomMark can attribute paintings of Van Gogh, then paintings from other artists are considered nearby concepts. For this evaluation, we use three attributable artists (first three cols of Fig. 6.5) and seven random nearby artists (see supp.).

For attributable concepts, the model achieves high bit accuracy (96.13%) and attribution accuracy (85.45%) with an all-zeros bit secret, indicating effective attribution of clean concepts. For nearby concepts, it maintains strong bit accuracy (92.36%) and attribution accuracy (81.90%), showcasing the adaptability of CustomMark for practical applications, allowing selective watermarking for certain styles while not watermarking concepts which don't specifically request it. The

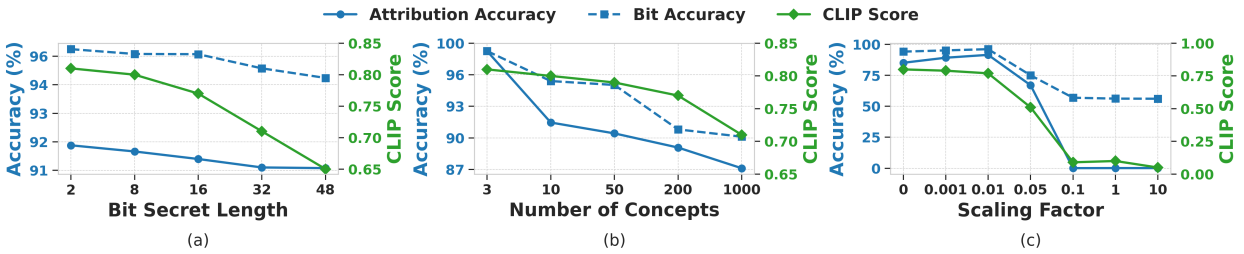


Figure 6.8 Ablation study for varying different parameters of CustomMark. We show the performance variation by varying bit secret length, number of concepts, and scaling factor.

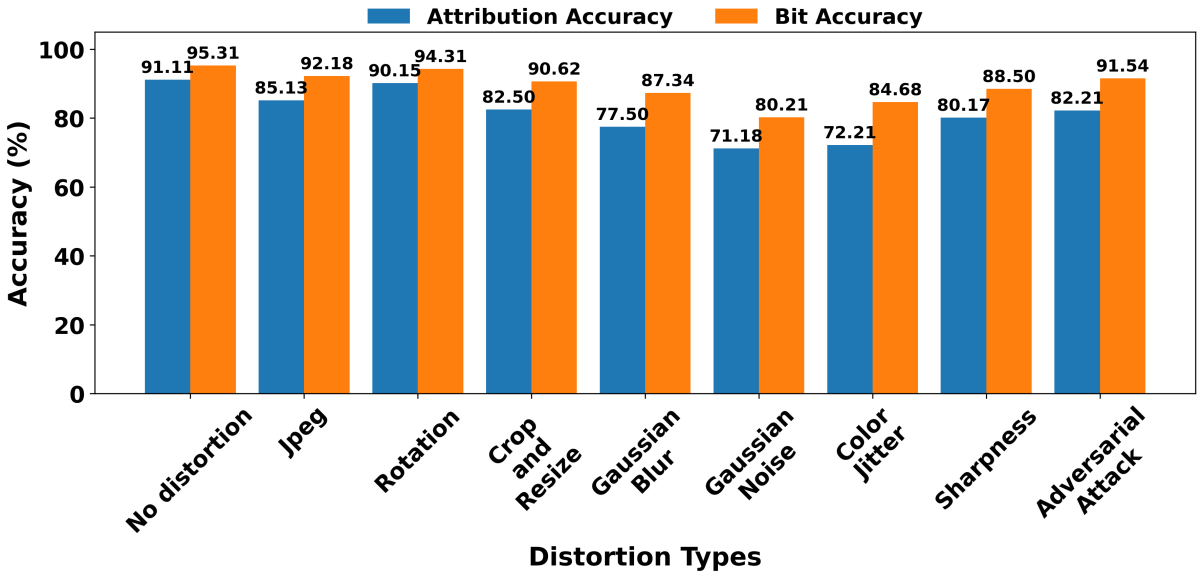


Figure 6.9 Robustness evaluation of decoder by applying distortion to generated images.

generation quality with CustomMark is comparable to the pretrained LDM, with an FID score of 14.51 between original and clean images.

Style Loss. Tab. 6.3 presents an ablation study on different style loss combinations and their impact on bit accuracy, attribution accuracy, and qualitative metrics. The baseline using only CSD performs well, but adding L2 loss in LDM's latent space improves accuracy, with a slight drop in the CSD score. Further applying L2 loss in image space enhances overall performance, boosting attribution accuracy, CLIP, and CSD scores. The best results are achieved by CustomMark, which combines CSD, L2 loss, and attention layer training, yielding the highest gains across all metrics and validating our design choice.

Robustness. Fig. 6.9 demonstrates CustomMark's robustness against various post-processing

distortions, including JPEG compression, rotation, cropping, resizing, Gaussian blur, noise, color jitter, and sharpness (see supp.). CustomMark maintains high attribution and bit accuracy, with minimal impact from common distortions like JPEG compression and rotation, while stronger distortions (e.g., Gaussian blur, noise) cause slight accuracy drops. Against adversarial attacks [379], it retains 82.21% attribution accuracy, only slightly lower than the original 91.11%. These results highlight CustomMark's resilience in real-world scenarios.

Bit Secret Length Fig. 6.8(a) analyzes the effect of bit secret length on bit accuracy, attribution accuracy, and CLIP score. As the secret length increases, both accuracy metrics decline, suggesting that longer secrets are harder for the decoder to recover, impacting attribution performance. Additionally, the CLIP score drops, indicating stylistic deviations. This trade-off suggests that a moderate bit length, such as 16, balances attribution accuracy and stylistic fidelity.

Number of Concepts. Fig. 6.8(b) examines how the number of unique artist concepts affects attribution and stylistic fidelity. As concepts increase, bit and attribution accuracy decline, likely due to the growing challenge of distinguishing among them. Similarly, the CLIP score drops, suggesting that maintaining stylistic consistency becomes harder with a broader range of styles in watermarked images.

Scaling Factor. Fig. 6.8(c) shows the impact of the scaling factor in Eq. (6.6) on attribution and stylistic similarity. Increasing the scaling factor sharply reduces both bit and attribution accuracy, likely due to overpowering the sampled Gaussian noise. Conversely, decreasing it too much causes the LDM to diverge, generating noise images, as reflected in the declining CLIP score. This underscores the need for a low scaling factor to balance attribution accuracy and stylistic preservation, leading us to select 0.01 for our experiments.

6.5 Conclusion

We propose CustomMark, an efficient and flexible technique for enabling concept attribution in pre-trained text-to-image LDMs. Addressing the growing demand for ethical content generation in GenAI models, CustomMark provides a customization-based approach to embed concept-specific watermarks, allowing artists to request attribution for their work. Unlike previous

methods, CustomMark allows selective attribution without requiring all concepts to be predefined before training, and entire watermarking of the training data. It supports sequential learning to add new concepts in a online-way. We demonstrate that CustomMark can handle hundreds of artist styles and diverse ImageNet classes while maintaining image quality and ensuring robust attribution. By fine-tuning the model for new concepts with minimal computational overhead, CustomMark streamlines the attribution process, helping bridge the gap between GenAI developers and the creative community, and so promoting responsible use of GenAI in content creation.

CHAPTER 7

PIVOT: PROACTIVE VIDEO TEMPLATES FOR ENHANCING VIDEO TASK PERFORMANCE

In this paper, we introduce PiVoT, a video-based proactive wrapper that enhances Action Recognition (AR) and Spatio-Temporal Action Detection (STAD) systems. By leveraging a proactive template-enhanced Low-Rank Adaptation (LoRA) paradigm, PiVoT integrates seamlessly with detectors while maintaining an efficient training approach. A 3D U-Net generates action-specific templates, capturing temporal dynamics through shadow-like artifacts that help detectors better identify motion cues and refine frame distribution. Fine-tuning only the LoRA layers within the CNN backbone or transformer attention layers ensures minimal computational overhead while improving detection accuracy. Applied to TSN, TSM, MViTv2, and SlowFast across datasets like Kinetics-400, Something-Something-v2, and AVA2.1, PiVoT consistently boosts performance, demonstrating its adaptability and scalability for video-based detection tasks. Models and code will be released upon publication[1].

7.1 Introduction

Video-based tasks in computer vision, such as Action Recognition (AR) and Spatio-Temporal Action Detection (STAD), play a crucial role in enabling machines to understand dynamic scenes and human behavior. AR focuses on recognizing the action occurring in a video by analyzing temporal sequences and identifying the action category. AR methods have evolved from traditional hand-crafted features in RGB videos [21, 175, 318, 319] to sophisticated deep learning techniques, such as two-stream Convolutional Neural Networks (CNNs) [280, 324, 323, 322, 334, 243, 86], Recurrent Neural Networks (RNNs) [290, 76, 307, 123, 101, 186, 12, 290, 217], 3D CNNs [86, 35, 17, 119, 302, 301], and Transformer-based models [3, 349, 347, 310, 79], which capture spatial and temporal features. STAD, on the other hand, aims to both localize and recognize actions within videos by assigning bounding boxes to each instance and classifying the type of action. STAD include two-stage methods, separating bounding box detection and the classification of

[1] *Vishal Asnani*, Xiaoming Liu, and Shruti Agarwal. "PiVoT: Proactive Video Templates for Enhancing Video Task Performance." In review, 2025.

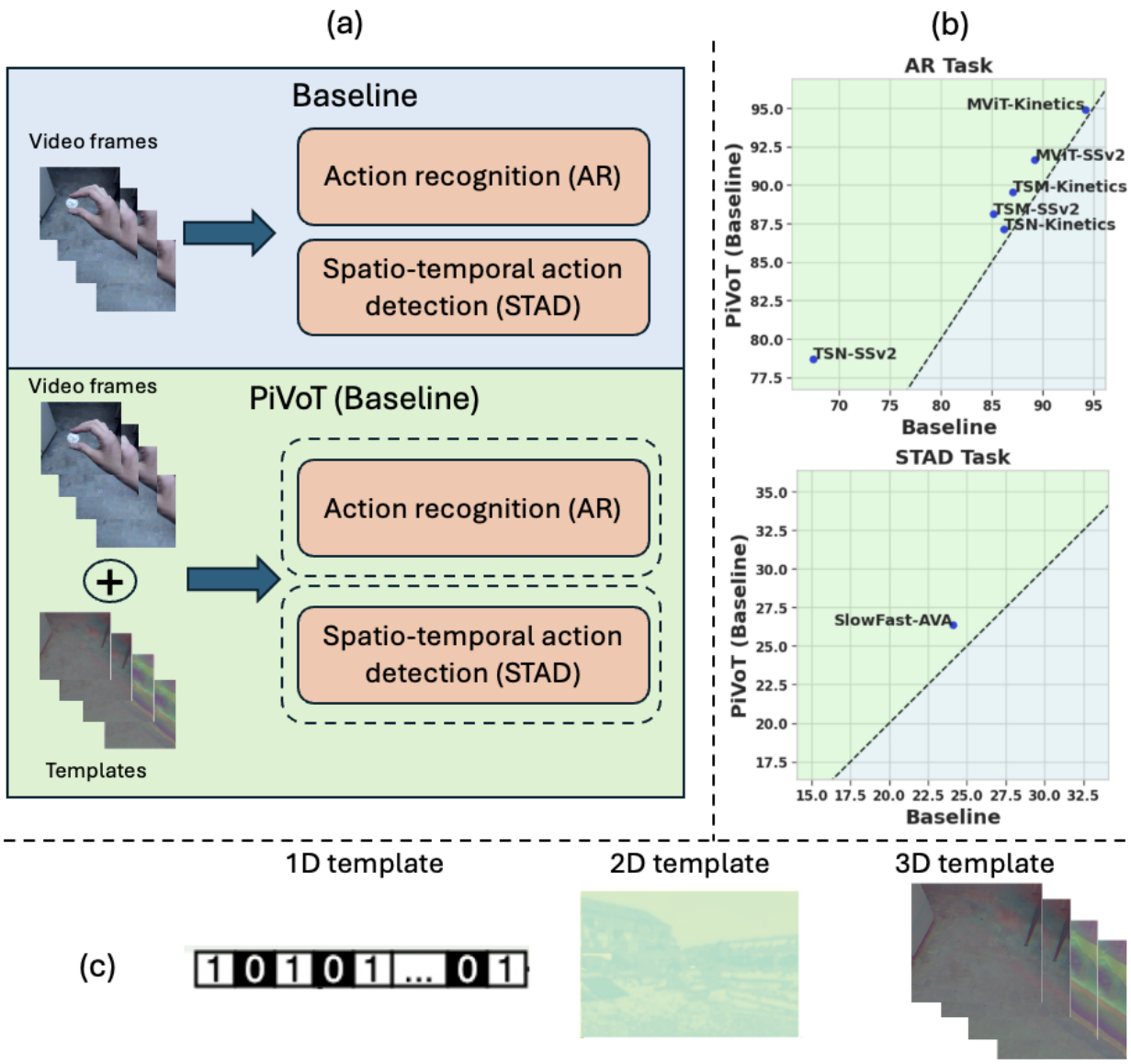


Figure 7.1 **PiVoT as a wrapper.** (a) We propose PiVoT which wraps around different video-based baseline detectors to improve the performance. (b) We show the effectiveness of incorporating PiVoT on various detectors for two different video-based tasks, across three different datasets. PiVoT is able to improve the performance for each detector. The plotted points represent performance improvements after applying PiVoT, with all points in the green region indicating enhanced performance compared to the baseline. (c) PiVoT uses 3D templates unlike prior proactive works which use 1D/2D templates.

actions [271, 112, 242], and query-based one-stage methods [167, 376], which integrate these steps into a unified process.

Many of these techniques use various type of modules and augmentation strategies [110, 352, 59, 344, 339, 142] resulting in performance gains by increasing diversity and model robustness.

Further for performance gain, a growing segment of deep learning involves the use of proactive learning schemes [9] which have demonstrated performance improvements across various computer vision tasks, including vision model and LLM defense [298, 342], privacy solutions [343, 299], attribution [4], manipulation detection [6], localization [7], and 2D object detection [5], among others. While these methods share similarities with traditional augmentation techniques, their distinct feature lies in learning an additional signal known as *templates*. Asnani *et al*. [5] propose a proactive wrapper for 2D object detectors which generate templates, when applied under certain conditions, can significantly enhance the performance of networks.

Building on augmentation and proactive learning insights, we introduce PiVoT, a video-based proactive wrapper that enhances video detectors (Fig. 7.1(a)). Developing such a wrapper poses key challenges: it must be plug-and-play, parameter-efficient, and compatible across diverse datasets while preserving temporal dynamics. Additionally, it should integrate seamlessly with detectors without extensive modifications or performance trade-offs. Overcoming these challenges makes PiVoT a scalable solution for improving video-based detection systems.

To address these challenges, PiVoT employs a proactive template-enhanced Low-Rank Adaptation (LoRA) training paradigm, using a 3D U-Net to generate action-specific 3D templates (Fig. 7.1(c)). These templates are applied to video frames before detector processing. The detector, initialized with pretrained weights, integrates LoRA layers into either the CNN backbone [45] or transformer attention layers [136], enabling efficient task adaptation. We show that the estimated templates capture temporal information, evident in shadow-like artifacts that provide motion cues, enhancing AR and STAD detectors' ability to interpret dynamic movements. By incorporating this temporal context, the templates refine feature extraction, improving model performance. Training is limited to the 3D U-Net and LoRA layers, keeping the rest of the detector fixed to minimize computational overhead. We demonstrate PiVoT's effectiveness on AR and STAD tasks, showing consistent performance gains across various detectors and datasets (Fig. 7.1(b)). Our key contributions are summarized below:

- **Proactive Wrapper for Enhanced Video-Based Detection**: We introduce PiVoT, a novel

video-based proactive wrapper designed to enhance the performance of multiple video-based detectors, including those for AR and STAD. PiVoT functions as a wrapper, effectively integrating with existing detectors and augmenting their capabilities without requiring significant architectural changes.

- **Template-Enhanced LoRA Training Paradigm**: PiVoT incorporates a 3D U-Net for generating action-specific templates, combined with a LoRA framework. This approach allows for targeted fine-tuning of specific components within the detector, such as the LoRA layers in the CNN backbone or transformer attention layers, resulting in improved performance with minimal training.

- **A plug and play architecture module**: Experiments demonstrate that PiVoT can be used as a plug-and-play architecture module resulting in consistent gains in performance across various datasets (e.g., Kinetics-400, Something-Something-v2, and AVA2.1) and detectors (TSN, TSM, MViTv2 and SlowFast), validating the efficacy of our approach.

7.2 Related Works

Proactive Schemes Proactive methods enhance various tasks by embedding signals or perturbations into input images, aiding in deepfake tagging [325], detection of manipulated content [6], and localization [7]. Asnani *et al*. [5] improve object detection with such techniques while approaches by Yeh *et al*. [356] and Ruiz *et al*. [267] modify training data to disrupt generative model outputs while [270] introduce fixed signals for dataset attribution. A survey by Asnani *et al*. [9] discusses perturbation methods, applications in vision model and LLM defense [298, 342, 221], and privacy solutions [343, 299, 88]. Proactive methods also boost generative AI [285, 163] and address challenges in 3D domains [134, 151]. Different from the above strategies, we propose to apply the proactive paradigm in improving the performance of video-based tasks.

Action Recognition In recent years, significant advancements have been made in AR through the integration of various modalities and deep learning techniques. Early works focus on hand-crafted features using RGB data, such as the Temporal Template method by Bobick *et al*. [21] and STIP by

Laptev *et al*. [175]. The advent of deep learning sees the rise of two-stream CNNs like Simonyan *et al*. [280] and RNNs such as LRCN by Donahue *et al*. [76], which improves spatiotemporal feature extraction. Temporal Segment Networks (TSN) by Wang *et al*. [324] and Temporal Shift Module (TSM) by Lin *et al*. [187] further enhance temporal modeling capabilities. More recently, 3D CNNs and Transformer-based methods, including I3D by Carreira *et al*. [35] and ViViT by Arnab *et al*. [3], have achieved state-of-the-art (SoTA) results. Multimodal fusion techniques, such as the three-stream CNN for RGB and audio by Wang *et al*. [315] and the combination of depth and inertial sensors by Dawar *et al*. [68], are also explored to improve accuracy. MViTv2 [182] is a unified architecture for image and video classification, as well as object detection. In egocentric action recognition (EAR), deep learning frameworks like Ego-ConvNet by Singh *et al*. [283] and the Mutual Context Network by Huang *et al*. [140] address the unique challenges of first-person video data. We propose a wrapper which plugs with a pre-existing AR detector, resulting in performance enhancement.

Spatio-temporal Action Detection (STAD) STAD has seen significant advancements in recent years, driven by the rapid development of deep learning techniques. Early works in STAD, such as those by Gkioxari *et al*. [105] and Weinzaepfel *et al*. [336], lay the foundation by introducing methods to link frame-level detections into action tubes. The introduction of region proposal networks (RPN) by Saha *et al*. [271] and the use of two-stream architectures to incorporate both appearance and motion cues [242] further improve detection accuracy. Recent approaches have leveraged 3D convolutional neural networks (3D CNNs) to capture motion information across multiple frames, as demonstrated by Gu *et al*. [112] and Girdhar *et al*. [99]. SlowFast [85] introduces two pathways to capture spatial and temporal dynamics. The integration of visual relation modeling, as seen in works by Sun *et al*. [289] and Girdhar *et al*. [100], enhances the understanding of interactions between actors and objects, leading to more accurate action detection. Additionally, the development of efficient and real-time models, such as YOWO [167] and WOO [44], has addressed the computational challenges associated with STAD. The use of transformer-based frameworks, like TubeR [376] and HIT [84], also shows promising results, highlighting the potential of transformers

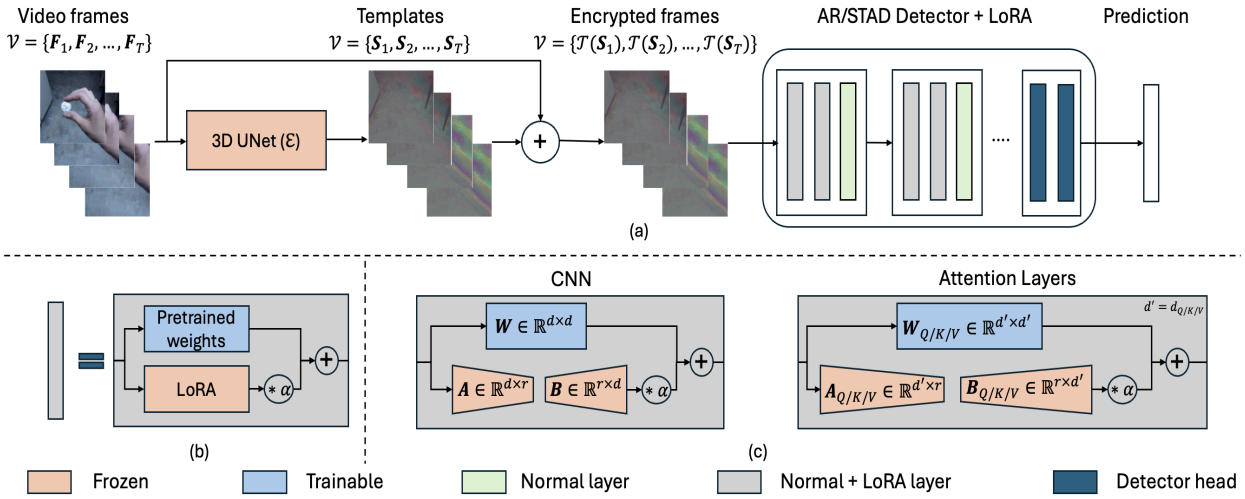


Figure 7.2 **Overview of PiVoT**. This figure illustrates the PiVoT framework for video-based tasks. (a) Video frames are processed through a 3D U-Net model to generate templates, which are then perturbed by adding them to the original frames. The perturbed frames are passed to a detector enhanced with LoRA layers, producing final predictions. (b) Detailed visualization of LoRA integration to the pretrained weights. (c) LoRA applied to CNN layers by injecting low-rank adaptation matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ into pretrained weights $\boldsymbol{W}$, and LoRA applied to attention layers, modifying the query/key/value weights $\boldsymbol{W}_{Q/K/V}$ with additional low-rank matrices $\boldsymbol{A}_{Q/K/V}$ and $\boldsymbol{B}_{Q/K/V}$. The framework is trained in an end-to-end manner using the respective baseline losses. Best viewed in color.

in this domain. Unlike these works, PiVoT uses proactive learning to improve STAD detector performance.

7.3 Method

7.3.1 Preliminary

In this paper we study two video-based tasks: Action Recognition (AR), and Spatio-Temporal Action Detection (STAD).

AR aims to identify human actions from a video sequence. Let $\mathcal{V} = \{\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_T\}$ represent a video sequence, where each frame $\boldsymbol{F}_t \in \mathbb{R}^{H \times W \times 3}$ corresponds to an image of height H, width W, and three color channels (RGB). The sequence consists of T frames. The goal is to classify the video $\mathcal{V}$ into one of the possible action classes $y \in \mathcal{A}$, where $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$ represents the set of K possible action labels.

The task can be formulated as learning a function f, parameterized by θ, that maps the video

sequence $\mathcal{V}$ to the predicted action label $\hat{y}$:

$$f_\theta : \mathcal{V} \rightarrow \hat{y}. \tag{7.1}$$

Thus, the predicted action label $\hat{y}$ is given by:

$$\hat{y} = \underset{y \in \mathcal{A}}{\operatorname{argmax}}\, p(y|\mathcal{V}; \theta), \tag{7.2}$$

where $p(y|\mathcal{V}; \theta)$ is the probability that the action label y corresponds to the video sequence $\mathcal{V}$, given the model parameters θ.

STAD aims to identify action types in a video and localize them across frames. For the video $\mathcal{V}$, the task is to detect the action label $y \in \mathcal{A} = \{a_1, a_2, \ldots, a_K\}$ and bounding box $\boldsymbol{B}_t$ for each frame.

Our goal is to learn a function f with parameters θ that maps the video sequence $\mathcal{V}$ to detected actions and spatial locations:

$$f_\theta : \mathcal{V} \rightarrow \{(y, \boldsymbol{B}_t) \mid \boldsymbol{F}_t, t = 1, \ldots, T\}, \tag{7.3}$$

where y is the action label and $\boldsymbol{B}_t \in \mathbb{R}^4$ represents bounding box coordinates for frame $\boldsymbol{F}_t$. Thus, the predicted action detection across frames is given by:

$$(y^*, \{\boldsymbol{B}_t^*\}_{t=1}^T) = \underset{y \in \mathcal{A}}{\operatorname{argmax}} \prod_{t=1}^{T} p(y, \boldsymbol{B}_t|\mathcal{V}; \theta), \tag{7.4}$$

where $p(y, \boldsymbol{B}_t|\mathcal{V}; \theta)$ denotes the likelihood that y and $\boldsymbol{B}_t$ describe the observed action in each frame given θ.

7.3.2 PiVoT

7.3.2.1 Overview

For video-based tasks, capturing spatial and temporal dynamics is crucial for accurately identifying actions in video sequences. As shown in Fig. 7.2(a), we propose a template-enhanced Low-Rank Adaptation (LoRA) approach, which utilizes a 3D U-Net to generate action-specific templates that are added to video frames before they are processed by a detector. The detector, initialized with pretrained weights, is modified by adding LoRA layers to specific components—either in the CNN backbone or transformer attention layers—allowing for efficient adaptation to the task.

This approach enables us to train only the 3D U-Net and the LoRA layers, while the rest of the detector remains fixed. Next we'll discuss each component in more detail.

7.3.2.2 Template Generation

Inspired by previous works [7, 6], proactive schemes are applied to enhance the performance of computer vision tasks by perturbing input images using a template. In the context of video-based tasks, we propose to use proactive schemes to enhance the performance of the baseline methods by perturbing each frame of a video sequence using a template.

Proactive approaches offer the flexibility of using fixed or learnable templates. The template can either be universal [7, 6] across the entire dataset, or data-dependent template [325, 5]. In our scenario, video-based tasks poses unique challenges due to the substantial variability in video-content across different contexts, including differences in motion patterns, temporal dynamics, and viewpoint changes. These fluctuations introduce a level of complexity that may surpass the representational capacity of a fixed template set, potentially limiting the performance.

To address this limitation, we propose an approach that dynamically generates a unique video template for each video sequence. This is achieved using an 3D U-Net based encoder network $\mathcal{E}$ designed to produce tailored templates based on the specific action and contextual cues within each sequence. In doing so, our method adapts more fluidly to the diverse and often nuanced visual patterns present across video datasets, ensuring enhanced representation and recognition of actions that vary significantly from one instance to another. Let each frame of the video be represented as $\boldsymbol{F}_t \in \mathbb{R}^{H\times W\times 3}$. For each frame, the model learns a frame-specific template $\boldsymbol{S}_t \in \mathbb{R}^{H\times W\times 3}$. In PiVoT, each frame $\boldsymbol{F}_t$ is perturbed using a transformation $\mathcal{T}$ that applies the template $\boldsymbol{S}_t$ to the frame. This is achieved through element-wise addition, resulting in the perturbed frame $\mathcal{T}(\boldsymbol{F}_t)$:

$$\mathcal{T}(\boldsymbol{F}_t) = \mathcal{T}(\boldsymbol{F}_t; \boldsymbol{S}_t) = \boldsymbol{F}_t + \boldsymbol{S}_t = \boldsymbol{F}_t + \mathcal{E}(\mathcal{V})_t. \tag{7.5}$$

Hence, the video $\mathcal{V} = \{\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_T\}$ is transformed to a perturbed video $\mathcal{T}(\mathcal{V})$, which is then used for video-based tasks. Therefore, our proactive wrapper changes the formulation defined

in Eq. (7.2) and Eq. (7.4) as follows:

$$\hat{y} = \underset{y \in \mathcal{A}}{\operatorname{argmax}}\, p(y|\mathcal{T}(\mathcal{V}); \theta), \tag{7.6}$$

$$(y^*, \{\boldsymbol{B}_t^*\}_{t=1}^T) = \underset{y \in \mathcal{A}}{\operatorname{argmax}} \prod_{t=1}^{T} p(y, \boldsymbol{B}_t|\mathcal{T}(\mathcal{V}); \theta). \tag{7.7}$$

7.3.2.3 Detector with LoRA

PiVoT enhances a detector using LoRA layers (Fig. 7.2(b)), which enable efficient fine-tuning by introducing adaptable components while keeping the core pretrained weights fixed. LoRA is applied to either the CNN backbone or the transformer attention layers, offering flexibility in adapting to action-specific features encoded by templates generated from $\mathcal{E}$. This section details the integration of LoRA into both architectures, illustrating its roles in improving detectors.

In models with CNN backbones, convolutional layers are key to extracting spatial features from video frames. Fine-tuning these layers requires updating a large number of parameters, which can be computationally expensive. LoRA addresses this challenge by introducing low-rank matrices into the convolutional layers [45]. As shown in Fig. 7.2(c), each convolutional layer contains a weight matrix $\boldsymbol{W}$ representing the filter kernels used to process input frames. LoRA modifies this weight matrix by decomposing it into two smaller, trainable matrices $\boldsymbol{A}$ and $\boldsymbol{B}$:

$$\boldsymbol{W}_{\text{LoRA}} = \boldsymbol{W} + \alpha \boldsymbol{A}\boldsymbol{B}. \tag{7.8}$$

Here, $\boldsymbol{W}$ represents the fixed pretrained weights, $\boldsymbol{A} \in \mathbb{R}^{d \times r}$ and $\boldsymbol{B} \in \mathbb{R}^{r \times d}$ are the low-rank matrices, with $r \ll d$, and α is a scaling factor that controls the influence of the adaptation. During training, only the low-rank matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are updated, significantly reducing the number of trainable parameters. This adaptation allows the convolutional layers to better respond to template-enhanced inputs from $\mathcal{E}$. The modified CNN backbone thus retains its pretrained strengths while adjusting its spatial feature extraction to the new patterns encoded by the templates, enabling efficient adaptation without the need for extensive retraining of the detector.

In contrast, transformer-based detectors leverage self-attention mechanisms to capture temporal dependencies across frames, making them well-suited for modeling complex action sequences. The

challenge, however, lies in fine-tuning these attention layers, which typically involves adjusting large query, key, and value matrices used in each attention head. LoRA offers a solution by introducing trainable low-rank matrices into these components (Fig. 7.2(c)), allowing for efficient adaptation while fixing the main parameters [136]. Specifically, LoRA modifies the query matrix $\boldsymbol{W}_Q$ as:

$$\boldsymbol{W}_{Q,\text{LoRA}} = \boldsymbol{W}_Q + \alpha \boldsymbol{A}_Q \boldsymbol{B}_Q, \tag{7.9}$$

where $\boldsymbol{W}_Q \in \mathbb{R}^{d_Q \times d_Q}$ is the original query weight matrix, $\boldsymbol{A}_Q \in \mathbb{R}^{d_Q \times r}$ and $\boldsymbol{B}_Q \in \mathbb{R}^{r \times d_Q}$ are the low-rank matrices specific to the query transformation. Similar adjustments are made to the key and value matrices, allowing the attention mechanism to adapt to action-specific dynamics encoded in the template-enhanced frames. By training only the low-rank matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, the model efficiently adapts to the template-enhanced inputs.

By focusing only on training the 3D U-Net and the LoRA layers, our approach allows the video detector to quickly adapt to the new information provided by template-enhanced inputs. This selective fine-tuning strategy results in a model that is both computationally efficient and capable of achieving high accuracy in recognizing actions across diverse video datasets.

7.4 Experiments

7.4.1 Implementation Details

We consider two video-based tasks for demonstrating the effectiveness of PiVoT, namely, AR and STAD.

Datasets For AR task, we conduct experiments on two datasets: Something-Something-v2 [111], and Kinetics-400 [160] dataset. For STAD task, we use AVA2.1 [112] dataset. Below are the details for each dataset.

1. **Something-Something v2**: A large-scale video dataset focused on fine-grained human-object interactions, containing over 220, 000 labeled video clips across 174 action classes.

2. **Kinetics 400**: A widely used video dataset with 400 action classes, featuring around 300, 000 high-quality clips sourced from YouTube to represent diverse human activities.

Table 7.1 Performance comparison on Something-Something-v2 dataset for TSN, TSM, and MViTv2 models, with and without PiVoT wrapper.

Method	**TSN** [324](%)↑		**TSM** [187](%)↑		**MViTv2** [182](%)↑	
	Top-1	**Top-5**	**Top-1**	**Top-5**	**Top-1**	**Top-5**
Reported	35.51	67.09	62.72	87.70	68.11	91.02
Reproduced	35.69	67.39	59.19	85.14	64.29	89.21
PiVoT	**51.37**	**78.71**	**63.41**	**88.11**	**68.81**	**91.63**

Table 7.2 Performance comparison on Kinetics-400 dataset for TSN, TSM, and MViTv2 models, with and without PiVoT wrapper.

Method	**TSN** [324](%)↑		**TSM** [187](%)↑		**MViTv2** [182](%)↑	
	Top-1	**Top-5**	**Top-1**	**Top-5**	**Top-1**	**Top-5**
Reported	**72.83**	**90.65**	**73.18**	**90.56**	81.11	94.73
Reproduced	67.19	86.21	69.14	87.03	79.12	94.21
PiVoT	69.61	87.13	71.31	89.56	**81.51**	**94.91**

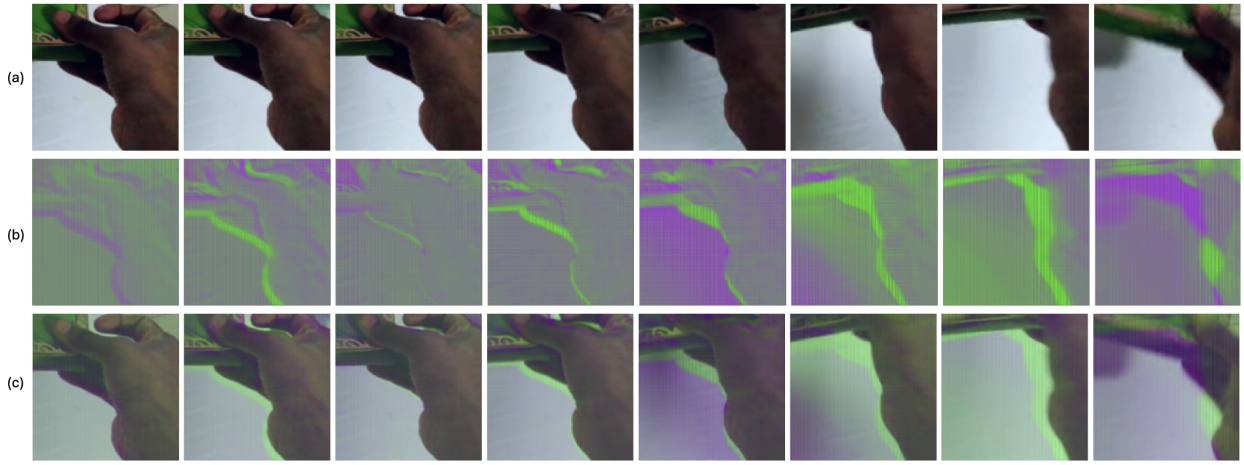


Figure 7.3 **Template visualization for TSN detector on Something-Something-v2.** We show the (a) input frames, (b) estimated template, and the (c) perturbed frames. The estimated template captures temporal information, as indicated by the shadow-like artifacts, which aid the action recognition (AR) detector in identifying motion cues more effectively. The template after being added changes the distribution of the input frames spatially to improve the performance accordingly.

3. **AVA 2.1**: An action detection dataset providing spatio-temporal annotations for actions in 430, 15-minute movie clips, designed for detailed analysis of person-centered activities in complex scenes.

Detectors and Evaluation For the AR task, we incorporate PiVoT into multiple detectors: TSN [324], TSM [187], and MViTv2 [182], and evaluate with and without the PiVoT wrapper. We report each detector's Top-1 and Top-5 accuracy as metrics. For the STAD task, we use

SlowFast [85] detector reporting mean Average Precision (mAP) (%) as the metric.

The performance for all the detectors is reported across three versions: the original reported values, the reproduced values (our baseline), and the results after incorporating the PiVoT wrapper. Starting with the pretrained models from our reproduced baselines, we observe that for some models, our reproduced performance differs slightly from the originally reported numbers. This variation could be due to differences in training setups or minor architectural adjustments in the models that were not disclosed in the original setup. Therefore, for a fair comparison, we apply our proposed wrapper on the reproduced pre-trained weights. We use the MMACTION2 toolbox [58] for each detector codebase and pretrained models. We select hyperparameter values as follows: LoRA rank $r = 4$ and scaling factor $\alpha = 0.01$. All the experiments are done on 8 A100 NVIDIA GPUs. We use the default parameters for each detector as reported in the respective papers by the authors (see details in supplementary).

7.4.2 Results

AR Task As shown in Tab. 7.1, PiVoT significantly improves both Top-1 and Top-5 accuracies across all detectors, particularly for TSN and TSM, where gains are more pronounced. This improvement is attributed to the addition of object-specific templates, which aid the models in capturing temporal consistency and semantic continuity in actions. LoRA layers enable PiVoT to adapt these templates without requiring substantial parameter updates, resulting in a parameter-efficient performance boost. We further show the input frames, templates, and the perturbed frames in Fig. 7.3. Perturbed frames are dominated by the templates, with their distribution modified to enhance AR accuracy. The estimated template aggregates temporal information, as evident from the shadow-like artifacts, which help the action recognition (AR) detector capture motion cues more effectively. This adaptation alters the distribution of input frames, enhancing model performance. The performance gains do not stem from merely increasing the number of trainable parameters but rather from the specially designed PiVoT wrapper, which effectively integrates proactive templates with LoRA-based adaptation. As demonstrated in our ablation studies, using only the 3D U-Net for template generation or solely incorporating LoRA layers does not consistently

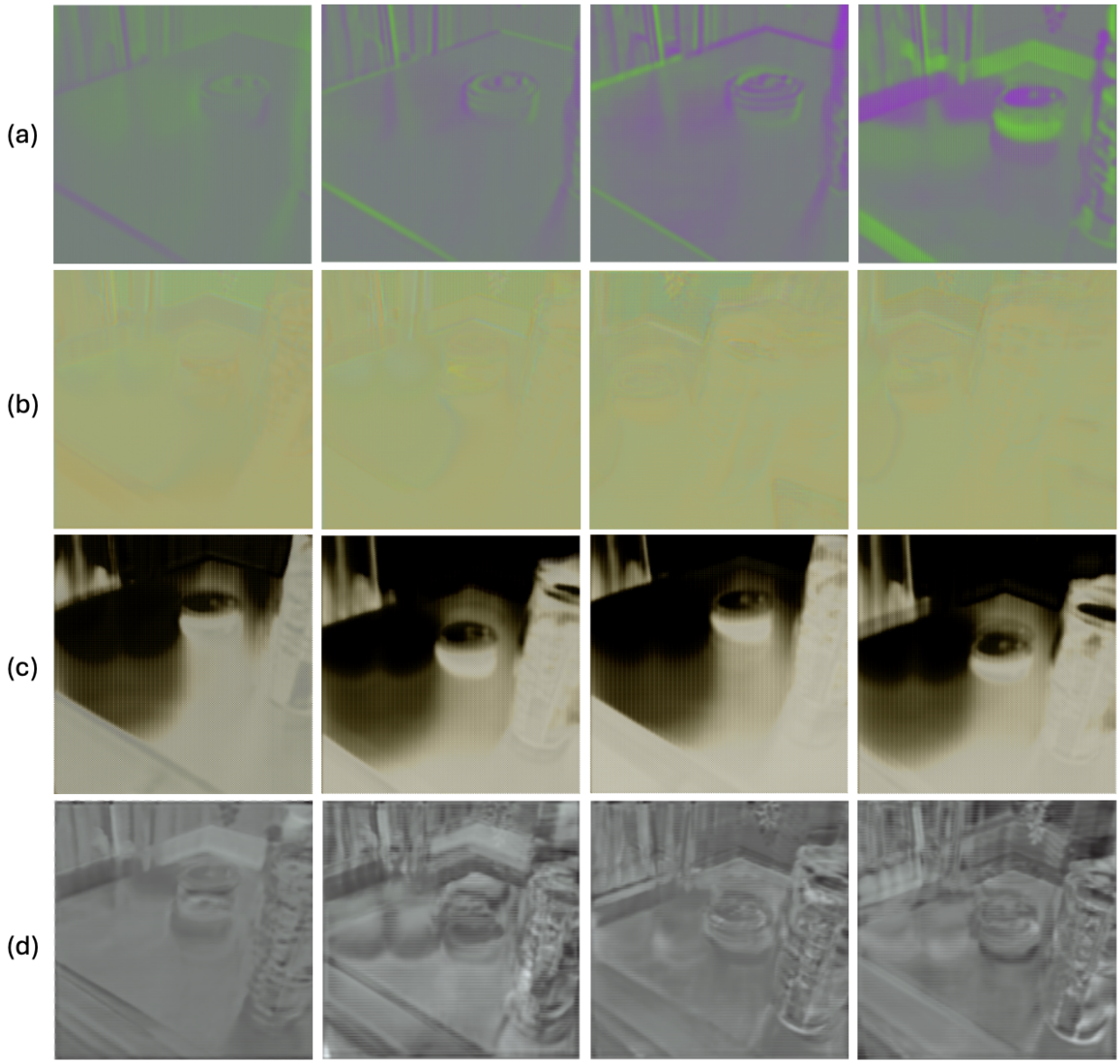


Figure 7.4 Template visualization for a video in something-something-v2 dataset across various detectors (a) TSN, (b) TSM, (c) MViTv2, and (d) SlowFast.

Table 7.3 Performance comparison on AVA2.1 dataset for SlowFast [85] detector, with and without PiVoT wrapper.

Metric	**Reported**	**Reproduced**	**PiVoT**
mAP (%)↑	24.32	24.11	**26.36**

guarantee performance improvements. Instead, PiVoT's unique combination of template-enhanced input transformations and parameter-efficient fine-tuning enables robust performance gains across different video-based tasks while maintaining computational efficiency.

Tab. 7.2 presents a similar comparison on the Kinetics-400 dataset for the three detectors. PiVoT

Table 7.4 **Training iterations ablation.** Analysis of training iterations on TSN and MViTv2 on Something-Something-v2, with extended iterations comparable to those after incorporating PiVoT. Inference times are shown before and after PiVoT application. Proactive training with PiVoT provides greater performance gains than merely increasing training iterations, with a slight increase in inference time. [KEYS: itr.: number of training iterations].

Method	**TSN** [324]			**MViTv2** [182]			**Itr.**
	Top-1 (%)↑	**Top-5** (%)↑	**Time** (ms)↓	**Top-1** (%)↑	**Top-5** (%)↑	**Time** (ms)↓	
Reproduced	35.69	67.39	19.23	64.29	89.21	13.21	1*X*
Reproduced	35.71	67.45		64.40	90.13		2*X*
PiVoT	**51.37**	**78.71**	20.68	**68.81**	**91.63**	15.28	2*X*

demonstrates clear improvements in both Top-1 and Top-5 accuracy. The gains, while slightly less than those observed on Something-Something-v2, indicate that PiVoT 's object-aware templates and LoRA-based adaptation are effective for enhancing action recognition on Kinetics-400. The Kinetics-400 dataset encompasses a wide variety of action types, featuring complex and dynamic object interactions that can be challenging for models to interpret consistently. This diversity makes performance enhancements particularly difficult, leading to less gains in performance.

Fig. 7.4 shows template variations across different video action detectors for a single video. Each row corresponds to a specific detector: (a) TSN, (b) TSM, (c) MViTv2, and (d) SlowFast. The distinct visual patterns reflect each detector's unique approach to capturing motion and spatial details to estimate the template. TSN emphasizes color variations, TSM appears more muted, MViTv2 focuses on high-contrast areas, and SlowFast highlights spatial outlines, demonstrating how each detector prioritizes different aspects of the video frames for video-based tasks.

STAD Task As shown in Tab. 7.3, the reported mAP for the SlowFast model is 24.32%, while the reproduced implementation achieves a similar mAP of 24.11%. Notably, integrating the PiVoT wrapper into the detector significantly improves performance, boosting the mAP to 26.36%. This improvement underscores the efficacy of PiVoT in enhancing model performance by incorporating proactive learning techniques. The gain in mAP reflects the ability of PiVoT to better capture temporal and spatial features within video frames, demonstrating its benefit in STAD task as well.

7.4.3 Ablations

The ablation study presented provides comprehensive insights into the impact of various components of PiVoT on the performance of video-based action detectors. This analysis evaluates the changes in performance when specific aspects of the PiVoT framework are modified.

Computational Overhead, Inference Time, and Additional Detector Training. Tab. 7.4 shows that PiVoT introduces a minor increase in inference time, or extra time delta, on both TSN and MViTv2 detectors. Specifically, for TSN, the inference time rises from 19.23ms to 20.68ms, representing a small delta of 1.45ms. For MViTv2, the inference time increases from 13.21ms to 15.28ms, a delta of 2.07ms. Despite this slight overhead, the performance gains in Top-1 and Top-5 accuracy are substantial, making the extra time a worthwhile trade-off. PiVoT enables more efficient training with greater performance improvements compared to merely increasing training iterations for the baseline detector. PiVoT's ability to deliver significant accuracy improvements with minimal increases in inference time underscores its efficacy as a proactive wrapper, enhancing detector performance without compromising computational efficiency.

Perturbation Process. First, we study the perturbation process by replacing the 3D-UNeT with a 3D-CNN and then switching the transformation operation from addition to multiplication. As shown in Tab. 7.5, replacing the 3D-UNeT with 3D-CNN results in decreased Top-1 and Top-5 accuracies for both TSN and MViTv2 detectors. For instance, TSN's Top-1 accuracy drops from 51.37% to 47.14% and MViTv2's Top-1 accuracy declines from 68.81% to 63.23%. This drop is attributed to the nature of the templates generated by these models. Fig. 7.5(c) shows that templates generated using a 3D-UNeT retain frame-dependent semantic content that aligns well with the underlying video frames, enhancing the detector's ability to capture temporal dynamics. Conversely, the templates generated by the 3D-CNN, shown in Fig. 7.5(b), lack this semantic coherence, which undermines the detector's ability to leverage temporal relationships, leading to less performance.

When the transformation operation is altered from addition to multiplication, the results in Tab. 7.5 indicate an even more pronounced drop in performance. For TSN, the Top-1 accuracy

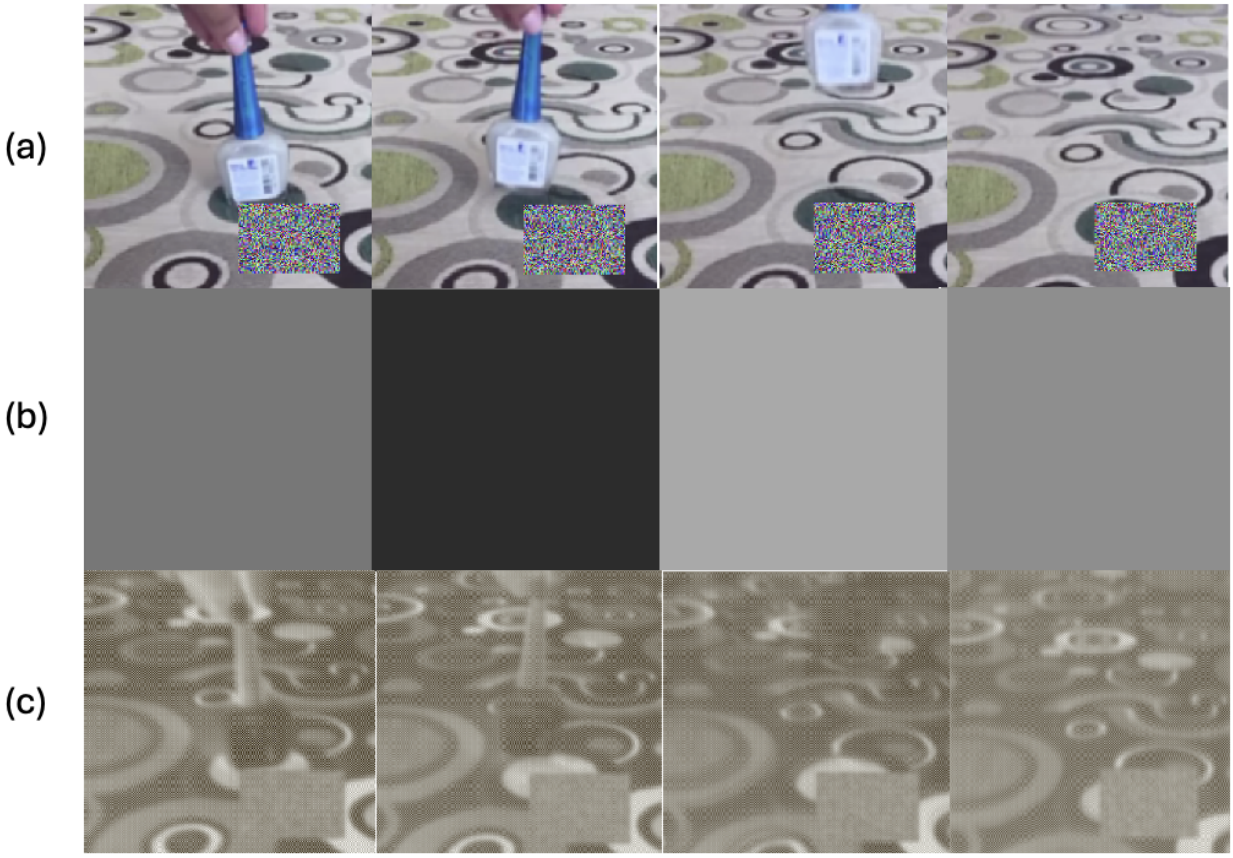


Figure 7.5 Template visualization for the (a) video frames estimated using (b) 3D-CNN, and (c) 3D-UNeT for MViTv2 detector. The templates estimated using 3D-CNN doesn't have semantic content, unlike estimation via 3D-UNeT which has frame-dependent semantic content useful for boosting the performance.

falls to 42.23%, and for MViTv2, it decreases to 60.12%. This suggests that addition, as a transformation, maintains the semantic integrity of the template and preserves crucial visual information, while multiplication may introduce extreme distortions to the input image disrupting the detector's learning process. This proves that the performance improvements achieved by PiVoT are not a result of simply increasing the number of trainable parameters but rather stem from its specialized design, which effectively combines proactive templates with LoRA-based adaptation. PiVoT 's unique combination of template-driven input transformations and parameter-efficient fine-tuning ensures significant performance gains across diverse video-based tasks while maintaining computational efficiency.

Detector Training. We ablate various ways of utilizing detector during our training. One

Table 7.5 Ablation study of various components of PiVoT wrapper.

Changed	**From→To**	**TSN** [324]		**MViTv2** [182]	
		Top-1	**Top-5**	**Top-1**	**Top-5**
PiVoT	-	51.37	78.71	68.81	91.63
Perturbation Process	3D-UNeT→3D-CNN	47.14	75.56	63.23	88.38
	Add→Multiplication	42.23	70.92	60.12	85.18
LoRA	Yes→No	31.96	60.40	41.82	72.68
	LoRA→ST-Adapter [235]	30.76	58.39	59.14	85.19
3D-UNeT	Yes→No	35.42	67.10	68.02	90.91
Detector	Frozen→Finetune	51.31	78.77	68.80	91.67
	Pretrain→Scratch	47.72	76.10	60.14	85.09
Frame selection	No→Yes	32.60	61.65	56.80	84.22

significant factor explored is the state of the detector during training. The results in Tab. 7.5 show that fine-tuning the detector while incorporating PiVoT yields similar performance as when using a frozen detector. For example, fine-tuning TSN results in a Top-1 accuracy of 51.31%, whereas using a frozen state has similar Top-1 accuracy *i.e.*51.37%. This suggests that using a frozen detector might be more suited for practical applications with just finetuning the LoRA layers with the templates while keeping the detector frozen, resulting in a fast and efficient training paradigm.

Furthermore, training the detector from scratch instead of using pretrained weights significantly impacts performance. The Top-1 accuracy for TSN decreases from 51.31% to 47.72%, and for MViTv2, it drops from 68.80% to 60.14%. This underscores the importance of leveraging pretrained weights to provide a strong starting point, allowing PiVoT to effectively enhance detection performance through incremental learning.

Frame Selection. We explored a **Frame Selection** strategy to enhance TSN and MViTv2 by prioritizing frames with high template norms, aiming to emphasize semantically rich content. Inspired by prior work [377, 133, 382] on selective frame sampling, we sampled four times more high-norm frames. However, this led to a performance drop, with TSN and MViTv2 Top-1 accuracy declining to 32.60% and 56.80%, respectively. This suggests that template norm-based selection overemphasizes specific segments, disrupts temporal continuity, and reduces frame diversity, ultimately hindering action recognition.

LoRA Ablation. We analyze the role of LoRA in PiVoT by first examining the impact of

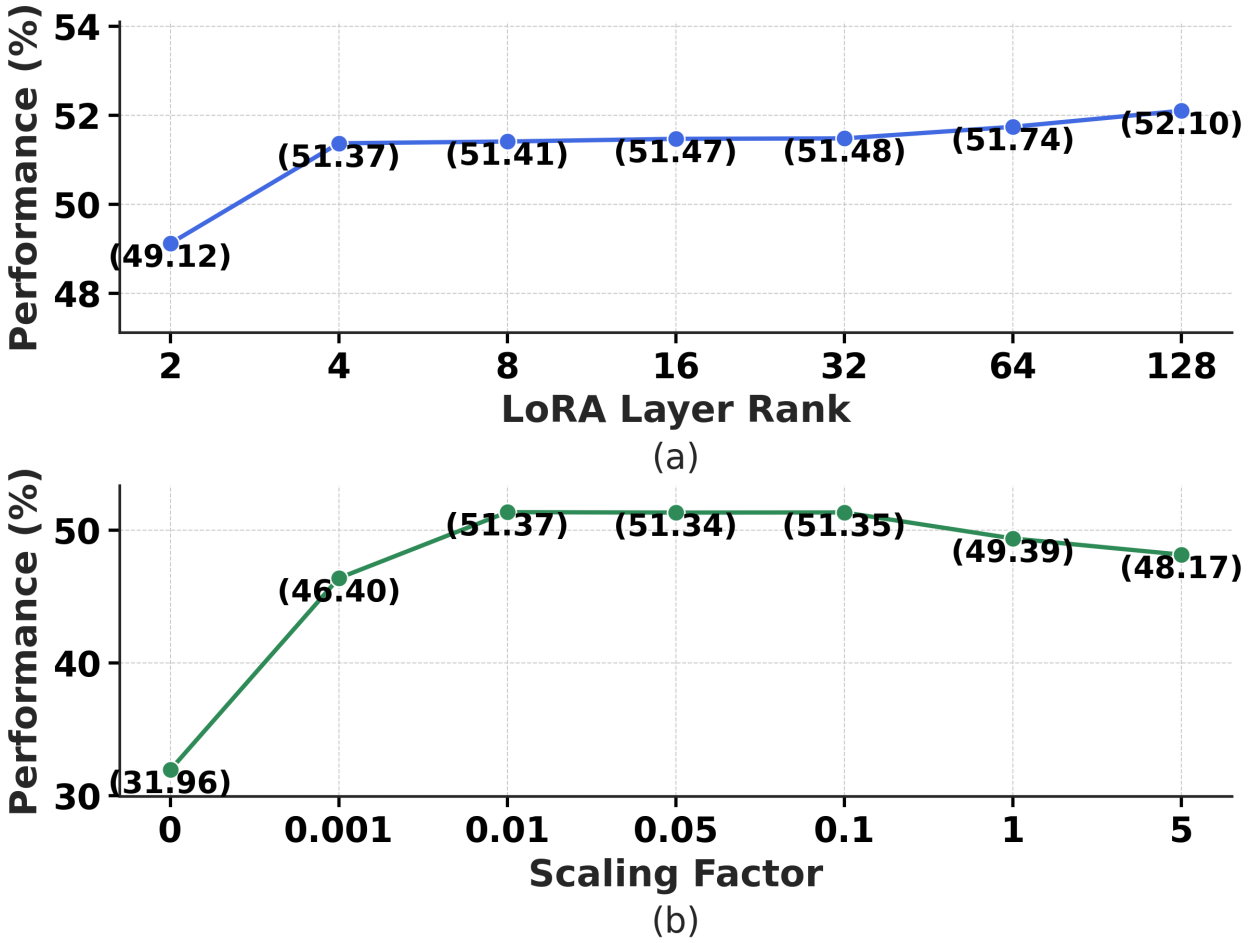


Figure 7.6 Ablation for (a) LoRA rank, and (b) scaling factor.

its removal. Eliminating LoRA significantly reduces performance, with TSN's Top-1 accuracy dropping from 51.37% to 31.96% and MViTv2's from 68.81% to 41.82%. This highlights LoRA's critical role in fine-tuning specific model components to enhance proactive template utility while avoiding extensive parameter updates.

Next, prior works have proposed parameter efficient modules to be integrated with the base network for transfer learning purposes. We compare LoRA with ST-Adapter [235] that trains an adapter at the network's end while keeping the base frozen. Replacing LoRA with ST-Adapter and fine-tuning TSN and MViTv2 (while keeping base detectors frozen) leads to inferior AR performance (Tab. 7.5). Unlike adapters, LoRA's lightweight design integrates directly into self-attention and convolution blocks, enabling better adaptation, lower compute cost, and faster convergence across both convolutional and transformer architectures.

In Fig. 7.6(a), we examine the effect of LoRA rank on TSN detector performance. Increasing the rank from 2 to 4 improves performance, indicating that additional capacity helps capture richer action recognition features. Beyond rank 4, performance stabilizes with minimal gains up to rank 128, suggesting diminishing returns. We select rank 4 for its near-optimal performance and lower parameter costs.

In Fig. 7.6(b), we analyze the impact of the scaling factor (α) from Eq. (7.9). At $\alpha = 0.001$, performance is low, indicating insufficient adaptation. Increasing to $\alpha = 0.01$ significantly improves performance, as LoRA effectively integrates template information while preserving pre-trained features. However, beyond $\alpha = 0.01$, performance degrades due to excessive modification of pre-trained weights. Thus, we use $\alpha = 0.01$, balancing adaptation and stability to optimize TSN detector performance.

7.5 Conclusion

We propose PiVoT, a proactive video-based wrapper that enhances action recognition (AR) and spatio-temporal action detection (STAD) systems. Leveraging proactive learning and augmentation, PiVoT integrates as a plug-and-play module with various detectors, improving performance. It employs a 3D U-Net to generate action-specific templates, which are added to input frames, and a LoRA-based training paradigm for efficient fine-tuning while preserving detector stability. The estimated templates capture temporal information, evident in shadow-like artifacts that aid AR detectors in identifying motion cues, refining frame distribution, and boosting performance. Experiments across multiple datasets and detectors validate PiVoT 's adaptability and scalability, demonstrating consistent improvements in video-based detection tasks.

CHAPTER 8

REVERSE ENGINEERING OF GENERATIVE MODELS: INFERRING MODEL HYPERPARAMETERS FROM GENERATED IMAGES

State-of-the-art (SOTA) Generative Models (GMs) can synthesize photo-realistic images that are hard for humans to distinguish from genuine photos. Identifying and understanding manipulated media are crucial to mitigate the social concerns on the potential misuse of GMs. We propose to perform reverse engineering of GMs to infer model hyperparameters from the images generated by these models. We define a novel problem, "model parsing", as estimating GM network architectures and training loss functions by examining their generated images – a task seemingly impossible for human beings. To tackle this problem, we propose a framework with two components: a Fingerprint Estimation Network (FEN), which estimates a GM fingerprint from a generated image by training with four constraints to encourage the fingerprint to have desired properties, and a Parsing Network (PN), which predicts network architecture and loss functions from the estimated fingerprints. To evaluate our approach, we collect a fake image dataset with 100K images generated by 116 different GMs. Extensive experiments show encouraging results in parsing the hyperparameters of the unseen models. Finally, our fingerprint estimation can be leveraged for deepfake detection and image attribution, as we show by reporting SOTA results on both the deepfake detection (Celeb-DF) and image attribution benchmarks[1].

8.1 Introduction

Image generation techniques have improved significantly in recent years, especially after the breakthrough of Generative Adversarial Networks (GANs) [108]. Many Generative Models (GMs), including both GAN and Variational Autoencoder (VAE) [158, 51, 156, 164, 29, 42, 74], can generate photo-realistic images that are hard for humans to distinguish from genuine photos. This photo-realism, however, raises increasing concerns for the potential misuse of these models, *e.g.*, by launching coordinated misinformation attack [314, 130]. As a result, deepfake detection [266,

[1] *Vishal Asnani*, Xi Yin, Tal Hassner, and Xiaoming Liu. "Reverse engineering of generative models: Inferring model hyperparameters from generated images." IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023.

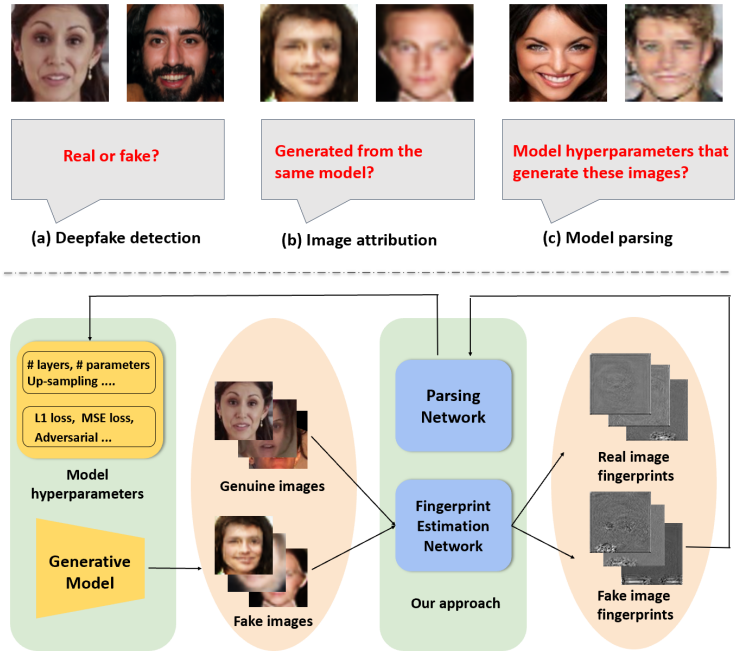


Figure 8.1 Top: Three increasingly difficult tasks: (a) *deepfake detection* classifies an image as genuine or fake; (b) *image attribution* predicts which of a closed set of GMs generated a fake image; and (c) *model parsing*, proposed here, infers hyperparameters of the GM used to generate an image, for those models unseen during training. Bottom: We present a framework for model parsing, which can also be applied to simpler tasks of deepfake detection and image attribution.

213, 114, 210, 66, 229] has recently attracted growing attention. Going beyond the binary genuine *vs*.fake classification as in deepfake detection, Yu *et al*. [365] proposed source model classification given a generated image. This *image attribution* problem assumes a *closed set* of GMs, used in both training and testing.

It is desirable to generalize image attribution to open-set recognition, *i.e.*, classify an image generated by GMs which were *not* seen during training. However, one may wonder what else we can do beyond recognizing a GM as an *unseen* or *new* model. Can we know more about how this new GM was designed? How its architecture differs from known GMs in the training set? Answering these questions is valuable when we, as defenders, strive to understand the source of images generated by malicious attackers or identify coordinated misinformation attacks which use the same GM. We view this as the grand challenge of reverse engineering of GMs.

While image attribution of GMs is both exciting and challenging, our work aims to take one step further with the following observation. When different GMs are designed, they mainly differ in

Table 8.1 Comparison of our approach with prior works on reverse engineering of models, fingerprint estimation and deepfake detection. We compare on the basis of input and output of methods, whether the testing is done on multiple unseen GMs and whether the testing is done on multiple datasets. [KEYS: R.E.: reverse engineering, I.A.: image attribution, D.D.: deepfake detection, Fing. est.: fingerprint estimation, mul.: multiple, un.: unknown, N.A.: network architecture, L.F.: Loss function, para.: parameters, sup.: supervised, unsup.: unsupervised].

Method (Year)	Purpose	Input	Output	Fing. est.	Test on mul. GMs	Test on un. GMs	Test on mul. data
[300] (2016)	R.E.	Attack on models	Training data	✖	✖	✖	✖
[233] (2018)	R.E.	Input-output images	N.A. para.	✖	✖	✖	✖
[137] (2018)	R.E.	Memory access patterns	Model weights	✖	✖	✖	✖
[15] (2018)	R.E.	Electromagnetic emanations	N.A. para.	✖	✖	✖	✖
[209] (2019)	I.A.	Image	✖	Sup.	✔	✖	✔
[365] (2019)	I.A.	Image	✖	Sup.	✔	✔	✔
[331] (2020)	I.A.	Image	✖	Sup.	✔	✖	✔
[372] (2019)	I.A.	Image	✖	Sup.	✔	✖	✔
[266] (2019)	D.D.	Image	✖	✖	✖	✖	✔
[114] (2020)	D.D.	Image	✖	✖	✖	✖	✔
[213] (2019)	D.D.	Image	✖	✖	✖	✖	✔
[210] (2019)	D.D.	Image	✖	✖	✖	✖	✔
[66] (2020)	D.D.	Image	✖	✖	✖	✖	✔
[229] (2020)	D.D.	Image	✖	✖	✖	✖	✔
[211] (2020)	D.D.	Image	✖	✖	✖	✖	✔
[192] (2021)	D.D.	Image	✖	✖	✖	✖	✔
Ours (2022)	R.E., I.A.,D.D.	Image	N.A. & L.F. para.	Unsup.	✔	✔	✔

their model hyperparameters, including the network architectures (*e.g.*, the number of layers/blocks, the type of normalization) and training loss functions. If we could map the generated images to the embedding space of the model hyperparameters used to generate them, there is a potential to tackle a new problem we termed as *model parsing*, *i.e.*, estimating hyperparameters of an *unseen* GM from only its generated image (Fig. 8.1). Reverse engineering machine learning models has been done before by relying on a model's input and output [300, 233], or accessing the hardware usage during inference [137, 15]. To the best of our knowledge, however, reverse engineering has not been explored for GMs, especially with only generated images as input.

There are many publicly available GMs that generate images of diverse contents, including faces, digits, and generic scenes. To improve the generalization of model parsing, we collect a large-scale fake image dataset with various contents so that our framework is not specific to a particular content. It consists of images generated from 116 CNN-based GMs, including 81 GANs, 13 VAEs, 6 Adversarial Attack models (AAs), 11 Auto-Regressive models (ARs) and 5 Normalizing Flow models (NFs). While GANs or VAEs generate an image by feeding a genuine image or latent code to the network, AAs modify a genuine image based on its objectives via back-propagation. ARs generate each pixel of a fake image sequentially, and NFs generate images via a flow-based

function. Despite such differences, we call all these models as GMs for simplicity. For each GM, our dataset includes $1,000$ generated images. We use each model's hyperparameters, including network architecture parameters and training loss types, as the ground-truth for model parsing training. We propose a framework to peek inside the black boxes of these GMs by estimating their hyperparameters from the generated images. Unlike the closed-set setting in [365], we venture into quantifying the generalization ability of our method in parsing *unseen* GMs.

Our framework consists of two components (Fig. 8.1, bottom). A *Fingerprint Estimation Network* (FEN) infers the subtle yet unique patterns left by GMs on their generated images. Image fingerprint was first applied to images captured by camera sensors [203, 106, 174, 92, 308, 202, 41] and then extended to GMs [209, 365]. We estimate fingerprints using different constraints which are based on the general properties of fingerprint, including the fingerprint magnitude, repetitive nature, frequency range and symmetrical frequency response. Different loss functions are defined to apply these constraints so that the estimated fingerprints manifest these desired properties. These constraints enable us to estimate fingerprints of GMs without ground truth.

The estimated fingerprints are discriminative and can serve as the cornerstone for subsequent tasks. The second part of our framework is a *Parsing Network* (PN), which takes the fingerprint as input and predicts the model hyperparameters. We consider parameters representing network architectures and loss function types. For the former, we form 15 parameters and categorize them into discrete and continuous types. For the latter, we form a 10-dimensional vector where each parameter represents the usage of a particular loss function type. Classification is used for estimating discrete parameters such as the normalization type, and regression is used for continuous parameters such as the number of layers. To leverage the similarity between different GMs, we group the GMs into several clusters based on their ground-truth hyperparameters. The mean and deviation are calculated for each GM. We use two different parsers: cluster parser and instance parser to predict the mean and deviation of these parameters, which are then combined as the final predictions.

Among the 116 GMs in our collected dataset, there are 47 models for face generation and 69

for non-face image generation. We partition all GMs into two categories: face *vs*.non-face. We carefully curate four evaluation sets for face and non-face categories respectively, where every set well represents the GM population. Cross-validation is used in our experiments. In addition to model parsing, our FEN can be used for deepfake detection and image attribution. For both tasks, we add a shallow network that inputs the estimated fingerprint and performs binary (deepfake detection) or multi-class classification (image attribution). Although our FEN is not tailored for these tasks, we still achieve state-of-the-art (SOTA) performance, indicating the superior generalization ability of our fingerprint estimation. Finally, in coordinated misinformation attack, attackers may use the same GM to generate multiple fake images. To detect such attacks, we also define a new task to evaluate how well our model parsing results can be used to determine if two fake images are generated from the same GM.

In summary, this paper makes the following contributions.

- We are the first to go beyond model classification by formulating a novel problem of model parsing for GMs.
- We propose a novel framework with fingerprint estimation and clustering of GMs to predict the network architecture and loss functions, given a single generated image.
- We assemble a dataset of generated images from 116 GMs, including ground-truth labels on the network architectures and loss function types.
- We show promising results for model parsing and our fingerprint estimation generalizes well to deepfake detection on the Celeb-DF benchmark [184] and image attribution [365], in both cases reporting results comparable or better than existing SOTA [66, 365]. The parsed model parameters can also be used in detecting coordinated misinformation attacks.

8.2 Related work

Reverse engineering of models There is a growing area of interest in reverse engineering the hyperparameters of machine learning models, with two types of approaches. First, some methods

treat a model as a black box API by examining its input and output pairs. For example, Tramer *et al*. [300] developed an avatar method to estimate training data and model architectures, while Oh *et al*. [233] trained a set of while-box models to estimate model hyperparameters. The second type of approach assumes that the intermediate hardware information is available during model inference. Hua *et al*. [137] estimated both the structure and the weights of a CNN model running on a hardware accelerator, by using information leaks of memory access patterns. Batina *et al*. [15] estimated the network architecture by using side-channel information such as timing and electromagnetic emanations.

Unlike prior methods which require access to the models or their inputs, our approach can reverse engineer GMs by examining *only* the images generated by these models, making it more suitable for real-world applications. We summarize our approach with previous works in Tab. 8.1.

Fingerprint estimation Every acquisition device leaves a subtle but unique pattern on its captured image, due to manufacturing imperfections. Such patterns are referred to as *device fingerprints*. Device fingerprint estimation [203, 61] was extended to fingerprint estimation of GMs by Marra *et al*. [209], who showed that hand-crafted fingerprints are unique to each GM and can be used to identify an image's source. Ning *et al*. [365] extended this idea to learning-based fingerprint estimation. Both methods rely on the noise signals in the image. Others explored frequency domain information. For example, Wang *et al*. [331] showed that CNN generated images have unique patterns in their frequency domain, regarded as model fingerprints. Zhang *et al*. [372] showed that features extracted from the middle and high frequencies of the spectrum domain were useful in detecting upsampling artifacts produced by GANs.

Unlike prior methods which derive fingerprints directly from noise signals or the frequency domain, we propose several novel loss functions to learn GM fingerprints in an unsupervised manner (Tab. 8.1). We further show that our fingerprint estimation can generalize well to other related tasks.

Deepfake detection Deepfake detection is a new and active field with many recent developments. Rossler *et al*. [266] evaluated different methods for detecting face and mouth replacement manipulation.

Figure 8.2 Example images generated by all 116 GMs in our collected dataset (one image per model).

Others proposed SVM classifiers on colour difference features [213]. Guarnera *et al*. [114] used Expectation Maximization [220] algorithm to extract features and convolution traces for classification. Marra *et al*. [210] proposed a multi-task incremental learning to classify new GAN generated images. Chai *et al*. [36] introduced a patch-based classifier to exaggerate regions that are more easily detectable. An attention mechanism [311] was proposed by Hao *et al*. [66] to improve the performance of deepfake detection. Masi *et al*. [211] amplifies the artifacts produced by deepfake methods to perform the detection. Nirkin *et al*. [229] seek discrepancies between face regions and their context [228] as telltale signs of manipulation. Finally, Liu [192] uses the spatial information as an additional channel for the classifier. In our work, the estimated fingerprint is fed into a classifier for genuine *vs*. fake classification.

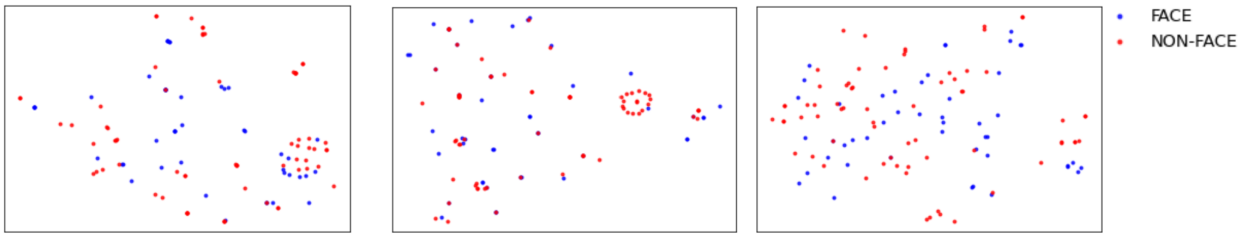


Figure 8.3 t-SNE visualization for ground-truth vectors for (a) network architecture, (b) loss function and (c) network architecture and loss function combined. The ground-truth vectors are fairly distributed across the embedding space regardless of the face/non-face data.

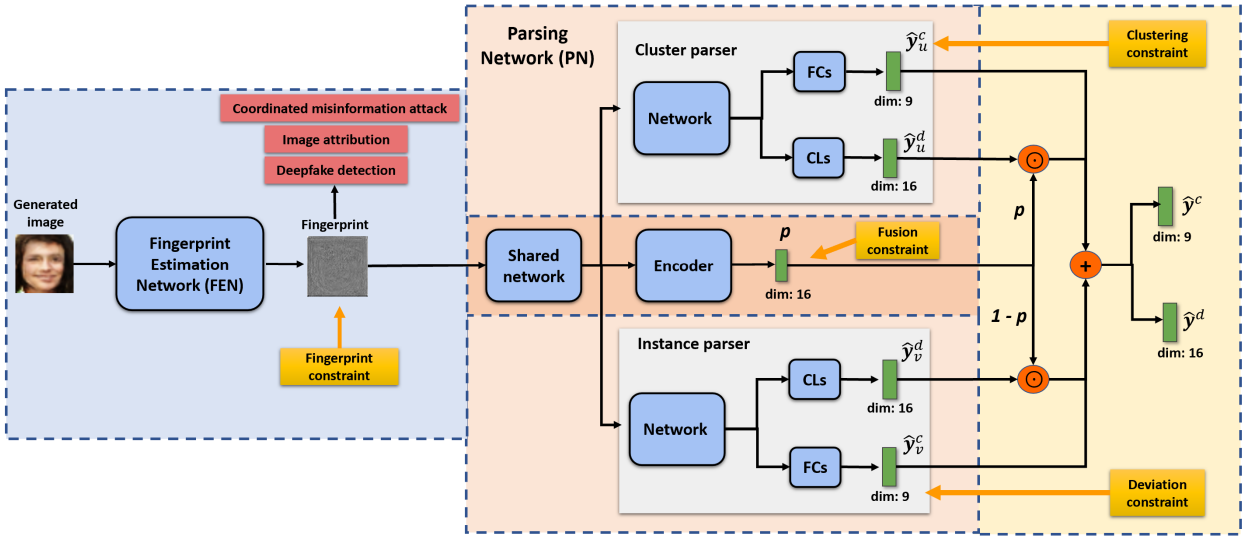


Figure 8.4 Our framework includes two components: 1) the FEN is trained with four objectives for fingerprint estimation; and 2) the PN consists of a shared network, two parsers to estimate mean and deviation for each parameter, an encoder to estimate fusion parameter, fully connected layers (FCs) for continuous type parameters and separate classifiers (CLs) for discrete type parameters in network architecture and loss function prediction. Blue boxes denote trainable components; green boxes denote feature vectors; orange boxes denote loss functions; red boxes denote other tasks our framework can handle; black arrows denote data flow; orange arrows denote loss supervisions. Best viewed in color.

8.3 Proposed approach

In this section, we first introduce our collected dataset in Sec. 8.3.1. We then present the fingerprint estimation method in Sec. 8.3.2 and model parsing in Sec. 8.3.3. Finally, we apply our estimated fingerprints to deepfake detection, image attribution, and detecting coordinated misinformation attacks, as described in Sec. 8.3.4.

Table 8.2 Hyper-parameters representing the network architectures of GMs. (KEYS: cont. int.: continuous integer).

Parameter	Type	Range	Parameter	Type	Range	Parameter	Type	Range
# layers	cont. int.	[5, 95]	# filter	cont. int.	[0, 8365]	non-linearity type in blocks	multi-class	0, 1, 2, 3
# convolutional layers	cont. int.	[0, 92]	# parameters	cont. int.	$[0.36M, 267M]$	non-linearity type in last layer	multi-class	0, 1, 2, 3
# fully connected layers	cont. int.	[0, 40]	# blocks	cont. int.	[0, 16]	up-sampling type	binary	0, 1
# pooling layers	cont. int.	[0, 4]	# layers per block	cont. int.	[0, 9]	skip connection	binary	0, 1
# normalization layers	cont. int.	[0, 57]	normalization type	multi-class	0, 1, 2, 3	down-sampling	binary	0, 1

Table 8.3 Loss function types used by all GMs. We group the 10 loss functions into three categories. We use the binary representation to indicate presence of each loss type in training the respective GM.

Category	Loss function
Pixel-level	L_1
	L_2
	Mean squared error (MSE)
	Maximum mean discrepancy (MMD)
	Least squares (LS)
Discriminator	Wasserstein loss for GAN (WGAN)
	Kullback–Leibler (KL) divergence
	Adversarial
	Hinge
Classification	Cross-entropy (CE)

8.3.1 Data collection

We make the first attempt to study the model parsing problem. Since data drives research, it is essential to collect a dataset for our new research problem. Given the large number of GMs published in recent years [335, 145], we consider a few factors while deciding which GMs to be included in our dataset. First of all, since it is desirable to study if model parsing is content-dependent, we hope to collect GMs with as diverse content as possible, such as the face, digits, and generic scenes. Secondly, we give preference to GMs where either the authors have publicly released pre-trained models, generated images, or the training script. Third, the network architecture of the GM should be clearly described in the respective paper.

To this end, we assemble a list of 116 publicly available GMs, including ProGan [156], StyleGAN [158], and others. A complete list is provided in the supplementary material. For each GM, we collect $1,000$ generated images. Therefore, our dataset $\mathcal{D}$ comprises of $116,000$ images. We show example images in Fig. 8.2. These GMs were trained on datasets with various contents, such as CelebA [200], MNIST [72], CIFAR10 [168], ImageNet [71], facades [385], edges2shoes [385], and apple2oranges [385]. The dataset is available here.

We further document the model hyperparameters for each GM as reported in their papers.

Specifically, we investigate two aspects: network architecture and training loss functions. We form a super-set of 15 network architecture parameters (*e.g.*, number of layers, normalization type) and 10 different loss function types. We obtain a large-scale fake image dataset $\mathbb{D} = \{\mathbf{X}_i, \mathbf{y}_i^n, \mathbf{y}_i^l\}_{i=1}^N$ where $\mathbf{X}_i$ is a fake image, $\mathbf{y}_i^n \in \mathbb{R}^{15}$ and $\mathbf{y}_i^l \in \mathbb{R}^{10}$ represent the ground-truth network architecture and loss functions, respectively. We also show the t-SNE distribution for both network architecture and loss functions in Fig. 8.3 for different types of models and datasets. We observe that the ground-truth vectors for both network architecture and loss function are evenly distributed across the space for both types of data: face and non-face.

8.3.2 Fingerprint estimation

We adopt a network structure similar to the DnCNN model used in [370]. As shown in Fig. 8.4, the input to FEN is a generated image $\mathbf{X}$, and the output is a fingerprint image $\mathbf{F}$ of the same size. Motivated by prior works on physical fingerprint estimation [153, 372, 331, 365, 209], we define the following four constraints to guide our estimated fingerprints to have the desirable properties.

Magnitude loss Fingerprints can be considered as image noise patterns with small magnitudes. Similar assumptions were made by others when estimating spoof noise for spoofed face images [153] and sensor noise for genuine images [203]. The first constraint is thus proposed to regularize the fingerprint image to have a low magnitude with an L_2 loss:

$$J_m = ||\mathbf{F}||_2^2. \tag{8.1}$$

Spectrum loss Previous work observed that fingerprints primarily lie in the middle and high-frequency bands of an image [372]. We thus propose to minimize the low-frequency content in a fingerprint image by applying a low pass filter to its frequency domain:

$$J_s = ||\mathcal{L}(\mathcal{F}(\mathbf{F}), f)||_2^2, \tag{8.2}$$

where $\mathcal{F}$ is the Fourier transform, $\mathcal{L}$ is the low pass filter selecting the $f \times f$ region in the center of the 2D Fourier spectrum and making everything else zero.

Repetitive loss Amin *et al*. [153] noted that the noise characteristics of an image are repetitive and exist everywhere in its spatial domain. Such repetitive patterns will result in a large magnitude in

the high-frequency band of the fingerprint. Therefore, we propose to maximize the high-frequency information to encourage this repetitive pattern:

$$J_r = -\max\{\mathcal{H}(\mathcal{F}(\mathbf{F}), f)\}, \tag{8.3}$$

where $\mathcal{H}$ is a high pass filter assigning the $f \times f$ region in the center of the 2D Fourier spectrum to zero.

Energy loss. Wang *et al*. [331] showed that unique patterns exist in the Fourier spectrum of the image generated by CNN networks. These patterns have similar energy in the vertical and horizontal directions of the Fourier spectrum. Our final constraint is proposed to incorporate this observation:

$$J_e = ||\mathcal{F}(\mathbf{F}) - \mathcal{F}(\mathbf{F})^T||_2^2, \tag{8.4}$$

where $\mathcal{F}(\mathbf{F})^T$ is the transpose of $\mathcal{F}(\mathbf{F})$.

These constraints guide the training of our fingerprint estimation. As shown in Fig. 8.4, the fingerprint constraint is given by:

$$J_f = \lambda_1 J_m + \lambda_2 J_s + \lambda_3 J_r + \lambda_4 J_e, \tag{8.5}$$

where λ_1, λ_2, λ_3, λ_4 are the loss weights for each term.

8.3.3 Model parsing

The estimated fingerprint is expected to capture unique patterns generated from a GM. Prior works adopted fingerprints for deepfake detection [213, 114] and image attribution [365]. However, we go beyond those efforts by parsing the hyperparameters of GMs. As shown in Fig. 8.4, we perform prediction using two parsers, namely, cluster parser and instance parser. We combine both outputs for network architecture and loss function prediction. We will now discuss the ground truth calculation and our framework in detail.

8.3.3.1 Ground truth hyperparamters

Network architecture In this work, we do not aim to recover the network parameters. The reason is that a typical deep network has millions of network parameters, which reside in a very high dimensional space and is thus hard to predict. Instead, we propose to infer the hyperparameters that

define the network architecture, which are much fewer than the network parameters. Motivated by prior works in neural architecture search [294, 244, 191], we form a set of 15 network architecture parameters covering various aspects of architectures. As shown in Tab. 8.2, these parameters fall into different data types and have different ranges. We further split the network architecture parameters $\mathbf{y}^n$ into two parts: $\mathbf{y}^{n_c} \in \mathbb{R}^9$ for continuous data type and $\mathbf{y}^{n_d} \in \mathbb{R}^6$ for discrete data type.

Loss function In addition to the network architectures, the learned network parameters of trained GM can also impact the fingerprints left on the generated images. These network parameters are determined mainly by the training data and the loss functions used to train these models. We, therefore, explore the possibility of also predicting the training loss functions from the estimated fingerprints. The 116 GMs were trained with 10 types of loss functions as shown in Tab. 8.3. For each model, we compose a ground-truth vector $\mathbf{y}^l \in \mathbb{R}^{10}$, where each element is a binary value indicating whether the corresponding loss is used or not in training this model.

Our framework parses two types of hyperparameters: continuous and discrete. The former includes the continuous network architecture parameters. The latter includes discrete network architecture parameters and loss function parameters. For clarity, we group these parameters into continuous and discrete types in the remaining of this section to describe the model parsing objectives. We use $\mathbf{y}^c$ and $\mathbf{y}^d$ to denote continuous and discrete parameters respectively.

8.3.3.2 Cluster parser prediction

We have observed that directly estimating the hyperparameters independently for each GM yields inferior results. In fact, some of the GMs in our dataset have similar network architectures and/or loss functions. It is intuitive to leverage the similarities among different GMs for better hyperparameter estimation. To do this, we perform k-means clustering to group all GMs into different clusters, as shown in Fig. 8.5. Then we propose to perform cluster-level coarse prediction and GM-level fine prediction, which are subsequently combined to obtain the final prediction results.

As we aim to estimate the parameters for network architecture and loss function, it is intuitive to combine them to perform grouping. Thus, we concatenate the ground truth network architecture

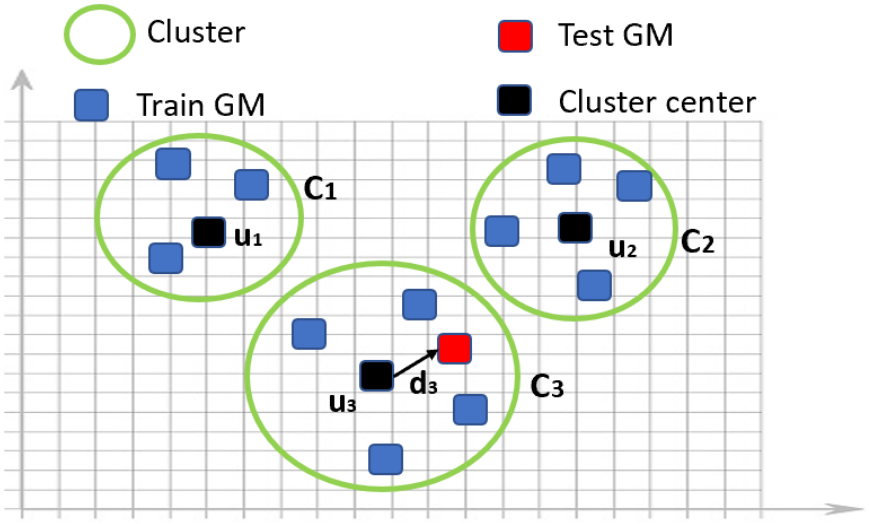


Figure 8.5 The idea of grouping various GMs into different clusters. For the test GM, we estimate its cluster mean and the deviation from that mean to predict network architecture and loss function type.

parameters $\mathbf{y}^n$ and loss function parameters $\mathbf{y}^l$, denoted as $\mathbf{y}^{nl}$. We use these ground truth vectors to perform k-means clustering to find the optimal k-clusters in the dataset $\mathcal{D} = \{\boldsymbol{C}_1, \boldsymbol{C}_2, ...\boldsymbol{C}_k\}$. Our clustering objective can be written as:

$$\underset{\mathcal{D}}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{\mathbf{y}_j^{nl} \in \boldsymbol{C}_i} ||\mathbf{y}_j^{nl} - \mu_i||^2, \tag{8.6}$$

where μ_i is the mean of the ground truth of the GMs in $\boldsymbol{C}_i$.

Our dataset comprises different kinds of GMs, namely GANs, VAEs, AAs, ARs, and NFs. We perform clustering after separating the training data into different kinds of GMs. This is done to ensure that each cluster would belong to one particular kind of GM. Next, we select the value of k *i.e.*, the number of clusters, using the elbow method adopted by previous works [18, 166]. After determining the clusters comprising of similar GMs, we estimate the ground truth $\mathbf{y}_u$ to represent the respective cluster. We estimate this cluster ground truth using different ways for continuous and discrete parameters. For the former, we take the average of each parameter using the ground truth for all GMs in the respective cluster. For the latter, we perform majority voting for every parameter to find the most common class across all GMs in the cluster.

We use different loss functions to perform cluster-level prediction. For continuous parameters,

we perform regression for parameter estimation. As these parameters are in different ranges, we further perform a min-max normalization to bring all parameters to the range of [0, 1]. An L_2 loss is used to estimate the prediction error:

$$J_u^c = ||\hat{\mathbf{y}}_u^c - \mathbf{y}_u^c||_2^2, \tag{8.7}$$

where $\hat{\mathbf{y}}_u^c$ is the cluster mean prediction and $\mathbf{y}_u^c$ is the normalized ground-truth cluster mean.

For discrete parameters, the prediction is made via individual classifiers. Specifically, we train $M = 16$ classifiers (6 for network architecture and 10 for loss function parameters), one for each discrete parameter. The loss term for discrete parameters cluster-prediction is defined as:

$$J_u^d = -\sum_{m=1}^{M} \text{sum}(\mathbf{y}_{u_m}^d \odot \log(\mathcal{S}(\hat{\mathbf{y}}_{u_m}^d))), \tag{8.8}$$

where $\mathbf{y}_{u_m}^d$ is the ground-truth one-hot vector for the respective class in the m-th discrete type parameter, $\hat{\mathbf{y}}_{u_m}^d$ are the class logits, $\mathcal{S}$ is the Softmax function that maps the class logits into the range of $[0, 1]$, $\odot$ is the element-wise multiplication, and sum() computes the summation of a vector's elements.

As shown in Fig. 8.4, the clustering constraint is given by:

$$J_u = \gamma_1 J_u^c + \gamma_2 J_u^d, \tag{8.9}$$

where γ_1 and γ_2 are the loss weights for each term.

8.3.3.3 Instance parser prediction

The cluster parser performs coarse-level prediction. To obtain a more fine-level prediction, we use an instance parser to estimate a GM-level prediction, which ignores any similarity among GMs. This parser aims to predict the deviation of every parameter from the coarse-level prediction. The ground truth deviation vector $\mathbf{y}_v$ can be estimated in different ways for two types of parameters. For continuous type parameters, the deviation can be the difference between the ground truth of the GM and the ground truth of the cluster the GM was assigned. However, in the case of discrete parameters, the actual ground truth class for the parameters can act as the deviation from the most common class estimated in cluster ground truth. We use different loss functions to perform deviation-level prediction. Specifically, we use an L_2 loss to estimate the prediction error for

continuous parameters:

$$J_v^c = ||\hat{\mathbf{y}}_v^c - \mathbf{y}_v^c||_2^2, \tag{8.10}$$

where $\hat{\mathbf{y}}_v^c$ is the deviation prediction and $\mathbf{y}_v^c$ is the deviation ground-truth of continuous data type.

We have noticed the class distribution for some discrete parameters is imbalanced. Therefore, we apply the weighted cross-entropy loss for every parameter to handle this challenge. We train $M = 16$ classifiers, one for each of the discrete parameters. For the m-th classifier with N_m classes ($N_m = 2$ or 4 in our case), we calculate a loss weight for each class as $w_m^i = \frac{N}{N_m^i}$ where N_m^i is the number of training examples for the ith class of m-th classifier, and N is the number of total training examples. As a result, the class with more examples is down-weighted, and the class with fewer examples is up-weighted to overcome the imbalance issue, which will be empirically demonstrated in Fig. 8.9. The loss term for discrete parameters deviation-prediction is defined as:

$$J_v^d = -\sum_{m=1}^{M} \text{sum}(\mathbf{w}_m \odot \mathbf{y}_{v_m}^d \odot \log(\mathcal{S}(\hat{\mathbf{y}}_{v_m}^d))), \tag{8.11}$$

where $\mathbf{y}_{v_m}^d$ is the ground-truth one-hot deviation vector for the m-th classifier, $\mathbf{w}_m$ is a weight vector for all classes in the m-th classifier and $\hat{\mathbf{y}}_{v_m}^d$ are the class logits.

As shown in Fig. 8.4, the deviation constraint is given by:

$$J_v = \gamma_3 J_v^c + \gamma_4 J_v^d. \tag{8.12}$$

where γ_3 and γ_4 are the loss weights for each term.

8.3.3.4 Combining predictions

We use a cluster parser to perform a coarse-level mean prediction and an instance parser to predict a deviation prediction for each GM. The final prediction of our framework, *i.e.*, the prediction at the fine-level is the combination of the outputs of these two parsers. For continuous parameters, we perform the element-wise addition of the coarse-level mean and deviation prediction:

$$\hat{\mathbf{y}}^c = \hat{\mathbf{y}}_u^c + \hat{\mathbf{y}}_v^c, \tag{8.13}$$

For discrete parameters, we have observed that element-wise addition of the logits for every classifier in both parsers didn't perform well. Therefore, to integrate the outputs, we train an encoder

network to predict a fusion parameter $\hat{p^d} \in [0, 1]$ for each classifier. For any parameter, the value of the fusion parameter is 1 if the cluster class is the same as the GM class, encouraging the parsing network to give importance to the cluster parser output. The value of the fusion parameter is 0 if the GM class is different from the cluster class. Therefore, for m-th classifier, the training of the model is supervised by the ground truth p_m^d as defined below:

$$p_m^d = \begin{cases} 1, & \mathbf{y}_{u_m}^d = \mathbf{y}_{v_m}^d \\ 0, & \mathbf{y}_{u_m}^d \neq \mathbf{y}_{v_m}^d. \end{cases} \tag{8.14}$$

To train our encoder, we use the ground truth fusion parameter $\mathbf{p}^d$ which is the concatenation for all parameters. The training is done via cross-entropy loss as shown below:

$$J_p = -\sum_{m=1}^{M} (p_m^d \log(\mathcal{G}(\hat{p}_m^d)) + (1 - p_m^d)\log(1 - \mathcal{G}(\hat{p}_m^d))). \tag{8.15}$$

where $\mathcal{G}$ is the Sigmoid function that maps the class logits into the range of $[0, 1]$.

As shown in Fig. 8.4 for discrete parameters, the final prediction is given by:

$$\hat{\mathbf{y}}^d = \hat{\mathbf{p}}^d \odot \hat{\mathbf{y}}_u^d + (\mathbf{1} - \hat{\mathbf{p}}^d) \odot \hat{\mathbf{y}}_v^d. \tag{8.16}$$

The overall loss function for model parsing is given by:

$$J = J_f + J_u + J_v + \gamma_5 J_p. \tag{8.17}$$

where γ_5 is the loss weight for fusion constraint. Our framework is trained end-to-end with fingerprint estimation (Eq. (8.5)) and model parsing (Eq. (8.17)).

8.3.4 Other applications

In addition to model parsing, our fingerprint estimation can be easily leveraged for other applications such as detecting coordinated misinformation attacks, deepfake detection and image attribution.

Coordinated misinformation attack In coordinated misinformation attacks, the attackers often use the same model to generate multiple fake images. One way to detect such attacks is to classify whether two fake images are generated from the same GM, despite that this GM might be unseen to the classifier. This task is not straightforward to perform by prior works. However, given the

ability of our model parsing, this is the ideal task that we can contribute. To perform this binary classification task, we use the parsed network architecture and loss function parameters to calculate the similarity score between two test images. We calculate the cosine similarity for continuous type parameters and fraction of the number of parameters having same class for discrete type. Both cosine similarity and fraction of parameters are averaged to get the similarity score. Comparing the cosine similarity with a threshold will lead to the binary classification decision of whether two images come from the same GM or not.

Deepfake detection We consider the binary classification of an image as either genuine or fake. We add a shallow network on the generated fingerprint to predict the probabilities of being genuine or fake. The shallow network consists of five convolution layers and two fully connected layers. Both genuine and fake face images are used for training. Both FEN and the shallow network are trained end-to-end with the proposed fingerprint constraints (Eq. (8.5)) and a cross-entropy loss for genuine *vs*.fake classification. Note that the fingerprint constraints (Eq. (8.5)) are not applied to the genuine input face images.

Image attribution We aim to learn a mapping from a given image to the model that generated it if it is fake or classified as genuine otherwise. All models are known during training. We solve image attribution as a closed-set classification problem. Similar to deepfake detection, we add a shallow network on the generated fingerprint for model classification with the cross-entropy loss. The shallow network consists of two convolutional layers and two fully connected layers.

8.4 Experiments

8.4.1 Settings

Dataset As described in Sec. 8.3.1, we have collected a fake image dataset consisting of $116K$ images from 116 GMs ($1K$ images per model) for model parsing experiments. These models can be split into two parts: 47 face models and 69 non-face models. Instead of performing one split of training and testing sets, we carefully construct four different splits with a focus on curating well-represented test sets. Specifically, each testing set includes six GANs, two VAEs, two ARs, one AA and one NF model. We perform cross-validation to train on 104 models and evaluate on

the remaining 12 models in testing sets. The performance is averaged across four testing sets.

For deepfake detection experiments, we conduct experiments on the recently released Celeb-DF dataset [184], consisting of 590 real and 5,639 fake videos. For image attribution experiments, a source database with genuine images needs to be selected, from which the fake images can be generated by various GAN models. We select two source datasets: CelebA [184] and LSUN [364], for two experiments. From each source dataset, we construct a training set of $100K$ genuine and $100K$ fake face images produced by each of the same four GAN models used in Yu *et al*. [365], and a testing set with $10K$ genuine and $10K$ fake images per model.

Implementation details Our framework is trained end-to-end with the loss functions of Eq. (8.5) and Eq. (8.17). The loss weights are set to make the magnitudes of all loss terms comparable: $\lambda_1 = 0.05$, $\lambda_2 = 0.001$, $\lambda_3 = 0.1$, $\lambda_4 = 1$, $\gamma_1 = 5$, $\gamma_2 = 5$, $\gamma_3 = 5$, $\gamma_4 = 5$, $\gamma_5 = 5$, $\gamma_6 = 5$, $\gamma_7 = 1$, $\gamma_8 = 1$. The value of f for spectrum loss and repetitive loss in the fingerprint estimation is set to 50. For each of the four test sets, we calculate the number of clusters k using the elbow method. We divide the data into different GM types and perform k-means clustering separately for each type. According to the sets defined in the supplementary, we obtain the value of k as 11, 11, 15, and 13. We use Adam optimizer with a learning rate of 0.0001. Our framework is trained with a batch size of 32 for 10 epochs. All the experiments are conducted using NVIDIA Tesla K80 GPUs.

Evaluation metrics For continuous type parameters, we report the L_1 error for the regression estimation of continuous type parameters. We also report the p-value of t-test, correlation coefficient, coefficient of determination [288] and slope of the RANSAC regression line [93] to show the effectiveness of regression in our approach. For discrete type parameters, as there is imbalance in the dataset for different parameters, we compute the F1 score [94, 147] for classification performance. We also report classification accuracy for discrete-type parameters. For all cross-validation experiments, we report the averaged results across all images and all GMs.

8.4.2 Model parsing results

As we are the first to attempt GM parsing, there are no prior works for comparison. To provide a baseline, we, therefore, draw an analogy with the image attribution task, where each model is

represented as a one-hot vector and different models have equal inter-model distances in the high-dimensional space defined by these one-hot vectors. In model parsing, we represent each model as a 25-D vector consisting of network architectures (15-D) and training loss functions (10-D). Thus, these models are not of equal distance in the 25-D space.

Based on the aforementioned observation, we define a baseline, referred to here as *random ground-truth*. Specifically, for each parameter, we shuffle the values/classes across all 116 GMs to ensure that the assigned ground-truth is different from the actual ground-truth but also preserves the actual distribution of each parameter, which means that the random ground-truth baseline is not based on random chance. These random ground-truth vectors have the same properties as our ground-truth vectors in terms of non-equal distances. But the shuffled ground truths are meaningless and are not corresponding to their true model hyperparameters. We train and test our proposed approach on this randomly shuffled ground-truth. Due to the random nature of this baseline, we perform three random shuffling and then report the average performance. We also evaluate a baseline of always predicting the mean for continuous hyperparameters, and always predicting the mode for discrete hyperparameters across the four sets. These mean/mode values of the hyperparameters are both measures of central tendency to represent the data, and they might result in a good enough performance for model parsing.

To validate the effects of our proposed fingerprint estimation constraints, we conduct an ablation study and train our framework end-to-end with only the model parsing objective in Eq. (8.17). This results in the *no fingerprint* baseline. Finally, to show the importance of our clustering and deviation parser, we estimate the network architecture and loss functions using just one parser, which estimates the parameters directly instead of a mean and deviation. We refer to this as *using one parser* baseline.

Network architecture prediction We report the results of network architecture prediction in Tab. 8.4 for the 4 testing sets, as defined in Sec. 8.4.1. Our method achieves a much lower L_1 error compared to the random ground-truth baseline for continuous type parameters and higher classification accuracy and F1 score for discrete type parameters. This result indicates that there is indeed a

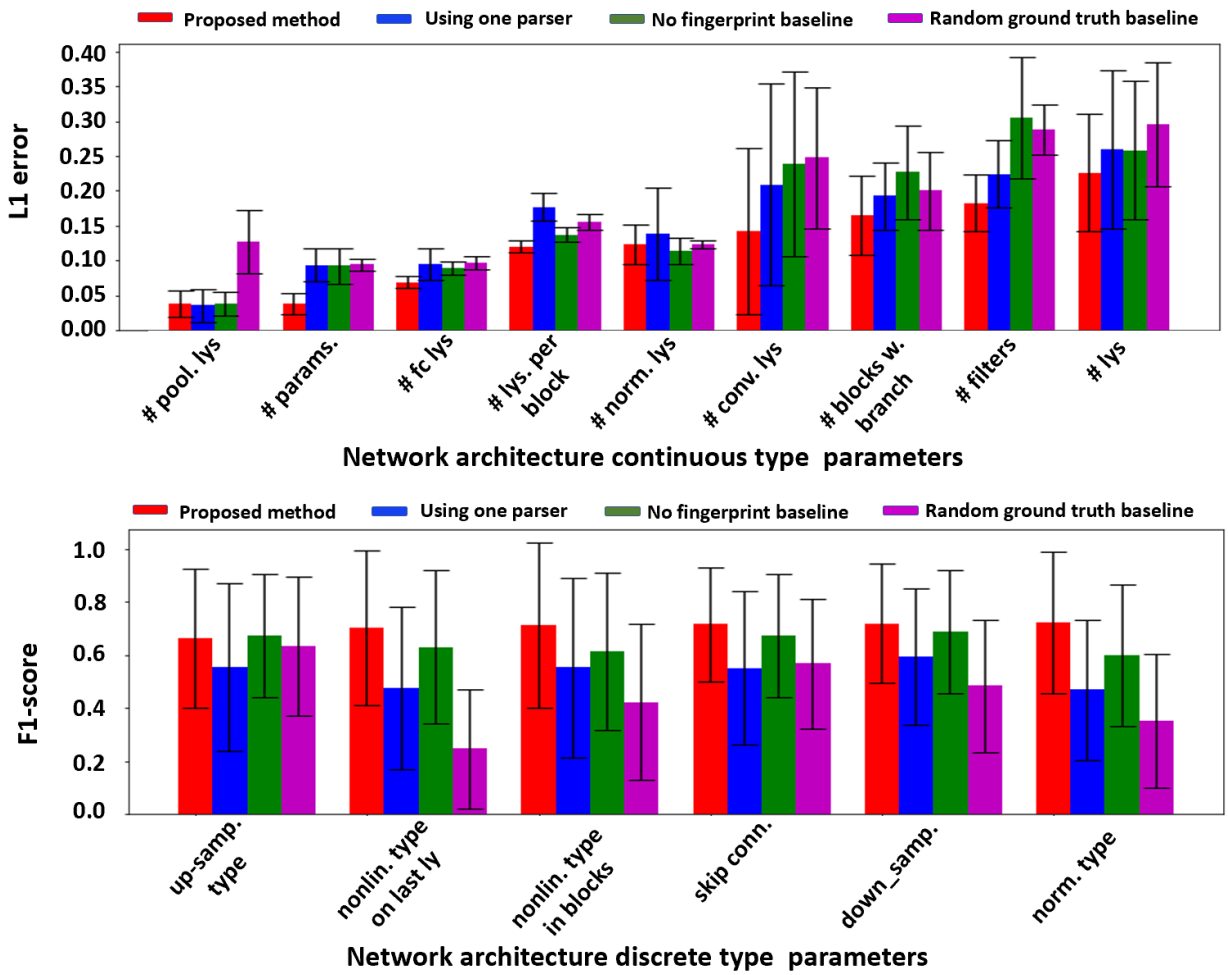


Figure 8.6 L_1 error and F1 score for continuous and discrete parameters respectively of network architecture averaged across all images of all models in the 4 test sets. Not only we have better average performance, but also our standard deviations are smaller.

much stronger and generalized correlation between generated images and the embedding space of meaningful architecture hyper-parameters and loss function types, compared to a random vector of the same length and distribution. This correlation is the foundation of why model parsing of GMs can be a valid and feasible task. Our approach also outperforms the mean/mode baseline, proving that always predicting the mean of the data for continuous parameters is not good enough. Removing fingerprint estimation objectives leads to worse results showing the importance of the fingerprint estimation in model parsing. We demonstrate the effectiveness of estimating mean and deviation by evaluating the performance of using just one parser. Our method clearly outperforms the approach of using one parser.

Table 8.4 Performance of network architecture prediction. We use L_1 error, p-value, correlation coefficient, coefficient of determination and slope of RANSAC regression line for continuous type parameters. For discrete parameters, we use F1 score and classification accuracy. We also show the standard deviation over all the test samples for L_1 error. The first value is the standard deviation across sets, while the second one is across the samples. The p-value would be estimated for every ours-baseline pair. Our method performs better for both types of variables compared to the three baselines. [KEYS: corr.: correlation, coef.: coefficient, det.: determination].

Method	Continuous type					Discrete type	
	L_1 error ↓	P-value ↓	Corr. coef. ↑	Coef. of det. ↑	Slope ↑	F1 score ↑	Accuracy ↑
Random ground-truth	0.184 ± 0.019/0.036	0.006 ± 0.001	0.261 ± 0.181	0.315 ± 0.095	0.592 ± 0.041	0.529 ± 0.078	0.575 ± 0.097
Mean/mode	0.164 ± 0.011/0.016	0.035 ± 0.005	0.326 ± 0.112	0.467 ± 0.015	0.632 ± 0.024	0.612 ± 0.048	0.604 ± 0.046
No fingerprint	0.170 ± 0.035/0.012	0.017 ± 0.004	0.738 ± 0.014	0.605 ± 0.152	0.892 ± 0.021	0.700 ± 0.032	0.663 ± 0.104
Using one parser	0.161 ± 0.028/0.035	0.032 ± 0.002	0.226 ± 0.030	0.512 ± 0.116	–0.529 ± 0.075	0.607 ± 0.034	0.593 ± 0.104
Ours	**0.149 ± 0.019/0.014**	-	**0.744 ± 0.098**	**0.612 ± 0.161**	**0.921 ± 0.021**	**0.718 ± 0.036**	**0.706 ± 0.040**

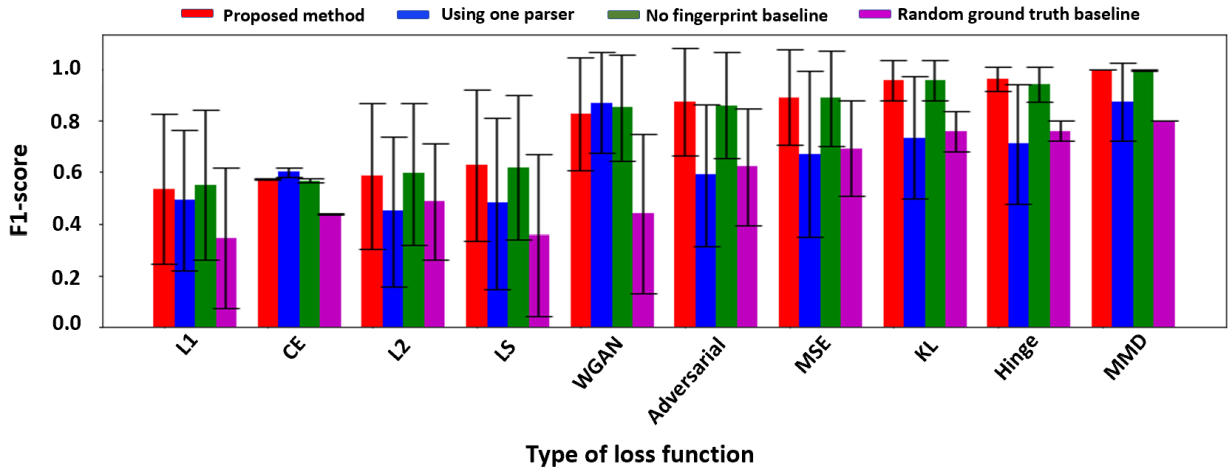


Figure 8.7 F1 score for each loss function type at coarse and fine levels averaged across all images of all models in the 4 test sets. We also show the standard deviation of performance across different sets.

Table 8.5 F1 score and classification accuracy for loss type prediction. Our method performs better than all the three baselines.

Method	Loss function prediction	
	F1 score ↑	Classification accuracy ↑
Random ground-truth	0.636 ± 0.017	0.716 ± 0.028
Mean/mode	0.751 ± 0.027	0.736 ± 0.056
No fingerprint	0.800 ± 0.116	0.763 ± 0.079
Using one parser	0.687 ± 0.036	0.633 ± 0.052
Ours	**0.813 ± 0.019**	**0.792 ± 0.021**

Fig. 8.6 shows the detailed L_1 error and F1 score for all network architecture parameters. We observe that our method performs substantially better than the random ground-truth baseline for almost all parameters. As for the no fingerprint and using one parser baselines, our method is still better in most cases with a few parameters showing similar results. We also show the standard

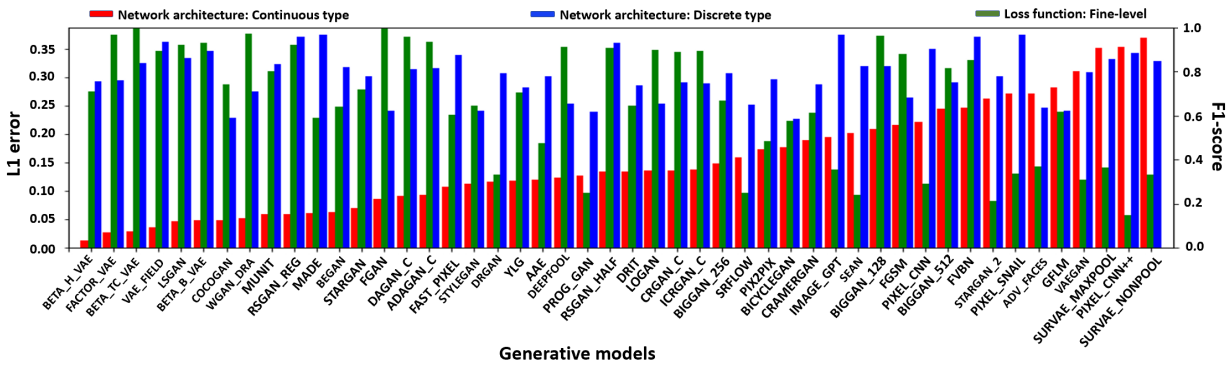


Figure 8.8 Performance of all GMs in our 4 testing sets. Similar performance trends are observed for network architecture and loss functions, *i.e.*, if the L_1 error is small for continuous type parameters in network architecture, the high F1 score is also observed for discrete type parameters in network architecture and loss function. In other words, the abilities to reverse engineer the network architecture and loss function types for one GM are reasonably consistent.

Table 8.6 Performance comparison by varying the training and testing data for face and non-face GMs. Testing performance on non-face GMs is better compared to face GMs. Training and testing on the same content produces better results than on the different contents.We also show the standard deviation over all the test samples for L_1 error. The first value is the standard deviation across sets, while the second one is across the samples.

Test GMs (# GMs)	Train GMs (# GMs)	Network architecture		Loss function
		Continuous type L_1 error ↓	Discrete type F1 score ↑	F1 score ↑
Face (6)	Face (41)	0.139 ± 0.042/0.015	**0.729 ± 0.106**	0.788 ± 0.146
	Non-face (69)	0.213 ± 0.066/0.136	0.688 ± 0.125	0.759 ± 0.100
	Full (110)	**0.118 ± 0.046/0.040**	0.712 ± 0.129	**0.833 ± 0.136**
Non-face (6)	Non-face (63)	0.118 ± 0.021/0.049	0.794 ± 0.110	0.864 ± 0.094
	Face (47)	0.125 ± 0.031/0.028	0.667 ± 0.099	0.858 ± 0.115
	Full (110)	**0.082 ± 0.045/0.049**	**0.832 ± 0.046**	**0.886 ± 0.061**
Random guess		0.393	0.500	0.500

deviation of every estimated parameter for all the methods. Our proposed approach in general has smaller standard deviations than the two baselines. For continuous type parameters, we further show the effectiveness of regression prediction by evaluating three metrics namely, correlation coefficient, coefficient of determination and slope of RANSAC regression line. These metrics are evaluated between prediction and ground-truth. Further, we also estimate a p-value of a t-test, where the null hypothesis is as follows: the sequence of sample-wise L_1 error differences between our method and the baseline method is sampled from zero-mean Gaussian. This p-value would be estimated for every ours-baseline pair. We report the mean and the standard deviation across all

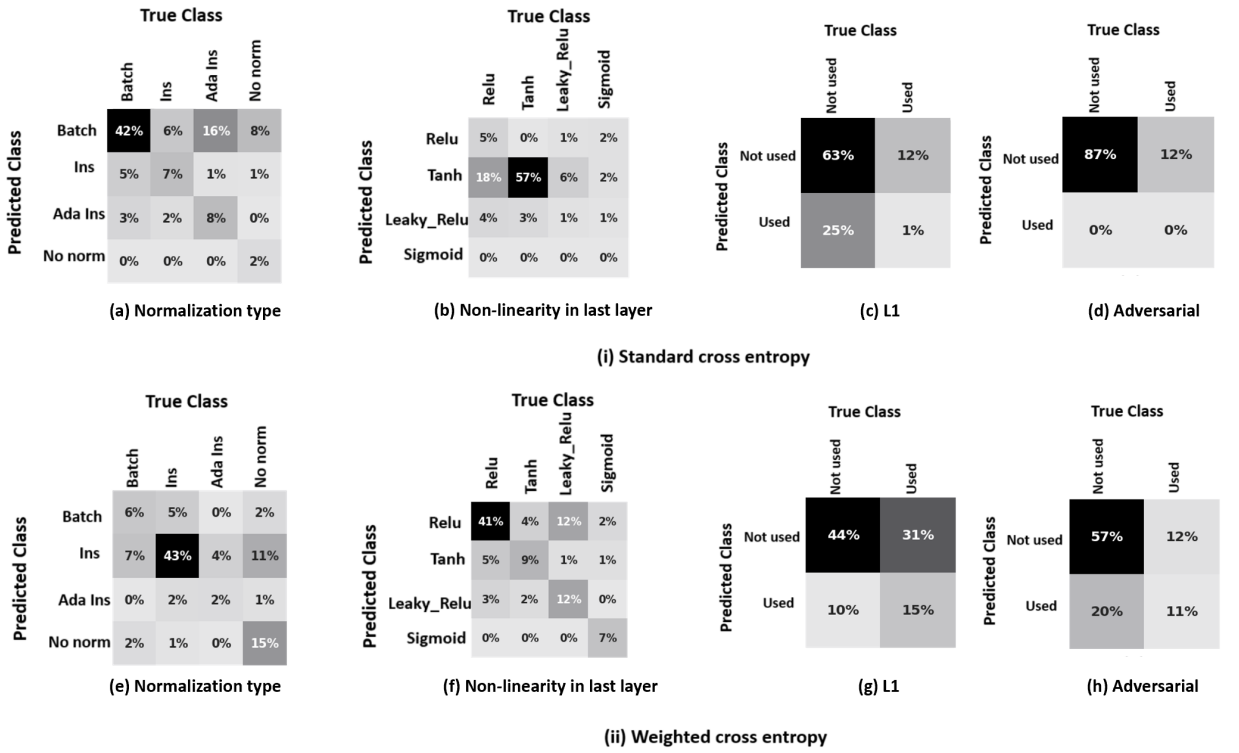


Figure 8.9 Confusion matrix in the estimation of four parameters in the network architecture and loss function. (a)-(d): Standard cross-entropy and (e)-(f): Weighted cross entropy. Weighted cross entropy handles imbalance data much better than the standard cross entropy which usually predicts one class.

four sets. The p-value of our approach when compared to all the three baselines is less than 0.05, thereby rejecting the null hypothesis and proving our improvement is statistically significant. For other three metrics, the values closer to 1 shows effective regression. For our method, we have slope of 0.921, correlation coefficient of 0.744 and coefficient of determination as 0.612 which shows the effectiveness of our approach. Further, our approach outperforms all the baselines for all three metrics.

Loss function prediction We calculate the F1 score and classification accuracy for loss function parameters. The performance are shown in Tab. 8.5. For the random ground-truth baseline, the performance is close to a random guess. Our approach performs much better than all the baselines. Fig. 8.7 shows the detailed F1 score for all loss function parameters. Apparently our method works better than all the baselines for almost all parameters. We also show the standard deviation of every estimated parameter for all the methods. Similar behaviour of standard deviation for different

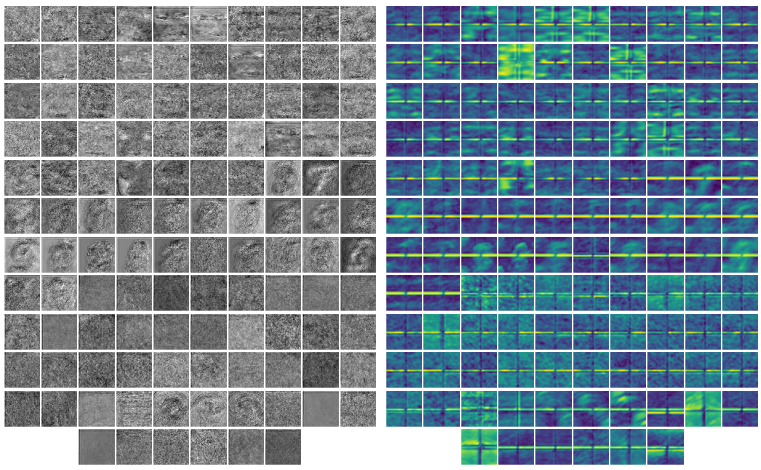


Figure 8.10 Estimated fingerprints (left) and corresponding frequency spectrum (right) from one generated image of each of 116 GMs. Many frequency spectrums show distinct high-frequency signals, while some appear to be similar to each other.

Table 8.7 Ablation study of the 4 loss terms in fingerprint estimation. Removing any one loss for fingerprint estimation deteriorates the performance with the worst results in the case of removing all losses. [KEYS: fing.: fingerprint]. We also show the standard deviation over all the test samples for L_1 error. The first value is the standard deviation across sets, while the second one is across the samples.

Loss removed	Network architecture		Loss function
	Continuous type L_1 error ↓	Discrete type F1 score ↑	F1 score ↑
Magnitude loss	$0.156 \pm 0.007/0.009$	0.674 ± 0.012	0.755 ± 0.046
Spectrum loss	$\mathbf{0.149 \pm 0.022/0.016}$	0.676 ± 0.034	0.786 ± 0.042
Repetitive loss	$0.150 \pm 0.018/0.026$	0.708 ± 0.031	0.794 ± 0.031
Energy loss	$0.162 \pm 0.032/0.038$	0.703 ± 0.045	0.785 ± 0.028
All (no fing.)	$0.170 \pm 0.035/0.037$	0.700 ± 0.032	0.800 ± 0.016
Nothing (ours)	$\mathbf{0.149 \pm 0.019/0.014}$	$\mathbf{0.718 \pm 0.036}$	$\mathbf{0.813 \pm 0.019}$

methods was observed as in the network architecture. Fig. 8.8 provides another perspective of model parsing by showing the performance in terms of 48 unique GMs across our 4 testing sets.

Practical Usage of Model Parsing. As our work is the first one to propose the task of model parsing, it's beneficial to ask the question: *what is the performance desired for practical usage of model parsing in the real world?* To answer this question, we can expect that an error less than 10% can be considered useful for the practical application of model parsing. The rationale is the following. We consider two of the most similar generative models, RSGAN_HALF and RSGAN_QUAR, in our dataset. Upon further analysis, we observe that these models differ in only 2 out of 15 parameters. Therefore, we argue that an error rate below 10% is reasonable for practical

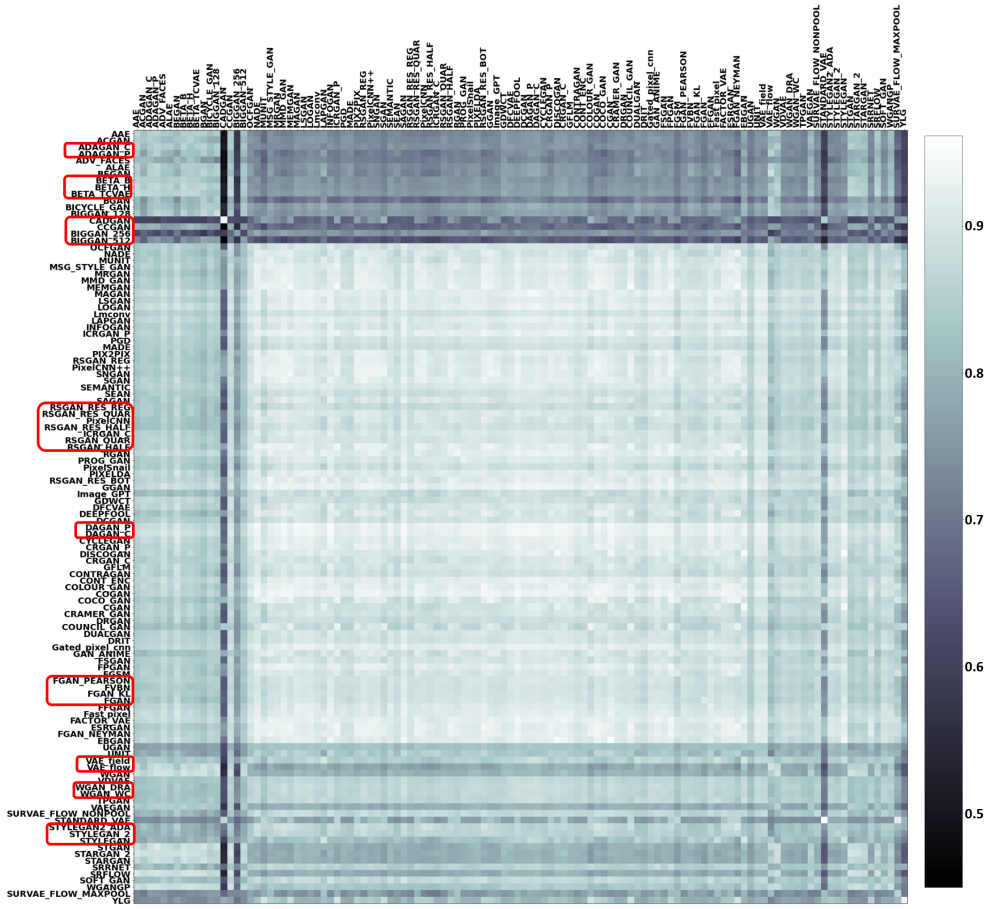


Figure 8.11 Cosine similarity matrix for pairs of 116 GM's fingerprints. Each element of this matrix is the average Cosine similarities of 50 pairs of fingerprints from two GMs. We see the higher intra-GM and lower inter-GM similarities. We can also see GMs with similar network architecture or loss function are clustered together, as shown in the red boxes on the left.

purposes as this error is less than the difference between the two most similar generative models. Therefore, for the task of model parsing, we expect L_1 error of less than 0.1 and an $F1$ score of over 90% for practical usage. Our proposed approach achieves an L_1 error slightly above 10% (0.14) and an $F1$ score of 80%, both of which have reasonable margins toward the above mentioned thresholds.

8.4.3 Ablation study

Face *vs*.non-face GMs Our dataset consists of 47 GMs trained on face datasets and 69 GMs trained on non-face datasets. Let's denote these GMs as face GMs and non-face GMs, respectively. All aforementioned experiments are conducted by training on 104 GMs and evaluating on 12 GMs.

Table 8.8 Network architecture estimation and loss function prediction when given multiple images of one GM. Performance increases when enlarging the number of images for evaluation from 1 to 10. Performance becomes stable for more than 10 images. We also show the standard deviation over all the test samples for L_1 error. The first value is the standard deviation across sets, while the second one is across the samples.

# images	Network architecture		Loss function
	Continuous type L_1 error ↓	Discrete type F1 score ↑	F1 score ↑
1	0.215 ± 0.054/0.067	0.696 ± 0.089	0.798 ± 0.010
10	0.151 ± 0.033/0.039	**0.726 ± 0.075**	0.793 ± 0.070
100	**0.145 ± 0.032/0.036**	0.721 ± 0.073	0.789 ± 0.071
500	0.146 ± 0.033/0.031	0.720 ± 0.070	**0.808 ± 0.007**

Here we conduct an ablation study to train and evaluate on different types of GMs. We study the performance on face and non-face testing GMs when training on three different training sets, including only face GMs, only non-face GMs and all GMs. Note that all testing GMs are excluded during training each time. We also add a baseline where both regression and classification make a random guess on their estimation.

The results are shown in Tab. 8.6. We have three observations. First, model parsing for non-face GMs are easier than face GMs. This might be partially due to the generally lower-quality images generated by non-face GMs compared to those by face GMs, thus more traces are remained for model parsing. Second, training and testing on the same content can generate better results than on different contents. Third, training on the full datasets improves some parameter estimation but may hurt other parameters slightly.

Weighted cross-entropy loss As mentioned before, the ground truth of many network hyperparameters have biased distributions. For example, the “normalization type" parameter in Tab. 8.2 has uneven distribution among its 4 possible types. With this biased distribution, our classifier might make a constant prediction to the type with the highest probability in the ground truth, as this could minimize the loss especially for severe biasness. This degenerate classifier clearly has no value to model parsing. To address this issue, we propose to use the weighted cross-entropy loss with different loss weights for each class. These weights are calculated using the ground-truth distribution of every parameter in the full dataset. To validate if the above approach is able to remedy this issue, we

compare it with the standard cross-entropy loss.

Fig. 8.9 shows the confusion matrix for discrete type parameters in network architecture prediction and coarse/fine level parameters in loss function prediction. The rows in the confusion matrix are represented by predicted classes and columns are represented by the ground-truth classes. We clearly see that the classifier is mostly biased towards more frequent classes in all 4 examples, when the standard cross-entropy loss is used. However, this problem is remedied when using the weighted cross-entropy loss, and the classifiers make meaningful predictions.

Fingerprint losses We proposed four loss terms in Sec. 8.3.2 to guide the training of the fingerprint estimation including magnitude loss, spectrum loss, repetitive loss and energy loss. We conduct an ablation study to demonstrate the importance of these four losses in our proposed method. This includes four experiments, each removing one of the loss terms and comparing the performance with our proposed method (remove nothing) and no fingerprint baseline (remove all). As shown in Tab. 8.7, removing any loss for fingerprint estimation hurts the performance. Our "no fingerprint" baseline, for which we remove all losses, performs worst of all. Therefore, each loss clearly has a positive effect on the fingerprint estimation and model parsing.

Model parsing with multiple images We evaluate model parsing when varying the number of test images. For each GM, we randomly select 1, 10, 100, and 500 images per GM from different face GMs sets for evaluation. With multiple images per GM, we average the prediction for continuous type parameters and take majority voting for discrete type parameters and loss function parameters. We compute the L_1 error and F1 score for the continuous and discrete type parameters respectively and average the result across different sets. We repeat the above experiment multiple times, each time randomly selecting the number of images. We compare the L_1 error and F1 score for respective parameters. Tab. 8.8 shows noticeable gains with 10 images and minor gains with 100 images. There is not much performance difference when evaluating on 100 or 500 images, which suggests that our framework is robust in generating consistent results when tested on different numbers of generated images by the same GM.

Content-independent fingerprint Ideally our estimated fingerprint should be independent of the

Table 8.9 Evaluation on diffusion models. We also show the standard deviation over all the test samples for L_1 error. The first value is the standard deviation across sets, while the second one is across the samples.

Method	Network architecture		Loss function
	Continuous type L_1 error ↓	Discrete type F1 score ↑	F1 score ↑
Random ground-truth	$0.240 \pm 0.065/0.069$	0.664 ± 0.105	0.619 ± 0.083
No fingerprint	$0.211 \pm 0.080/0.078$	0.764 ± 0.112	0.711 ± 0.085
Using one parser	$0.201 \pm 0.045/0.041$	0.564 ± 0.101	0.654 ± 0.054
Ours	$\mathbf{0.189 \pm 0.051/0.049}$	$\mathbf{0.787 \pm 0.099}$	$\mathbf{0.724 \pm 0.076}$

content of the image. That is, the fingerprint only includes the trace left by the GM while not indicating the content in any way. To validate this, we partition all GMs into four classes based on their contents: FACES (47 GMs), MNIST (25), CIFAR10 (31), and OTHER (13). Every class has images generated by the GMs belong to this class. We feed these images to a pre-trained FEN and obtain their fingerprints. Then we train a shallow network consisting of five convolutional layers and two fully connected layers for a 4-way classification. However, we observe the training cannot converge. This means that our estimated fingerprint from FEN doesn't have any content-specific properties for content classification. As a result, the model parsing of the hyperparameters doesn't leverage the content information across different GMs, which is a desirable property.

Evaluation on diffusion models Due to the recent advancement of diffusion models for fake media generation, we evaluate our approach for these generative models. Specifically, we collect 7 diffusion models with $1K$ images each. We create 4 different test set splits, each set containing 3 diffusion models selected randomly. The remaining diffusion models, along with the full dataset is used for training. The result for our approach along with all the baselines is shown in Tab. 8.9. Our method clearly outperforms all the baselines, indicating the effectiveness of our approach for unseen models proposed in future. We also show the standard deviation over all the test samples for L_1 error. The first value is the standard deviation across sets, while the second one is across the samples.

Table 8.10 Binary classification performance for coordinated misinformation attack.

Method	AUC (%)	Classification accuracy (%)
FEN	83.5	76.85
FEN + PN	**87.3**	**80.6**

8.4.4 Visualization

Fig. 8.10 shows an estimated fingerprint image and its frequency spectrum averaged over 25 randomly selected images per GM. We observe that estimated fingerprints have the desired properties defined by our loss terms, including low magnitude and highlights in middle and high frequencies.

We also find that the fingerprints estimated from different generated images of the same GM are similar. To quantify this, we compute a Cosine similarity matrix $\mathbf{C} \in \mathbb{R}^{116\times116}$ where $\mathbf{C}(i, j)$ is the averaged Cosine similarity of 25 randomly sampled fingerprint pairs from GM i and j. The matrix $\mathbf{C}$ in Fig. 8.11 clearly illustrates the higher intra-GM ad lower inter-GM fingerprint similarities.

8.4.5 Applications

Coordinated misinformation attack Our model parsing framework can be leveraged to estimate whether there exists a coordinated misinformation attack. That is, given two fake images, we hope to classify whether they are generated from the same GM or not. We do so by computing the Cosine similarity between the hyperparameters parsed from the given two images. First, we train our framework on 101 GMs, and test on 15 seen GMs and 15 unseen GMs. The list of GMs are mentioned in the supplementary. To evaluate this task, we report the Area Under Curve (AUC) and the classification accuracy at the optimum threshold. The results are shown in Tab. 8.10 comparing two methods, just using FEN network and using both FEN and PN. We conclude that our framework using FEN and PN can identify whether two images came from the same source with around 80% accuracy. Using only FEN network to compare the similarities of the fingerprint performs worse. This justifies the benefit of using parsed parameters for coordinated misinformation attack.

In fact, due to the nature of our test set, each pair of test samples can come from five different categories, namely, 1. Same seen GM, 2. Same unseen GM, 3. Different seen GMs, 4. Different unseen GMs, and 5. One seen and one unseen GM. We show an analysis of the wrongly classified

Table 8.11 AUC for deepfake detection on the Celeb-DF dataset [184].

Method	Training Data	AUC (%)
Methods training with *pixel-level* supervision		
Xception+Reg [66]	DFFD	64.4
Xception+Reg [66]	DFFD, UADFV	71.2
Methods training with *image-level* supervision		
Two-stream [118]	Private	53.8
Meso4 [2]		54.8
VA-LogReg [212]		55.1
DSP-FWA [126]		64.6
Multi-task [226]	FF	54.3
Capsule [227]	FF++	57.5
Xception-c40 [266]		65.5
Two-branch [211]		73.4
SPSL [192]		**76.8**
SPSL [192] (reproduced)		73.2
Ours (fingerprint)		69.6
Ours (image+fingerprint)		71.1
Ours (image+fingerprint+phase)		74.6
Ours (model parsing)		64.3
HeadPose [350]	UADFV	54.6
FWA [183]		56.9
Xception [66]		52.2
Xception+Reg [66]		57.1
Ours		**64.7**
Xception [66]	DFFD	63.9
Ours		**65.3**
Xception [66]	DFFD, UADFV	67.6
Ours		**70.2**

Table 8.12 Classification rates of image attribution. The baseline results are cited from [365].

Method	CelebA	LSUN
kNN	28.00	36.30
Eigenface [284]	53.28	-
PRNU [209]	86.61	67.84
Yu *et al*. [365]	99.43	98.58
Ours	**99.66**	**99.84**

samples in Fig. 8.12 with respect to total number of samples and total number of samples in each category. Around 70% of the wrongly classified samples belong to the category of images coming from categories having atleast one GM unseen in training which is expected. However, if one of the test GM was seen in training, the number of wrongly classified samples decreased. This can be advantageous in detecting a manipulated image from an unknown GM.

Deepfake detection Our FEN can be adopted for deepfake detection by adding a shallow network for

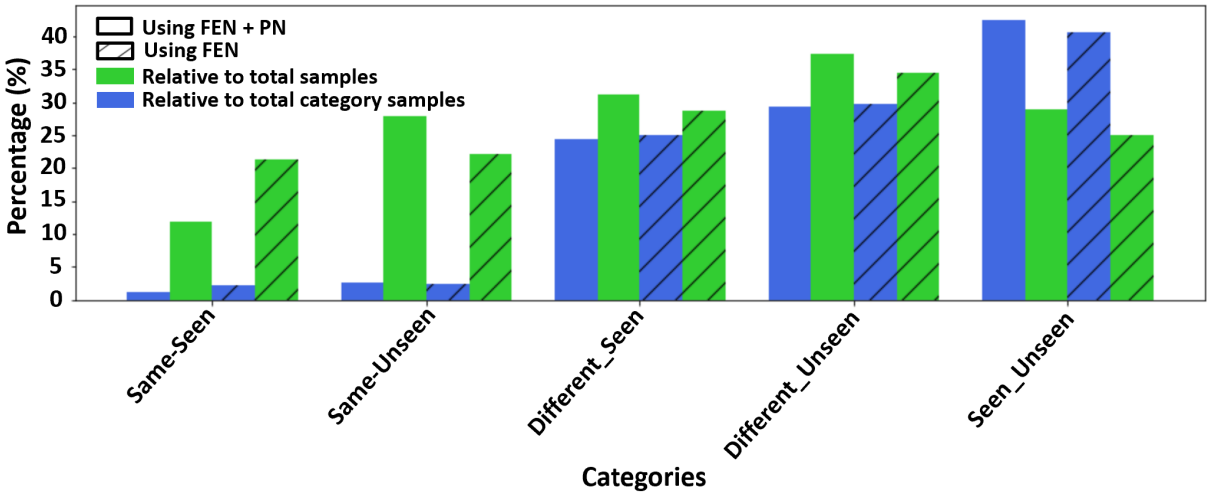


Figure 8.12 Percentage of wrongly classified samples for five different categories of test sample pair. A larger number of sample pairs are wrongly classified if the pair of images come from same unseen GMs.

binary classification. We evaluate our method on the recently introduced Celeb-DF dataset [184]. We experiment with three training sets, UADFV, DFFD, and FF++, in order to compare with previous results. We follow the same training protocols used in [66] for UADFV and DFFD and [192] for FF++.

We report the AUC in Tab. 8.11. Compared with methods trained on UADFV, our approach achieves a significantly better result, despite the more advanced backbones used by others. Our results when trained on DFFD and UADFV fall only slightly behind the best performance reported by Xception+Reg [66]. Importantly, however, they trained with pixel-level supervision which is typically unavailable. These results are provided for completeness, but are not directly comparable to all other methods trained with only image-level supervision for binary classification. Compared to all other methods, our method achieves the highest deepfake detection AUC.

Finally, we compare the performance of our method when trained on FF++ dataset. [192] performs the best by using the phase information as an additional channel to the Xception classifier. However, as the pre-trained models were not released for [192], we reproduce their method and report the performance shown in Tab. 8.11. We observe a performance gap between the reproduced and reported performance which should be further investigated in the future. Following [192], we concatenate the fingerprint information with the RGB image and phase channels which are passed

through a Xception classifier. Our method outperforms the reproduced performance of [192] showing the additional benefit of our fingerprint. Finally, we also perform the classification based on the pre-trained model parsing network and fine-tune it using the classification loss. The performance deteriorated compared to using the fingerprint. This shows that although the model parsing network have some deepfake detection abilities, they are less informative to perform deepfake detection well.

Image attribution Similar to deepfake detection, we use a shallow network for image attribution. The only difference is that image attribution is a multi-class task and depends on the number of GMs during training. Following [365], we train our model on $100K$ genuine and $100K$ fake face images each from four GMs: SNGAN [218], MMDGAN [179], CRAMERGAN [16] and ProGAN [156], for five-class classification. Tab. 8.12 reports the performance. Our result on CelebA [184] and LSUN [364] outperform the performance in [365]. This again validates the generalization ability of the proposed fingerprint estimation.

8.5 Conclusion

In this paper, we define the model parsing problem as inferring the network architectures and training loss functions of a GM from the generative images. We make the first attempt to tackle this challenging problem. The main idea is to estimate the fingerprint for each image and use it for model parsing. Four constraints are developed for fingerprint estimation. We propose hierarchical learning to parse the hyperparameters in coarse-level and fine-level that can leverage the similarities between different GMs. Our fingerprint estimation framework can not only perform model parsing, but also extend to detecting coordinated misinformation attack, deepfake detection and image attribution. We have collected a large-scale fake image dataset from 116 different GMs. Various experiments have validated the effects of different components in our approach.

BIBLIOGRAPHY

[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX-S*, 2018.

[2] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. MesoNet: a compact facial video forgery detection network. In *WIFS*, 2018.

[3] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6836–6846, 2021.

[4] Vishal Asnani, John Collomosse, Tu Bui, Xiaoming Liu, and Shruti Agarwal. Promark: Proactive diffusion watermarking for causal attribution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10802–10811, 2024.

[5] Vishal Asnani, Abhinav Kumar, Suya You, and Xiaoming Liu. PrObeD: Proactive object detection wrapper. In *NeurIPS*, 2023.

[6] Vishal Asnani, Xi Yin, Tal Hassner, Sijia Liu, and Xiaoming Liu. Proactive image manipulation detection. In *CVPR*, 2022.

[7] Vishal Asnani, Xi Yin, Tal Hassner, and Xiaoming Liu. MaLP: Manipulation localization using a proactive scheme. In *CVPR*, 2023.

[8] Vishal Asnani, Xi Yin, Tal Hassner, and Xiaoming Liu. Reverse engineering of generative models: Inferring model hyperparameters from generated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(12):15477–15493, 2023.

[9] Vishal Asnani, Xi Yin, and Xiaoming Liu. Proactive schemes: A survey of adversarial attacks for social good. *arXiv preprint arXiv:2409.16491*, 2024.

[10] Yousef Atoum, Joseph Roth, Michael Bliss, Wende Zhang, and Xiaoming Liu. Monocular video-based trailer coupler detection using multiplexer convolutional neural network. In *ICCV*, 2017.

[11] Kar Balan, Shruti Agarwal, Simon Jenni, Andy Parsons, Andrew Gilbert, and John Collomosse. EKILA: Synthetic media provenance and attribution for generative art. In *CVPR*, 2023.

[12] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015.

[13] Shumeet Baluja. Hiding images in plain sight: Deep steganography. 2017.

[14] Abdullah Bamatraf, Rosziati Ibrahim, and Mohd Najib B Mohd Salleh. Digital watermarking algorithm using LSB. In *ICCAIE*, 2010.

[15] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel. In *USENIXSS*, 2019.

[16] Marc G. Bellemare, Ivo Danihelka, Will Dabney, Shakir Mohamed, Balaji Lakshminarayanan, Stephan Hoyer, and Rémi Munos. The cramer distance as a solution to biased wasserstein gradients. *arXiv preprint arXiv:1705.10743*, 2017.

[17] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, volume 2, page 4, 2021.

[18] Purnima Bholowalia and Arvind Kumar. Ebk-means: A clustering technique based on elbow method and k-means in wsn. *International Journal of Computer Applications*, 105(9), 2014.

[19] Alex Black, Tu Bui, Hailin Jin, Vishy Swaminathan, and John Collomosse. Deep image comparator: Learning to visualize editorial change. In *CVPR WMF*, 2021.

[20] Andreas Blattmann, Robin Rombach, Kaan Oktay, Jonas Müller, and Björn Ommer. Retrieval-augmented diffusion models. *NeurIPS*, 2022.

[21] Aaron F. Bobick and James W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on pattern analysis and machine intelligence*, 23(3):257–267, 2001.

[22] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[23] Daniel Bolya, Sean Foley, James Hays, and Judy Hoffman. TIDE: A general toolbox for identifying object detection errors. In *ECCV*, 2020.

[24] Oliver Bown. AI doesn't like to credit its sources. for artists, that's a problem. *Tatlor*, 2024.

[25] Garrick Brazil and Xiaoming Liu. Pedestrian detection with autoregressive network phases. In *CVPR*, 2019.

[26] Jonathan Brokman, Omer Hofman, Roman Vainshtein, Amit Giloni, Toshiya Shimizu, Inderjeet Singh, Oren Rachmil, Alon Zolfi, Asaf Shabtai, Yuki Unno, and Hisashi Kojima. Montrage: Monitoring training for attribution of generative diffusion models. In *Proc. ECCV*, 2024.

[27] Tu Bui, Shruti Agarwal, Ning Yu, and John Collomosse. RoSteALS: Robust steganography using autoencoder latent space. In *CVPR*, 2023.

[28] Tu Bui, Ning Yu, and John Collomosse. RepMix: Representation mixing for robust attribution of synthesized images. In *ECCV*, 2022.

[29] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-VAE. In *NeurIPS*, 2017.

[30] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *CVPR*, 2018.

[31] Xirong Cao, Xiang Li, Divyesh Jadav, Yanzhao Wu, Zhehui Chen, Chen Zeng, and Wenqi Wei. Invisible watermarking for audio generation diffusion models. In *TPS-ISA*, 2023.

[32] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[33] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX*, 2019.

[34] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *SSP*, 2017.

[35] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[36] Lucy Chai, David Bau, Ser-Nam Lim, and Phillip Isola. What makes fake images detectable? understanding properties that generalize. In *ECCV*, 2020.

[37] Chang Chen, Zhiwei Xiong, Xiaoming Liu, and Feng Wu. Camera trace erasing. In *CVPR*, 2020.

[38] Geng Chen, Si-Jie Liu, Yu-Jia Sun, Ge-Peng Ji, Ya-Feng Wu, and Tao Zhou. Camouflaged object detection via context-aware cross-level fusion. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(10), 2022.

[39] Huili Chen, Bita Darvish Rouhani, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepmarks: A secure fingerprinting framework for digital rights management of deep learning models. In *ICMR*, 2019.

[40] Liang Chen, Yong Zhang, Yibing Song, Lingqiao Liu, and Jue Wang. Self-supervised learning of adversarial example: Towards good generalizations for deepfake detection. In *CVPR*, 2022.

[41] Mo Chen, Jessica Fridrich, Miroslav Goljan, and Jan Lukás. Determining image origin and integrity using sensor noise. *IEEE Transactions on Information Forensics and Security*, 3(1):74–90, 2008.

[42] Ricky T. Q. Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *NeurIPS*, 2018.

[43] Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. DiffusionDet: Diffusion model for object detection. In *CVPR*, 2023.

[44] Shoufa Chen, Peize Sun, Enze Xie, Chongjian Ge, Jiannan Wu, Lan Ma, Jiajun Shen, and Ping Luo. Watch only once: An end-to-end video action detection framework. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8178–8187, 2021.

[45] Wei Chen, Zichen Miao, and Qiang Qiu. Parameter-efficient tuning of large convolutional models. *arXiv preprint arXiv:2403.00269*, 2024.

[46] Wei-Chen Chen, Xin-Yi Yu, and Lin-Lin Ou. Pedestrian attribute recognition in video surveillance scenarios based on view-attribute attention localization. *Machine Intelligence Research*, 2022.

[47] Zejia Chen, Fabing Duan, Francois Chapeau-Blondeau, and Derek Abbott. Training threshold neural networks by extreme learning machine and adaptive stochastic resonance. *Physics Letters A*, 2022.

[48] Xuelian Cheng, Huan Xiong, Deng-Ping Fan, Yiran Zhong, Mehrtash Harandi, Tom Drummond, and Zongyuan Ge. Implicit motion handling for video camouflaged object detection. In *CVPR*, 2022.

[49] Wonwoong Cho, Sungha Choi, David Keetae Park, Inkyu Shin, and Jaegul Choo. Image-to-image translation via group-wise deep whitening-and-coloring transformation. In *CVPR*, 2019.

[50] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. ILVR: Conditioning method for denoising diffusion probabilistic models. In *ICCV*, 2021.

[51] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018.

[52] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018.

[53] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse image

synthesis for multiple domains. In *CVPR*, 2020.

[54] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[55] Pengyu Chu, Zhaojian Li, Kyle Lammers, Renfu Lu, and Xiaoming Liu. Deepapple: Deep learning-based apple detection using a suppression mask R-CNN. *PRL*, 2021.

[56] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*, pages 424–432. Springer, 2016.

[57] John Collomosse and Andy Parsons. To Authenticity, and Beyond! Building safe and fair generative AI upon the three pillars of provenance. *IEEE Computer Graphics and Applications*, May 2024.

[58] MMAction2 Contributors. Openmmlab's next generation video understanding toolbox and benchmark, 2020.

[59] Mickael Cormier, Yannik Schmid, and Jürgen Beyerer. Enhancing skeleton-based action recognition in real-world scenarios through realistic data augmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 290–299, 2024.

[60] Davide Cozzolino, Justus Thies, Andreas Rössler, Christian Riess, Matthias Nießner, and Luisa Verdoliva. Forensictransfer: Weakly-supervised domain adaptation for forgery detection. *arXiv preprint arXiv:1812.02510*, 2018.

[61] Davide Cozzolino and Luisa Verdoliva. Noiseprint: a CNN-based camera model fingerprint. *IEEE Transactions on Information Forensics and Security*, 15:144–159, 2019.

[62] Yingqian Cui, Jie Ren, Han Xu, Pengfei He, Hui Liu, Lichao Sun, and Jiliang Tang. DiffusionShield: A watermark for copyright protection against generative diffusion models. *arXiv preprint arXiv:2306.04642*, 2023.

[63] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object detection via region-based fully convolutional networks. *NeurIPS*, 2016.

[64] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[65] Hao Dang, Feng Liu, Joel Stehouwer, Xiaoming Liu, and Anil K Jain. On the detection of digital face manipulation. In *CVPR*, 2020.

[66] Hao Dang, Feng Liu, Joel Stehouwer, Xiaoming Liu, and Anil K Jain. On the detection of digital face manipulation. In *CVPR*, 2020.

[67] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 485–497, 2019.

[68] Neha Dawar and Nasser Kehtarnavaz. Action detection and recognition in continuous action streams by deep learning-based sensing fusion. *IEEE Sensors Journal*, 18(23):9660–9668, 2018.

[69] Debayan Deb, Xiaoming Liu, and Anil Jain. Unified detection of digital and physical face attacks. In *arXiv preprint arXiv:2104.02156*, 2021.

[70] Debayan Deb, Xiaoming Liu, and Anil K Jain. Unified detection of digital and physical face attacks. In *2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG)*, pages 1–8. IEEE, 2023.

[71] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[72] Li Deng. The MNIST database of handwritten digit images for machine learning research [best of the web]. *Signal Processing Magazine*, 29(6):141–142, 2012.

[73] Mohammad Derakhshani, Saeed Masoudnia, Amir Shaker, Omid Mersa, Mohammad Sadeghi, Mohammad Rastegari, and Babak Araabi. Assisted excitation of activations: A learning technique to improve object detectors. In *CVPR*, 2019.

[74] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[75] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

[76] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[77] Jiahua Dong, Wenqi Liang, Hongliu Li, Duzhen Zhang, Meng Cao, Henghui Ding, Salman Khan, and Fahad Shahbaz Khan. How to continually adapt text-to-image diffusion models for flexible customization? *arXiv preprint arXiv:2410.17594*, 2024.

[78] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[79] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[80] Bruce Draper. Reverse engineering of deceptions (red). `https://www.darpa.mil/program/reverse-engineering-of-deceptions`.

[81] Deng-Ping Fan, Ge-Peng Ji, Ming-Ming Cheng, and Ling Shao. Concealed object detection. *TPAMI*, 2021.

[82] Deng-Ping Fan, Ge-Peng Ji, Guolei Sun, Ming-Ming Cheng, Jianbing Shen, and Ling Shao. Camouflaged object detection. In *CVPR*, 2020.

[83] Deng-Ping Fan, Ge-Peng Ji, Tao Zhou, Geng Chen, Huazhu Fu, Jianbing Shen, and Ling Shao. Pranet: Parallel reverse attention network for polyp segmentation. In *MICCAI*, 2020.

[84] Gueter Josmy Faure, Min-Hung Chen, and Shang-Hong Lai. Holistic interaction transformer network for action detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3340–3350, 2023.

[85] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[86] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1933–1941, 2016.

[87] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[88] Weitao Feng, Jiyan He, Jie Zhang, Tianwei Zhang, Wenbo Zhou, Weiming Zhang, and Nenghai Yu. Catch you everything everywhere: Guarding textual inversion via concept watermarking. *arXiv preprint arXiv:2309.05940*, 2023.

[89] Weitao Feng, Wenbo Zhou, Jiyan He, Jie Zhang, Tianyi Wei, Guanlin Li, Tianwei Zhang, Weiming Zhang, and Nenghai Yu. Aqualora: Toward white-box protection for customized stable diffusion models via watermark lora. *arXiv preprint arXiv:2405.11135*, 2024.

[90] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. In *ICCV*, 2023.

[91] Sanja Fidler, Roozbeh Mottaghi, Alan Yuille, and Raquel Urtasun. Bottom-up segmentation for top-down detection. In *CVPR*, 2013.

[92] Tomás Filler, Jessica Fridrich, and Miroslav Goljan. Using sensor pattern noise for camera model identification. In *ICIP*, 2008.

[93] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[94] George Forman and Martin Scholz. Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. *Association for Computing Machinery SIGKDD Explorations Newsletter*, 12(1):49–57, 2010.

[95] Luca Gammaitoni, Peter Hänggi, Peter Jung, and Fabio Marchesoni. Stochastic resonance. *Reviews of modern physics*, 1998.

[96] Rohit Gandikota, Joanna Materzynska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2426–2436, 2023.

[97] Candice R Gerstner and Hany Farid. Detecting real-time deep-fake videos using active illumination. In *CVPR*, 2022.

[98] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *ICCV*, 2015.

[99] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. A better baseline for ava. *arXiv preprint arXiv:1807.10066*, 2018.

[100] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 244–253, 2019.

[101] Rohit Girdhar and Deva Ramanan. Attentional pooling for action recognition. *Advances in neural information processing systems*, 30, 2017.

[102] Ross Girshick. Fast R-CNN. In *ICCV*, 2015.

[103] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[104] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *TPAMI*, 2015.

[105] Georgia Gkioxari and Jitendra Malik. Finding action tubes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 759–768, 2015.

[106] Miroslav Goljan, Jessica Fridrich, and Tomáš Filler. Large scale test of sensor fingerprint camera identification. *Media forensics and security*, 7254:72540I, 2009.

[107] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[108] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[109] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[110] Shreyank N Gowda, Marcus Rohrbach, Frank Keller, and Laura Sevilla-Lara. Learn2augment: learning to composite videos for data augmentation in action recognition. In *European conference on computer vision*, pages 242–259. Springer, 2022.

[111] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017.

[112] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6047–6056, 2018.

[113] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *CVPR*, 2022.

[114] Luca Guarnera, Oliver Giudice, and Sebastiano Battiato. Deepfake detection by analyzing convolutional traces. In *CVPRW*, 2020.

[115] Xiao Guo, Yaojie Liu, Anil Jain, and Xiaoming Liu. Multi-domain learning for updating face anti-spoofing models. In *ECCV*, 2022.

[116] Xiao Guo, Iacopo Masi, Xiaohong Liu, Zhiyuan Ren, Steven Grosz, and Xiaoming Liu. Hierarchical fine-grained image forgery detection and localization. In *CVPR*, 2023.

[117] Agrim Gupta, Piotr Dollár, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019.

[118] Xintong Han, Vlad Morariu, Peng IS Larry Davis, et al. Two-stream neural networks for tampered face detection. In *CVPRW*, 2017.

[119] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018.

[120] Chunming He, Kai Li, Yachao Zhang, Longxiang Tang, Yulun Zhang, Zhenhua Guo, and Xiu Li. Camouflaged object detection with feature decomposition and edge reconstruction. In *CVPR*, 2023.

[121] Chunming He, Kai Li, Yachao Zhang, Guoxia Xu, Longxiang Tang, Yulun Zhang, Zhenhua Guo, and Xiu Li. Weakly-supervised concealed object segmentation with SAM-based pseudo labeling and multi-scale feature grouping. *arXiv preprint arXiv:2305.11003*, 2023.

[122] Chunming He, Kai Li, Yachao Zhang, Yulun Zhang, Zhenhua Guo, Xiu Li, Martin Danelljan, and Fisher Yu. Strategic preys make acute predators: Enhancing camouflaged object detectors by generating camouflaged objects. *arXiv preprint arXiv:2308.03166*, 2023.

[123] Jun-Yan He, Xiao Wu, Zhi-Qi Cheng, Zhaoquan Yuan, and Yu-Gang Jiang. Db-lstm: Densely-connected bi-directional lstm for human action recognition. *Neurocomputing*, 444:319–331, 2021.

[124] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

[125] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *TPAMI*, 2015.

[126] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[127] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[128] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. In *CVPR*, 2019.

[129] Zhenliang He, Wangmeng Zuo, Meina Kan, Shiguang Shan, and Xilin Chen. Attgan: Facial attribute editing by only changing what you want. *IEEE transactions on image processing*, 28:5464–5478, 2019.

[130] Victoria Heath. From a sleazy Reddit post to a national security threat: A closer look at the

deepfake discourse. In *Disinformation and Digital Democracies in the 21st Century*. The NATO Association of Canada, 2019.

[131] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[132] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

[133] Younggi Hong, Min Ju Kim, Isack Lee, and Seok Bong Yoo. Fluxformer: Flow-guided duplex attention transformer via spatio-temporal clustering for action recognition. *IEEE Robotics and Automation Letters*, 2023.

[134] Gangyang Hou, Bo Ou, Min Long, and Fei Peng. Separable reversible data hiding for encrypted 3d mesh models based on octree subdivision and multi-msb prediction. *IEEE Transactions on Multimedia*, 2023.

[135] Jianqin Yin Yanbin Han Wendi Hou and Jinping Li. Detection of the mobile object with camouflage color under dynamic background based on optical flow. *Procedia Engineering*, 2011.

[136] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[137] Weizhe Hua, Zhiru Zhang, and G Edward Suh. Reverse engineering convolutional neural networks through side-channel information leaks. In *DAC*, 2018.

[138] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[139] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.

[140] Yifei Huang, Minjie Cai, Zhenqiang Li, Feng Lu, and Yoichi Sato. Mutual context network for jointly estimating egocentric gaze and action. *IEEE Transactions on Image Processing*, 29:7795–7806, 2020.

[141] Yihao Huang, Felix Juefei-Xu, Qing Guo, Yang Liu, and Geguang Pu. FakeLocator: Robust localization of gan-based face manipulations. *IEEE Transactions on Information Forensics and Security*, 17:2657–2672, 2022.

[142] Thien Huynh-The, Cam-Hao Hua, and Dong-Seong Kim. Encoding pose features to images with data augmentation for 3-d action recognition. *IEEE Transactions on Industrial*

Informatics, 16(5):3100–3111, 2019.

[143] Mobarakol Islam, VS Vibashan, V Jeya Maria Jose, Navodini Wijethilake, Uppal Utkarsh, and Hongliang Ren. Brain tumor segmentation and survival prediction using 3d attention unet. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 5th International Workshop, BrainLes 2019, Held in Conjunction with MICCAI 2019, Shenzhen, China, October 17, 2019, Revised Selected Papers, Part I 5*, pages 262–272. Springer, 2020.

[144] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[145] Abdul Jabbar, Xi Li, and Bourahla Omar. A survey on generative adversarial networks: Variants, applications, and training. *arXiv preprint arXiv:2006.05132*, 2020.

[146] Youngdong Jang, Dong In Lee, MinHyuk Jang, Jong Wook Kim, Feng Yang, and Sangpil Kim. Waterf: Robust watermarks in radiance fields for protection of copyrights. In *CVPR*, 2024.

[147] László A Jeni, Jeffrey F Cohn, and Fernando De La Torre. Facing imbalanced data–recommendations for the use of performance metrics. In *ACII*, 2013.

[148] Hyeonho Jeong, Geon Yeong Park, and Jong Chul Ye. Vmc: Video motion customization using temporal attention adaption for text-to-video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9212–9221, 2024.

[149] Ge-Peng Ji, Deng-Ping Fan, Yu-Cheng Chou, Dengxin Dai, Alexander Liniger, and Luc Van Gool. Deep gradient learning for efficient camouflaged object detection. *Machine Intelligence Research*, 2023.

[150] Ge-Peng Ji, Lei Zhu, Mingchen Zhuge, and Keren Fu. Fast camouflaged object detection via edge-based reversible re-calibration network. *Pattern Recognition*, 123, 2022.

[151] Ruiqi Jiang, Hang Zhou, Weiming Zhang, and Nenghai Yu. Reversible data hiding in encrypted three-dimensional mesh models. *IEEE Transactions on Multimedia*, 2017.

[152] Mei Jiansheng, Li Sukang, and Tan Xiaomei. A digital watermarking algorithm based on DCT and DWT. In *WISA*, 2009.

[153] Amin Jourabloo, Yaojie Liu, and Xiaoming Liu. Face de-spoofing: Anti-spoofing via noise modeling. In *ECCV*, 2018.

[154] Nobukatsu Kajiura, Hong Liu, and Shin'ichi Satoh. Improving camouflaged object detection with the uncertainty of pseudo-edge labels. In *ACM Multimedia Asia*, 2021.

[155] Satoshi Kanai, Hiroaki Date, Takeshi Kishinami, et al. Digital watermarking for 3d polygons using multiresolution wavelet decomposition. In *Proc. Sixth IFIP WG*, volume 5, pages 296–307, 1998.

[156] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

[157] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

[158] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[159] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[160] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[161] Mohammad Ibrahim Khan, Md Maklachur Rahman, and Md Iqbal Hasan Sarker. Digital watermarking for image authentication based on combined DCT, DWT and SVD transformation. *International Journal of Computer Science Issues*, 10:223, 2013.

[162] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. DiffusionCLIP: Text-guided diffusion models for robust image manipulation. In *CVPR*, 2022.

[163] Youngeun Kim, Yuhang Li, Abhishek Moitra, Ruokai Yin, and Priyadarshini Panda. Do we really need a large number of visual prompts? *Neural Networks*, 2024.

[164] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[165] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *ICML*, 2023.

[166] Trupti M Kodinariya and Prashant R Makwana. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013.

[167] Okan Köpüklü, Xiangyu Wei, and Gerhard Rigoll. You only watch once: A unified cnn architecture for real-time spatiotemporal action localization. *arXiv preprint arXiv:1911.06644*, 2019.

[168] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[169] Abhinav Kumar, Garrick Brazil, Enrique Corona, Armin Parchami, and Xiaoming Liu. DEVIANT: Depth EquiVarIAnt NeTwork for Monocular 3D Object Detection. In *ECCV*, 2022.

[170] Abhinav Kumar, Garrick Brazil, and Xiaoming Liu. GrooMeD-NMS: Grouped mathematically differentiable nms for monocular 3D object detection. In *CVPR*, 2021.

[171] Nupur Kumari, Binazeiang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1931–1941, 2023.

[172] Nupur Kumari, Bingliang Zhang, Sheng-Yu Wang, Eli Shechtman, Richard Zhang, and Jun-Yan Zhu. Ablating concepts in text-to-image diffusion models. In *ICCV*, 2023.

[173] Nilakshan Kunananthaseelan, Jing Zhang, and Mehrtash Harandi. Lavip: Language-grounded visual prompting. In *AAAI*, 2024.

[174] Kenji Kurosawa, Kenro Kuroki, and Naoki Saitoh. CCD fingerprint method-identification of a video camera from videotaped images. In *ICIP*, 1999.

[175] Ivan Laptev. On space-time interest points. *International journal of computer vision*, 64:107–123, 2005.

[176] Trung-Nghia Le, Tam Nguyen, Zhongliang Nie, Minh-Triet Tran, and Akihiro Sugimoto. Anabranch network for camouflaged object segmentation. *CVIU*, 2019.

[177] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *ECCV*, 2018.

[178] Aixuan Li, Jing Zhang, Yunqiu Lv, Bowen Liu, Tong Zhang, and Yuchao Dai. Uncertainty-aware joint salient object and camouflaged object detection. In *CVPR*, 2021.

[179] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: Towards deeper understanding of moment matching network. In *NeurIPS*, 2017.

[180] Lingzhi Li, Jianmin Bao, Ting Zhang, Hao Yang, Dong Chen, Fang Wen, and Baining Guo. Face x-ray for more general face forgery detection. In *CVPR*, 2020.

[181] Pan Li, Da Li, Wei Li, Shaogang Gong, Yanwei Fu, and Timothy M Hospedales. A simple feature augmentation for domain generalization. In *ICCV*, 2021.

[182] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4804–4814, 2022.

[183] Yuezun Li and Siwei Lyu. Exposing DeepFake videos by detecting face warping artifacts. In *CVPRW*, 2019.

[184] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, and Siwei Lyu. Celeb-DF: A large-scale challenging dataset for deepfake forensics. In *CVPR*, 2020.

[185] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Light-head R-CNN: In defense of two-stage object detector. *arXiv preprint arXiv:1711.07264*, 2017.

[186] Zhenyang Li, Kirill Gavrilyuk, Efstratios Gavves, Mihir Jain, and Cees GM Snoek. Videolstm convolves, attends and flows for action recognition. *Computer Vision and Image Understanding*, 166:41–50, 2018.

[187] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7083–7093, 2019.

[188] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[189] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[190] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[191] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018.

[192] Honggu Liu, Xiaodan Li, Wenbo Zhou, Yuefeng Chen, Yuan He, Hui Xue, Weiming Zhang, and Nenghai Yu. Spatial-phase shallow learning: rethinking face forgery detection in frequency domain. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 772–781, 2021.

[193] Jiannan Liu, Bo Dong, Shuai Wang, Hui Cui, Deng-Ping Fan, Jiquan Ma, and Geng Chen. Covid-19 lung infection segmentation with a novel two-stage cross-domain transfer learning framework. *Medical image analysis*, 2021.

[194] Ming Liu, Yukang Ding, Min Xia, Xiao Liu, Errui Ding, Wangmeng Zuo, and Shilei Wen. STGAN: A unified selective transfer network for arbitrary image attribute editing. In *CVPR*, 2019.

[195] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation

networks. In *NeurIPS*, 2017.

[196] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[197] Xiaohong Liu, Yaojie Liu, Jun Chen, and Xiaoming Liu. PSCC-Net: Progressive spatio-channel correlation network for image manipulation detection and localization. In *arXiv preprint arXiv:2103.10596*, 2021.

[198] Yugeng Liu, Zheng Li, Michael Backes, Yun Shen, and Yang Zhang. Watermarking diffusion model. *arXiv preprint arXiv:2305.12502*, 2023.

[199] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[200] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[201] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *CVPR*, 2022.

[202] Jan Lukáš, Jessica Fridrich, and Miroslav Goljan. Detecting digital image forgeries using sensor pattern noise. *Security, Steganography, and Watermarking of Multimedia Contents VIII*, 6072:60720Y, 2006.

[203] Jan Lukas, Jessica Fridrich, and Miroslav Goljan. Digital camera identification from sensor pattern noise. *IEEE Transactions on Information Forensics and Security*, 1(2):205–214, 2006.

[204] Yunqiu Lv, Jing Zhang, Yuchao Dai, Aixuan Li, Bowen Liu, Nick Barnes, and Deng-Ping Fan. Simultaneously localize, segment and rank the camouflaged objects. In *CVPR*, 2021.

[205] Kang Ma, Ying Fu, Chunshui Cao, Saihui Hou, Yongzhen Huang, and Dezhi Zheng. Learning visual prompt for gait recognition. In *CVPR*, 2024.

[206] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[207] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[208] Yuxin Mao, Jing Zhang, Zhexiong Wan, Yuchao Dai, Aixuan Li, Yunqiu Lv, Xinyu Tian, Deng-Ping Fan, and Nick Barnes. Transformer transforms salient object detection and

camouflaged object detection. *arXiv preprint arXiv:2104.10127*, 2021.

[209] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. Do GANs leave artificial fingerprints? In *MIPR*, 2019.

[210] Francesco Marra, Cristiano Saltori, Giulia Boato, and Luisa Verdoliva. Incremental learning for the detection and classification of GAN-generated images. In *WIFS*, 2019.

[211] Iacopo Masi, Aditya Killekar, Royston Marian Mascarenhas, Shenoy Pratik Gurudatt, and Wael AbdAlmageed. Two-branch recurrent network for isolating deepfakes in videos. In *ECCV*. Springer, 2020.

[212] Falko Matern, Christian Riess, and Marc Stamminger. Exploiting visual artifacts to expose deepfakes and face manipulations. In *WACVW*, 2019.

[213] Scott McCloskey and Michael Albright. Detecting GAN-generated imagery using saturation cues. In *ICIP*, 2019.

[214] Safa C. Medin, Bernhard Egger, Anoop Cherian, Ye Wang, Joshua B. Tenenbaum, Xiaoming Liu, and Tim K. Marks. MOST-GAN: 3D morphable StyleGAN for disentangled face image manipulation. In *AAAI*, 2022.

[215] Haiyang Mei, Ge-Peng Ji, Ziqi Wei, Xin Yang, Xiaopeng Wei, and Deng-Ping Fan. Camouflaged object segmentation with distraction mining. In *CVPR*, 2021.

[216] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *CVPR*, 2023.

[217] Lili Meng, Bo Zhao, Bo Chang, Gao Huang, Wei Sun, Frederick Tung, and Leonid Sigal. Interpretable spatio-temporal attention for video action recognition. In *Proceedings of the IEEE/CVF international conference on computer vision workshops*, pages 0–0, 2019.

[218] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

[219] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6038–6047, 2023.

[220] Todd K Moon. The expectation-maximization algorithm. *Signal processing magazine*, 13(6):47–60, 1996.

[221] Travis Munyer and Xin Zhong. Deeptextmark: Deep learning based text watermarking for detection of large language model generated text. *arXiv preprint*, 2023.

[222] Lakshmanan Nataraj, Tajuddin Manhar Mohammed, BS Manjunath, Shivkumar Chandrasekaran, Arjuna Flenner, Jawadul H Bappy, and Amit K Roy-Chowdhury. Detecting GAN generated fake images using co-occurrence matrices. *Electronic Imaging*, 2019:532–1, 2019.

[223] Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi. Image colorization using generative adversarial networks. In *AMDO*, 2018.

[224] Eric Nguyen, Tu Bui, Vishy Swaminathan, and John Collomosse. OSCAR-Net: Object-centric scene graph attention for image attribution. In *ICCV*, 2021.

[225] Huy H Nguyen, Fuming Fang, Junichi Yamagishi, and Isao Echizen. Multi-task learning for detecting and segmenting manipulated facial images and videos. In *BTAS*, 2019.

[226] Huy H Nguyen, Fuming Fang, Junichi Yamagishi, and Isao Echizen. Multi-task learning for detecting and segmenting manipulated facial images and videos. In *BTAS*, 2019.

[227] Huy H Nguyen, Junichi Yamagishi, and Isao Echizen. Capsule-forensics: Using capsule networks to detect forged images and videos. In *ICASSP*, 2019.

[228] Yuval Nirkin, Iacopo Masi, Anh Tran Tuan, Tal Hassner, and Gerard Medioni. On face segmentation, face swapping, and face perception. In *FGR*, pages 98–105. IEEE, 2018.

[229] Yuval Nirkin, Lior Wolf, Yosi Keller, and Tal Hassner. Deepfake detection based on the discrepancy between the face and its context. *arXiv preprint arXiv:2008.12262*, 2020.

[230] Yuval Nirkin, Lior Wolf, Yosi Keller, and Tal Hassner. Deepfake detection based on discrepancies between faces and their context. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 2021.

[231] Kento Nishi, Yi Ding, Alex Rich, and Tobias Hollerer. Augmentation strategies for learning with noisy labels. In *CVPR*, 2021.

[232] Ori Nizan and Ayellet Tal. Breaking the cycle - colleagues are all you need. In *CVPR*.

[233] Seong Joon Oh, Max Augustin, Mario Fritz, and Bernt Schiele. Towards reverse-engineering black-box neural networks. In *ICLR*, 2018.

[234] Ryutarou Ohbuchi, Hiroshi Masuda, and Masaki Aono. Watermarking three-dimensional polygonal models through geometric and topological modifications. *IEEE Journal on selected areas in communications*, 16(4):551–560, 1998.

[235] Junting Pan, Ziyi Lin, Xiatian Zhu, Jing Shao, and Hongsheng Li. St-adapter: Parameter-efficient image-to-video transfer learning. *Advances in Neural Information Processing Systems*, 35:26462–26477, 2022.

[236] Yuxin Pan, Yiwang Chen, Qiang Fu, Ping Zhang, and Xin Xu. Study on the camouflaged target detection method based on 3D convexity. *Modern Applied Science*, 2011.

[237] Sungho Park and Hyeran Byun. Fair-vpt: Fair visual prompt tuning for image classification. In *CVPR*, 2024.

[238] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. GauGAN: semantic image synthesis with spatially adaptive normalization. In *ACM*, 2019.

[239] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[240] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[241] Sen Peng, Yufei Chen, Cong Wang, and Xiaohua Jia. Protecting the intellectual property of diffusion models by the watermark diffusion process. *arXiv preprint arXiv:2306.03436*, 2023.

[242] Xiaojiang Peng and Cordelia Schmid. Multi-region two-stream r-cnn for action detection. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 744–759. Springer, 2016.

[243] Yuxin Peng, Yunzhen Zhao, and Junchao Zhang. Two-stream collaborative learning with spatial-temporal attention for video classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(3):773–786, 2018.

[244] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *ICML*, 2018.

[245] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *CVPR*, 2020.

[246] Ed Pizzi, Sreya Dutta Roy, Sugosh Nagavara Ravindra, Priya Goyal, and Matthijs Douze. A self-supervised descriptor for image copy detection. In *CVPR*, 2022.

[247] Ryan Po, Guandao Yang, Kfir Aberman, and Gordon Wetzstein. Orthogonal adaptation for modular customization of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7964–7973, 2024.

[248] Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. Ganimation: Anatomically-aware facial animation from a single image. In

ECCV, 2018.

[249] Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. GANimation: One-shot anatomically consistent facial animation. *International Journal of Computer Vision*, 128:698–713, 2020.

[250] Yuyang Qian, Guojun Yin, Lu Sheng, Zixuan Chen, and Jing Shao. Thinking in frequency: Face forgery detection by mining frequency-aware clues. In *ECCV*, 2020.

[251] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[252] Arezoo Rajabi, Rakesh B Bobba, Mike Rosulek, Charles Wright, and Wu-chi Feng. On the (im) practicality of adversarial perturbation for image privacy. *PETS*, 2021.

[253] VP Subramanyam Rallabandi and Prasun Kumar Roy. Magnetic resonance image enhancement using stochastic resonance in fourier domain. *Magnetic resonance imaging*, 2010.

[254] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[255] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *CVPR*, 2017.

[256] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[257] Atique Rehman, Rafia Rahim, Shahroz Nadeem, and Sibt Hussain. End-to-end trained CNN encoder-decoder networks for image steganography. In *ECCVW*, 2018.

[258] Atique-ur Rehman, Rafia Rahim, Shahroz Nadeem, and Sibt-ul Hussain. End-to-end trained CNN encoder-decoder networks for image steganography. In *ECCVW*, 2019.

[259] Jingjing Ren, Xiaowei Hu, Lei Zhu, Xuemiao Xu, Yangyang Xu, Weiming Wang, Zijun Deng, and Pheng-Ann Heng. Deep texture-aware features for camouflaged object detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[260] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015.

[261] Yuhao Ren, Fabing Duan, François Chapeau-Blondeau, and Derek Abbott. Self-gating stochastic-resonance-based autoencoder for unsupervised learning. *Physical Review E*, 2024.

[262] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data:

Ground truth from computer games. In *ECCV*, 2016.

[263] Anna Rogers. The attribution problem with generative ai. *Hacking Semantics*, 2022.

[264] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[265] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. In *CVPR*, 2019.

[266] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. FaceForensics++: Learning to detect manipulated facial images. In *ICCV*, 2019.

[267] Nataniel Ruiz, Sarah Adel Bargal, and Stan Sclaroff. Disrupting deepfakes: Adversarial attacks against conditional image translation networks and facial manipulation systems. In *ECCV*, 2020.

[268] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510, 2023.

[269] Dan Ruta, Saeid Motiian, Baldo Faieta, Zhe Lin, Hailin Jin, Alex Filipkowski, Andrew Gilbert, and John Collomosse. ALADIN: All layer adaptive instance normalization for fine-grained style similarity. In *ICCV*, 2021.

[270] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Radioactive data: tracing through training. In *ICML*, 2020.

[271] Suman Saha, Gurkirt Singh, Michael Sapienza, Philip HS Torr, and Fabio Cuzzolin. Deep learning for detecting multiple space-time action tubes in videos. *arXiv preprint arXiv:1608.01529*, 2016.

[272] Eran Segalis and Eran Galili. OGAN: Disrupting deepfakes with an adversarial attack that survives training. *arXiv preprint arXiv:2006.12247*, 2020.

[273] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch, August 2020. Version 0.3.0.

[274] Vladimir V Semenov and Anna Zakharova. Multiplexing-based control of stochastic resonance. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 2022.

[275] P Sengottuvelan, Amitabh Wahi, and A Shanmugam. Performance of decamouflaging through exploratory image analysis. In *ICETET*, 2008.

[276] Shawn Shan, Jenna Cryan, Emily Wenger, Haitao Zheng, Rana Hanocka, and Ben Y Zhao. Glaze: Protecting artists from style mimicry by {Text-to-Image} models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 2187–2204, 2023.

[277] Mengen Shen, Jianhua Yang, Wenbo Jiang, Miguel AF Sanjuan, and Yuqiao Zheng. Stochastic resonance in image denoising as an alternative to traditional methods and deep learning. *Nonlinear Dynamics*, 2022.

[278] Jing Shi, Wei Xiong, Zhe Lin, and Hyun Joon Jung. Instantbooth: Personalized text-to-image generation without test-time finetuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8543–8552, 2024.

[279] Kaede Shiohara and Toshihiko Yamasaki. Detecting deepfakes with self-blended images. In *CVPR*, 2022.

[280] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in neural information processing systems*, 27, 2014.

[281] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[282] Amit Kumar Singh, Nomit Sharma, Mayank Dave, and Anand Mohan. A novel technique for digital image watermarking in spatial domain. In *PDGC*, 2012.

[283] Suriya Singh, Chetan Arora, and CV Jawahar. First person action recognition using deep learned descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2620–2628, 2016.

[284] Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America*, 4(3):519–524, 1987.

[285] Kihyuk Sohn, Huiwen Chang, José Lezama, Luisa Polania, Han Zhang, Yuan Hao, Irfan Essa, and Lu Jiang. Visual prompt tuning for generative transfer learning. In *CVPR*, 2023.

[286] Gowthami Somepalli, Anubhav Gupta, Kamal Gupta, Shramay Palta, Micah Goldblum, Jonas Geiping, Abhinav Shrivastava, and Tom Goldstein. Measuring style similarity in diffusion models. *arXiv preprint arXiv:2404.01292*, 2024.

[287] Kritaphat Songsri-in and Stefanos Zafeiriou. Complement face forensic detection and localization with facial landmarks. *arXiv preprint arXiv:1910.05455*, 2019.

[288] Anil K Srivastava, Virendra K Srivastava, and Aman Ullah. The coefficient of determination and its adjusted version in linear regression models. *Econometric reviews*, 14(2):229–240, 1995.

[289] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Kevin Murphy, Rahul Sukthankar, and Cordelia Schmid. Actor-centric relation network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 318–334, 2018.

[290] Lin Sun, Kui Jia, Kevin Chen, Dit-Yan Yeung, Bertram E Shi, and Silvio Savarese. Lattice long short-term memory for human action recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 2147–2156, 2017.

[291] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, and Ping Luo. Sparse R-CNN: End-to-end object detection with learnable proposals. In *CVPR*, 2021.

[292] Yujia Sun, Geng Chen, Tao Zhou, Yi Zhang, and Nian Liu. Context-aware cross-level fusion network for camouflaged object detection. *arXiv preprint arXiv:2105.12555*, 2021.

[293] Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N Asokan. Dawn: Dynamic adversarial watermarking of neural networks. In *ACM-MM*, 2021.

[294] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. MnasNet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019.

[295] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[296] Wei Ren Tan, Chee Seng Chan, Hernan Aguirre, and Kiyoshi Tanaka. Improved ArtGAN for conditional synthesis of natural image and artwork. *IEEE Transactions on Image Processing*, 28(1):394–409, 2019.

[297] Matthew Tancik, Ben Mildenhall, and Ren Ng. StegaStamp: Invisible hyperlinks in physical photographs. In *CVPR*, 2020.

[298] Li Tang, Qingqing Ye, Haibo Hu, Qiao Xue, Yaxin Xiao, and Jin Li. Deepmark: A scalable and robust framework for deepfake video detection. *ACM Transactions on Privacy and Security*, 2024.

[299] Long Tang, Dengpan Ye, Yunna Lv, Chuanxi Chen, and Yunming Zhang. Once and for all: Universal transferable adversarial perturbation against deep hashing-based facial image retrieval. In *AAAI*, 2024.

[300] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *USENIXSS*, 2016.

[301] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE*

international conference on computer vision, pages 4489–4497, 2015.

[302] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[303] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning GAN for pose-invariant face recognition. In *CVPR*, 2017.

[304] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *CVPR*, 2017.

[305] Yuan-Yu Tsai and Hong-Lin Liu. Integrating coordinate transformation and random sampling into high-capacity reversible data hiding in encrypted polygonal models. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[306] Radim Tyleček and Radim Šára. Spatial pattern templates for recognition of objects with regular structure. In *GCPR*, 2013.

[307] Amin Ullah, Jamil Ahmad, Khan Muhammad, Muhammad Sajjad, and Sung Wook Baik. Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE access*, 6:1155–1166, 2017.

[308] Diego Valsesia, Giulio Coluccia, Tiziano Bianchi, and Enrico Magli. Compressed fingerprint matching and camera identification via random projections. *IEEE Transactions on Information Forensics and Security*, 10(7):1472–1485, 2015.

[309] Thanh Van Le, Hao Phung, Thuan Hoang Nguyen, Quan Dao, Ngoc N Tran, and Anh Tran. Anti-dreambooth: Protecting users from personalized text-to-image synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2116–2127, 2023.

[310] Ashish Vaswani. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[311] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[312] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[313] Paul Viola and Michael Jones. Robust real-time face detection. *IJCV*, 2004.

[314] Christoffer Waldemarsson. *Disinformation, Deepfakes & Democracy; The European response to election interference in the digital age*. The Alliance of Democracies Foundation, 2020.

[315] Cheng Wang, Haojin Yang, and Christoph Meinel. Exploring multimodal video representation for action recognition. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1924–1931. IEEE, 2016.

[316] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *CVPR*, 2023.

[317] Feifei Wang, Zhentao Tan, Tianyi Wei, Yue Wu, and Qidong Huang. Simac: A simple anti-customization method for protecting face privacy against text-to-image synthesis of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12047–12056, 2024.

[318] Heng Wang and A Kl. aser, c. schmid, and c.-l. liu,"action recognition by dense trajectories,". In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*, pages 3169–3176, 2011.

[319] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE international conference on computer vision*, pages 3551–3558, 2013.

[320] Jiangfeng Wang, Hanzhou Wu, Xinpeng Zhang, and Yuwei Yao. Watermarking in deep neural networks via error back-propagation. *Electronic Imaging*, 2020.

[321] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *TPAMI*, 2020.

[322] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4305–4314, 2015.

[323] Limin Wang, Yuanjun Xiong, Zhe Wang, and Yu Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015.

[324] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016.

[325] Run Wang, Felix Juefei-Xu, Meng Luo, Yang Liu, and Lina Wang. FakeTagger: Robust safeguards against deepfake dissemination via provenance tracking. In *ACMM*, 2021.

[326] Run Wang, Felix Juefei-Xu, Lei Ma, Xiaofei Xie, Yihao Huang, Jian Wang, and Yang Liu. FakeSpotter: A simple yet robust baseline for spotting ai-synthesized fake faces. In *IJCAI*, 2020.

[327] Sheng-Yu Wang, David Bau, and Jun-Yan Zhu. Sketch your own gan. In *ICCV*, 2021.

[328] Sheng-Yu Wang, Alexei A Efros, Jun-Yan Zhu, and Richard Zhang. Evaluating data attribution for text-to-image models. In *ICCV*, 2023.

[329] Sheng-Yu Wang, Aaron Hertzmann, Alexei A Efros, Jun-Yan Zhu, and Richard Zhang. Data attribution for text-to-image models by unlearning synthesized images. *arXiv preprint arXiv:2406.09408*, 2024.

[330] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. CNN-generated images are surprisingly easy to spot... for now. In *CVPR*, 2020.

[331] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. CNN-generated images are surprisingly easy to spot... for now. In *CVPR*, 2020.

[332] Xiaogang Wang and Xiaoou Tang. Face photo-sketch synthesis and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31:1955–1967, 2008.

[333] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-ESRGAN: Training real-world blind super-resolution with pure synthetic data. In *CVPR*, 2021.

[334] Yunbo Wang, Mingsheng Long, Jianmin Wang, and Philip S Yu. Spatiotemporal pyramid network for video action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1529–1538, 2017.

[335] Zhengwei Wang, Qi She, and Tomás E. Ward. Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Computing Surveys*, 54(2), 2021.

[336] Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Learning to track for spatio-temporal action localization. In *Proceedings of the IEEE international conference on computer vision*, pages 3164–3172, 2015.

[337] Michael J Wilber, Chen Fang, Hailin Jin, Aaron Hertzmann, John Collomosse, and Serge Belongie. BAM! the behance artistic media dataset for recognition beyond photography. In *ICCV*, 2017.

[338] Adrian Wolny, Lorenzo Cerrone, Athul Vijayan, Rachele Tofanelli, Amaya Vilches Barro, Marion Louveaux, Christian Wenzl, Sören Strauss, David Wilson-Sánchez, Rena Lymbouridou, Susanne S Steigleder, Constantin Pape, Alberto Bailoni, Salva Duran-Nebreda, George W Bassel, Jan U Lohmann, Miltos Tsiantis, Fred A Hamprecht, Kay Schneitz, Alexis Maizel, and Anna Kreshuk. Accurate and versatile 3d segmentation of plant tissues at cellular resolution. *eLife*, 9:e57613, jul 2020.

[339] Di Wu, Junjun Chen, Nabin Sharma, Shirui Pan, Guodong Long, and Michael Blumenstein. Adversarial action data augmentation for similar gesture action recognition. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[340] Xi Wu, Zhen Xie, YuTao Gao, and Yu Xiao. SSTNET: Detecting manipulated faces through spatial, steganalysis and temporal features. In *ICASSP*, 2020.

[341] Xiaoshuai Wu, Xin Liao, and Bo Ou. Sepmark: Deep separable watermarking for unified source tracing and deepfake detection. *arXiv preprint*, 2023.

[342] Xiaoshuai Wu, Xin Liao, Bo Ou, Yuling Liu, and Zheng Qin. Are watermarks bugs for deepfake detectors? rethinking proactive forensics. *arXiv preprint*, 2024.

[343] Zihao Xiao, Xianfeng Gao, Chilin Fu, Yinpeng Dong, Wei Gao, Xiaolu Zhang, Jun Zhou, and Jun Zhu. Improving transferability of adversarial patches on face recognition with generative models. In *CVPR*, 2021.

[344] Chu Xin, Seokhwan Kim, and Kyoung Shin Park. A comparison of machine learning models with data augmentation techniques for skeleton-based human action recognition. In *Proceedings of the 14th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 1–6, 2023.

[345] Ying Xu, Kiran Raja, and Marius Pedersen. Supervised contrastive learning for generalizable and explainable deepfakes detection. In *WACV*, 2022.

[346] Jian-Ru Xue, Jian-Wu Fang, and Pu Zhang. A survey of scene understanding by event reasoning in autonomous driving. *International Journal of Automation and Computing*, 2018.

[347] Shen Yan, Xuehan Xiong, Anurag Arnab, Zhichao Lu, Mi Zhang, Chen Sun, and Cordelia Schmid. Multiview transformers for video recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3333–3343, 2022.

[348] Fan Yang, Qiang Zhai, Xin Li, Rui Huang, Ao Luo, Hong Cheng, and Deng-Ping Fan. Uncertainty-guided transformer reasoning for camouflaged object detection. In *ICCV*, 2021.

[349] Jiewen Yang, Xingbo Dong, Liujun Liu, Chao Zhang, Jiajun Shen, and Dahai Yu. Recurring the transformer for video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14063–14073, 2022.

[350] Xin Yang, Yuezun Li, and Siwei Lyu. Exposing deep fakes using inconsistent head poses. In *ICASSP*, 2019.

[351] Yang Yang, Wen Wang, Liang Peng, Chaotian Song, Yao Chen, Hengjia Li, Xiaolong Yang, Qinglin Lu, Deng Cai, Boxi Wu, et al. Lora-composer: Leveraging low-rank adaptation for multi-concept customization in training-free diffusion models. *arXiv preprint arXiv:2403.11627*, 2024.

[352] Leiyue Yao, Wei Yang, and Wei Huang. A data augmentation method for human action

recognition using dense joint motion images. *Applied Soft Computing*, 97:106713, 2020.

[353] Yuguang Yao, Yifan Gong, Yize Li, Yimeng Zhang, Xue Lin, and Sijia Liu. Reverse engineering of imperceptible adversarial image perturbations. In *ICLR*, 2022.

[354] Yuguang Yao, Xiao Guo, Vishal Asnani, Yifan Gong, Jiancheng Liu, Xue Lin, Xiaoming Liu, and Sijia Liu. Reverse engineering of deceptions on machine- and human-centric attacks. *Foundations and Trends in Privacy and Security*, 2024.

[355] Erkan Yavuz and Ziya Telatar. Improved SVD-DWT based digital image watermarking against watermark ambiguity. In *SAC*, 2007.

[356] Chin-Yuan Yeh, Hsi-Wen Chen, Shang-Lun Tsai, and Sheng-De Wang. Disrupting image-translation-based deepfake algorithms with adversarial attacks. In *WACVW*, 2020.

[357] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. DualGAN: Unsupervised dual learning for image-to-image translation. In *CVPR*, 2017.

[358] Dong Yin, Raphael Gontijo Lopes, Jon Shlens, Ekin Dogus Cubuk, and Justin Gilmer. A fourier perspective on model robustness in computer vision. *NeurIPS*, 2019.

[359] Innfarn Yoo, Huiwen Chang, Xiyang Luo, Ondrej Stava, Ce Liu, Peyman Milanfar, and Feng Yang. Deep 3d-to-2d watermarking: Embedding messages in 3d meshes and extracting them from 2d renderings. In *CVPR*, 2022.

[360] Masakazu Yoshimura, Junji Otsuka, Atsushi Irie, and Takeshi Ohashi. Rawgment: Noise-accounted raw augmentation enables recognition in a wide variety of environments. In *CVPR*, 2023.

[361] A. Yu and K. Grauman. Fine-grained visual comparisons with local learning. In *CVPR*, 2014.

[362] A. Yu and K. Grauman. Semantic jitter: Dense supervision for visual comparisons via synthetic images. In *ICCV*, 2017.

[363] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[364] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[365] Ning Yu, Larry S Davis, and Mario Fritz. Attributing fake images to GANs: Learning and analyzing GAN fingerprints. In *ICCV*, 2019.

[366] Ziyang Yuan, Mingdeng Cao, Xintao Wang, Zhongang Qi, Chun Yuan, and Ying Shan. Customnet: Zero-shot object customization with variable-viewpoints in text-to-image diffusion models. *arXiv preprint arXiv:2310.19784*, 2023.

[367] Matthew Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[368] Qiang Zhai, Xin Li, Fan Yang, Chenglizhao Chen, Hong Cheng, and Deng-Ping Fan. Mutual graph learning for camouflaged object detection. In *CVPR*, 2021.

[369] Jie Zhang, Dongdong Chen, Jing Liao, Weiming Zhang, Huamin Feng, Gang Hua, and Nenghai Yu. Deep model intellectual property protection via deep watermarking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[370] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[371] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in GAN fake images. In *WIFS*, 2019.

[372] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in GAN fake images. In *WIFS*, 2019.

[373] Yimeng Zhang, Xin Chen, Jinghan Jia, Sijia Liu, and Ke Ding. Text-visual prompting for efficient 2d temporal video grounding. In *CVPR*, 2023.

[374] Yushu Zhang, Jiahao Zhu, Mingfu Xue, Xinpeng Zhang, and Xiaochun Cao. Adaptive 3d mesh steganography based on feature-preserving distortion. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[375] Yuxin Zhang, Nisha Huang, Fan Tang, Haibin Huang, Chongyang Ma, Weiming Dong, and Changsheng Xu. Inversion-based style transfer with diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10146–10156, 2023.

[376] Jiaojiao Zhao, Yanyi Zhang, Xinyu Li, Hao Chen, Bing Shuai, Mingze Xu, Chunhui Liu, Kaustav Kundu, Yuanjun Xiong, Davide Modolo, et al. Tuber: Tubelet transformer for video action detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13598–13607, 2022.

[377] Mingjun Zhao, Yakun Yu, Xiaoli Wang, Lei Yang, and Di Niu. Search-map-search: a frame selection paradigm for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10627–10636, 2023.

[378] Xuandong Zhao, Yu-Xiang Wang, and Lei Li. Protecting language generation models via invisible watermarking. *arXiv preprint*, 2023.

[379] Xuandong Zhao, Kexun Zhang, Zihao Su, Saastha Vasan, Ilya Grishchenko, Christopher Kruegel, Giovanni Vigna, Yu-Xiang Wang, and Lei Li. Invisible image watermarks are provably removable using generative AI. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[380] Yunqing Zhao, Tianyu Pang, Chao Du, Xiao Yang, Ngai-Man Cheung, and Min Lin. A recipe for watermarking diffusion models. *arXiv preprint arXiv:2303.10137*, 2023.

[381] Zhengyue Zhao, Jinhao Duan, Kaidi Xu, Chenan Wang, Rui Zhang, Zidong Du, Qi Guo, and Xing Hu. Can protective perturbation safeguard personal data from being exploited by stable diffusion? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24398–24407, 2024.

[382] Yuan Zhi, Zhan Tong, Limin Wang, and Gangshan Wu. Mgsampler: An explainable sampling strategy for video action recognition. In *Proceedings of the IEEE/CVF International conference on Computer Vision*, pages 1513–1522, 2021.

[383] Yaoyao Zhong and Weihong Deng. Towards transferable adversarial attack against deep face recognition. *IEEE Transactions on Information Forensics and Security*, 2020.

[384] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. HiDDeN: Hiding data with deep networks. In *ECCV*, 2018.

[385] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[386] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NeurIPS*, 2017.

[387] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. SEAN: Image synthesis with semantic region-adaptive normalization. In *CVPR*, 2020.

[388] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *ICLR*, 2020.

[389] Mingchen Zhuge, Xiankai Lu, Yiyou Guo, Zhihua Cai, and Shuhan Chen. CubeNet: X-shape connection for camouflaged object detection. *Pattern Recognition*, 2022.

APPENDIX A

PUBLICATIONS

A list of all peer-reviewed publications during the MSU Ph.D. program listed chronologically.

- *Vishal Asnani*, Xi Yin, Tal Hassner, Sijia Liu, and Xiaoming Liu. "Proactive image manipulation detection." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022.
- *Vishal Asnani*, Xi Yin, Tal Hassner, and Xiaoming Liu. "Malp: Manipulation localization using a proactive scheme." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023.
- *Vishal Asnani*, Xi Yin, Tal Hassner, and Xiaoming Liu. "Reverse engineering of generative models: Inferring model hyperparameters from generated images." IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023.
- *Vishal Asnani*, Abhinav Kumar, Suya You, and Xiaoming Liu. "PrObeD: proactive object detection wrapper." Advances in Neural Information Processing Systems 36, 2024.
- *Vishal Asnani*, John Collomosse, Tu Bui, Xiaoming Liu, and Shruti Agarwal. "ProMark: Proactive Diffusion Watermarking for Causal Attribution." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024.
- *Vishal Asnani*, John Collomosse, Xiaoming Liu, and Shruti Agarwal. "CustomMark: Customization of Diffusion Models for Proactive Attribution." In Review, 2025.
- *Vishal Asnani*, Xiaoming Liu, and Shruti Agarwal. "PiVoT: Proactive Video Templates for Enhancing Video Task Performance." In Review, 2025.

APPENDIX B

PROACTIVE IMAGE MANIPULATION DETECTION APPENDIX

B.1 Cross Encoder-Template Set Evaluation

Our framework encrypts a real image using a template from the template set. This encryption would aid in the image manipulation detection if the image is corrupted by any unseen GM. The framework is divided in two stages namely, image encryption and recovery of template where each stage works independently in inference. We therefore provide an ablation to study the performance using different encoder and template set, *i.e.*, we evaluate recovering ability of an encoder using a template set trained with different initialization seeds. The results are shown in Tab. B.1. We observe that even though the template set and the encoder are initialized with different seeds, the performance of our framework doesn't vary much. This shows the stability of our framework even though the initialization seeds of both stages during training are different.

B.2 Template Strength

We provide the ablation for hyperparameter m used to control the strength of the added template in Sec. 4.3. We observe that the performance is better if we increase the template strength. However, this comes at a trade-off with PSNR which declines if the template strength increases. This is also justified in Fig. B.1 which shows the images with different strength of added template. The images become noisier as the template strength is increased. This is not desirable as there shouldn't be much distortion in the encrypted real image due to our added template. Therefore for our experiments, we select 30% as the strength for the added template.

B.3 Implementation Details

Image editing techniques We use various image editing techniques in Sec. 4.2. All the techniques are applied after addition of our template. We provide the implementation details for all these techniques below:

1. Blur: We apply Gaussian blur to the image with 50% probability using σ sampled from $[0, 3]$,

Table B.1 Cross encoder-template set evaluation with different initialization seeds.

Initialization seed		Test GM Average precision (%)		
Encoder	Template set	StarGAN	CycleGAN	GauGAN
1	1	96.12	100	91.62
	2	94.65	100	91.15
	3	94.83	100	91.46
2	1	95.48	100	91.56
	2	95.54	100	90.85
	3	95.84	100	91.06
3	1	95.56	100	91.32
	2	95.62	100	91.42
	3	96.14	100	90.41

2. JPEG: We JPEG-compress the image with 50% probability images using Imaging Library (PIL), with quality sampled from Uniform$\{30, 31, ..., 100\}$.

3. Blur + JPEG (p): The image is possibly blurred and JPEG-compressed, each with probability p.

4. Resizing: We perform the training using 50% of the images with $256 \times 256 \times 3$ resolution and rest with $128 \times 128 \times 3$ resolution images in CelebA-HQ dataset.

5. Crop: We randomly crop the images with 50% probability on each side with pixels sampled from $[0, 30]$. The images are resized to $128 \times 128 \times 3$ resolution.

6. Gaussian noise: We add Gaussian noise with zero mean and unit variance to the images with 50% probability.

Figure B.1 Visualization of input images with different template strength. As the template strength is increased, the images become noisier.

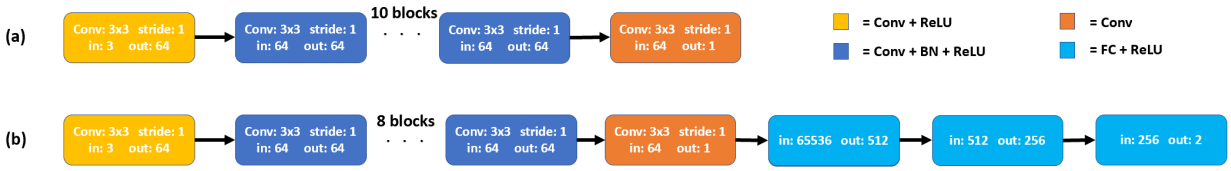


Figure B.2 Network architecture for our (a) encoder (b) classifier network for image manipulation detection.

Table B.2 List of GMs with their datasets and input image resolution used for evaluating our framework's generalization ability.

GM	STGAN [194]	StarGAN [52]	CycleGAN [385]	GauGAN [238]	UNIT [195]	MUNIT [139]	StarGAN2 [53]	BicycleGAN [386]
Dataset	CelebA-HQ [157]	CelebA-HQ [157]	Facades [306]	COCO [30]	GTA2City [262]	Edges2Shoes [361, 362]	CelebA-HQ [157]	Facades [306]
Resolution	$128 \times 128 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$512 \times 931 \times 3$	$256 \times 512 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$
GM	CONT_Encoder [240]	SEAN [387]	ALAE[245]	Pix2Pix[144]	DualGAN[357]	CouncilGAN[232]	ESRGAN[333]	GANimation[249]
Dataset	Paris Street-View [232]	CelebA-HQ [157]	CelebA-HQ [157]	Facades [306]	Sketch-Photo [332]	CelebA [200]	CelebA [200]	CelebA [200]
Resolution	$64 \times 64 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$256 \times 256 \times 3$	$128 \times 128 \times 3$	$128 \times 128 \times 3$

Network architecture Fig. B.2 shows the network architecture used in different experiments for our framework's evaluation. For our framework, our encoder has 2 stem convolution layers and 10 convolution blocks to recover the added template from encrypted real images. Each block comprises of convolution, batch normalization and ReLU activation.

In ablation experiments for Table 8, we use a classification network with the similar number of layers as our encoder. This is done to show the importance of recovering templates using encoder. This classification networks has 8 convolution blocks followed by three fully connected layers with ReLU activation in between the layers. The network outputs 2 dimension logits used for image manipulation detection.

B.4 List of GMs

We use a variety of GMs to test the generalization ability of our framework. These GMs have varied network architectures and many of them are trained on different datasets. We summarize all the GMs in Tab. B.2. We also provide visualization for different real image samples used in evaluating the performance for all these GMs in Fig. B.3 - Fig. B.18. We show the added template and the recovered templates in "gist_rainbow" cmap for better visualization and indicate the cosine similarity of the recovered template with the added template. As shown in Fig. B.3 for training with STGAN, the encrypted real images have higher cosine similarity compared to their manipulated counterparts. However, during testing, the difference between the two cosine similarities decreases as shown in Fig. B.4 - Fig. B.18 for different GMs.

B.5 Dataset License Information

We use diverse datasets for our experiments which include face and non-face datasets. For face datasets, we use existing datasets including CelebA [200] and CelebA-HQ [157]. The CelebA dataset contains images entirely from the internet and has no associated IRB approval. The authors mention that the dataset is available for non-commercial research purposes only, which we strictly adhere to. We only use the database internally for our work and primarily for evaluation. CelebA-HQ consists images collected from the internet. Although there is no associated IRB approval, the authors assert in the dataset agreement that the dataset is only to be used for non-commercial

research purposes, which we strictly adhere to.

We use some non-face datasets too for our experiments. The Facades [306] dataset was collected at the Center for Machine Perception and is provided under Attribution-ShareAlike license. Edges2Shoes [361, 362] is a large shoe dataset consisting of images collected from `https://www.zappos.com`. The authors mention that this dataset is for academic, non-commercial use only. GTA2City [262] dataset consists of a large number of densely labelled frames extracted from computer games. The authors mention that the data is for research and educational use only. The sketch-photo [332] datset refers to the CUHK face sketch FERET database. The authors assert in the dataset agreement that the dataset is only to be used for noncommercial research purposes, which we strictly adhere to. Paris street-view [232] dataset contains images collected using google street view and is to be used for noncommercial research purposes.

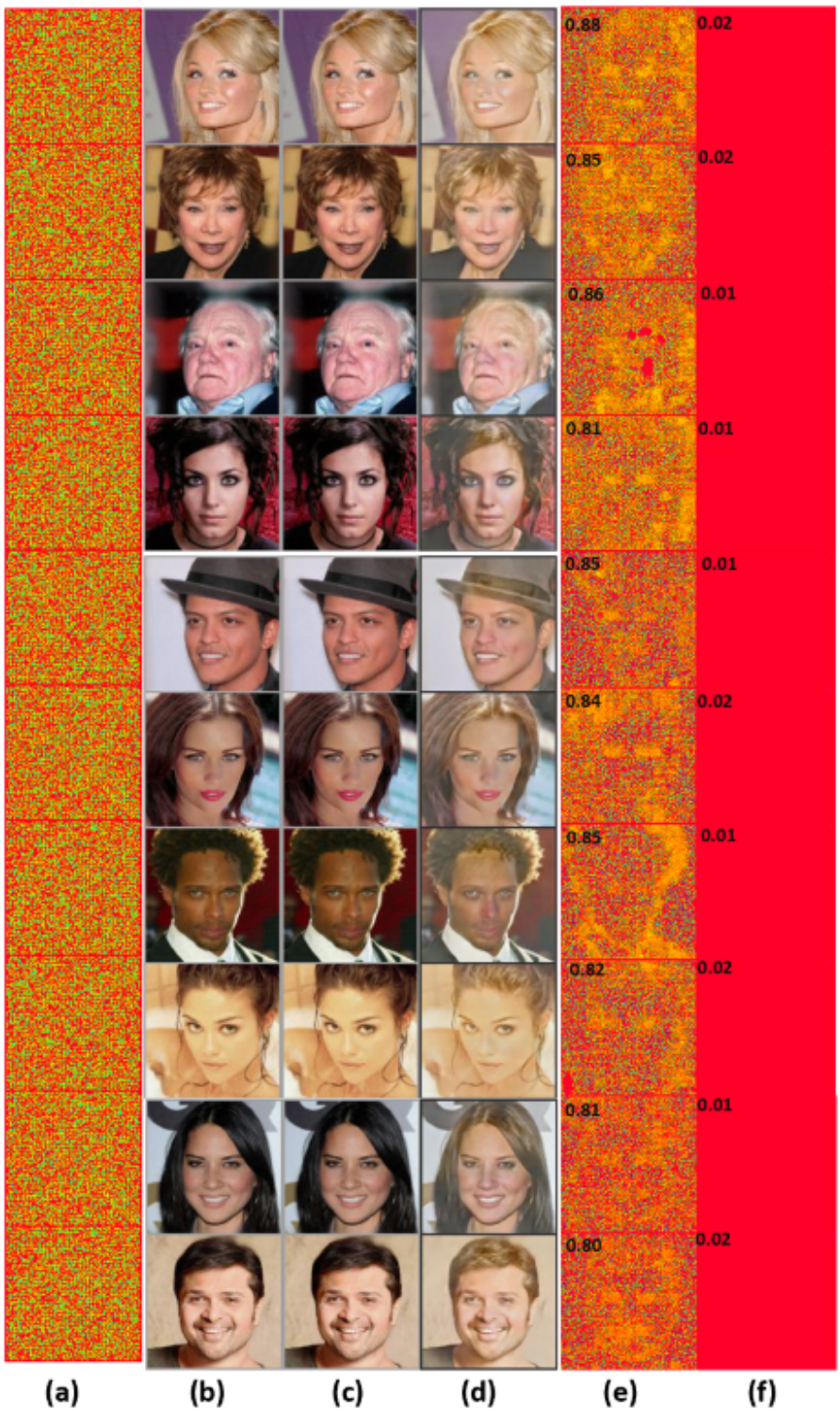


Figure B.3 Visualization of samples used for GM STGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

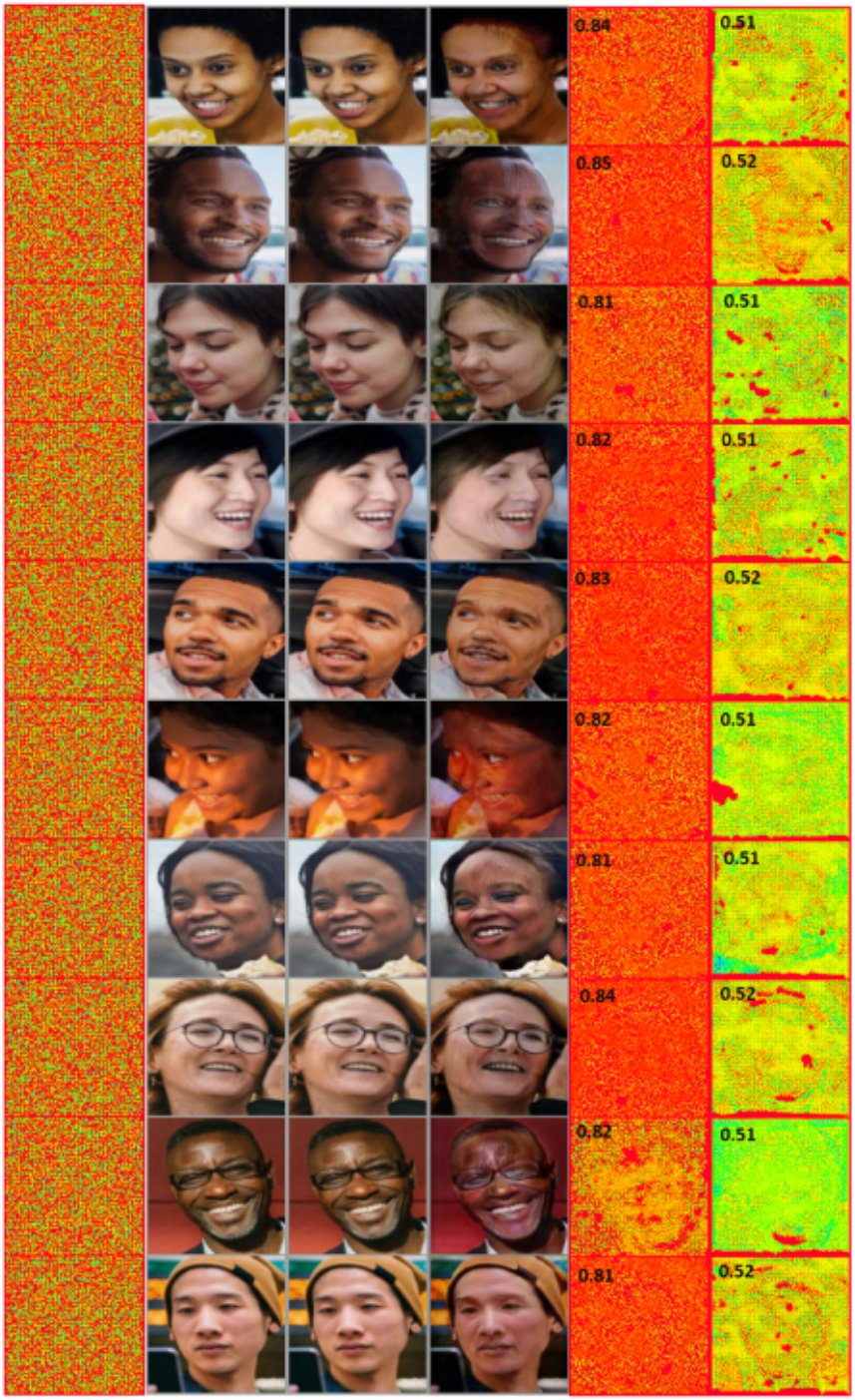


Figure B.4 Visualization of samples used for GM StarGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

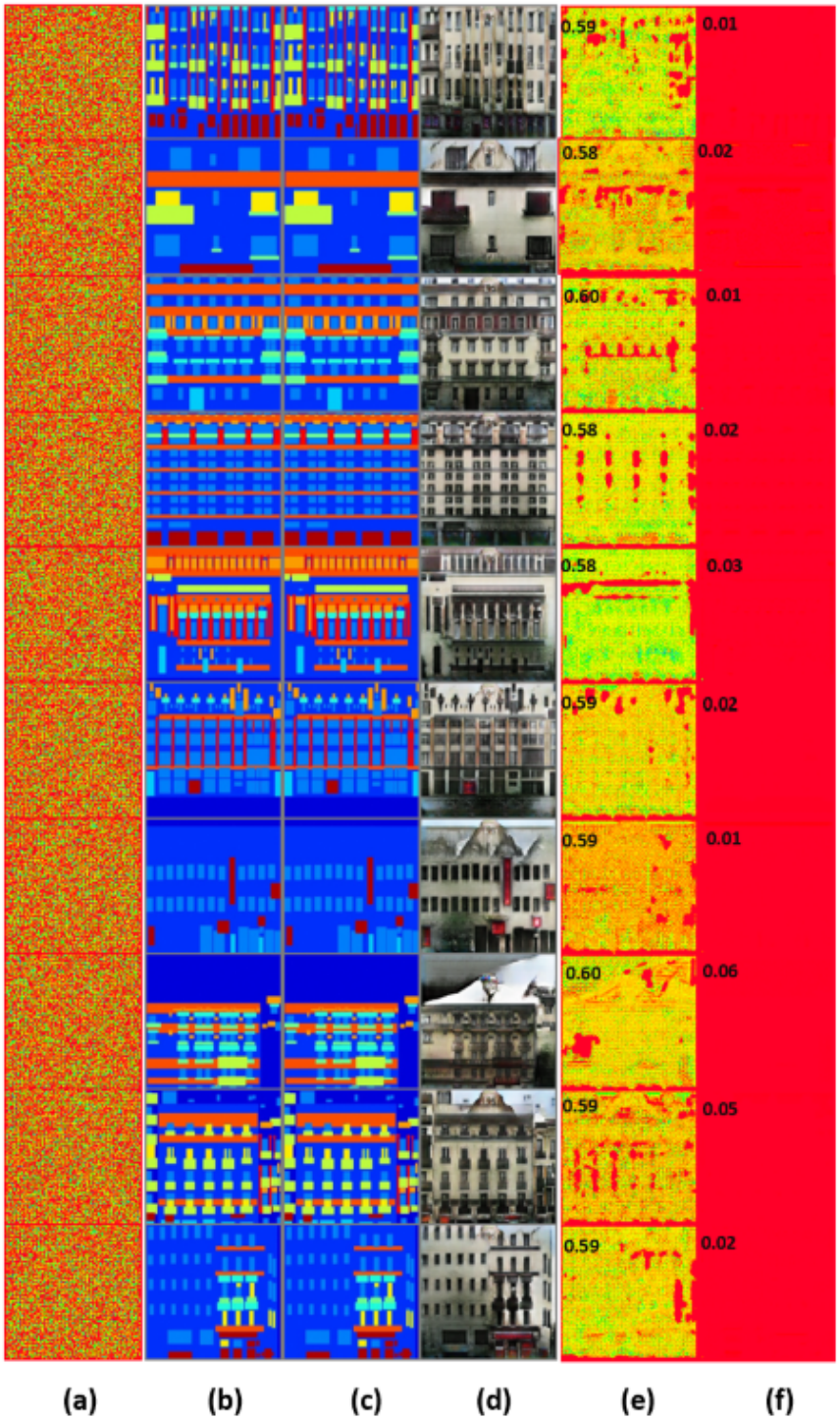


Figure B.5 Visualization of samples used for GM CycleGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

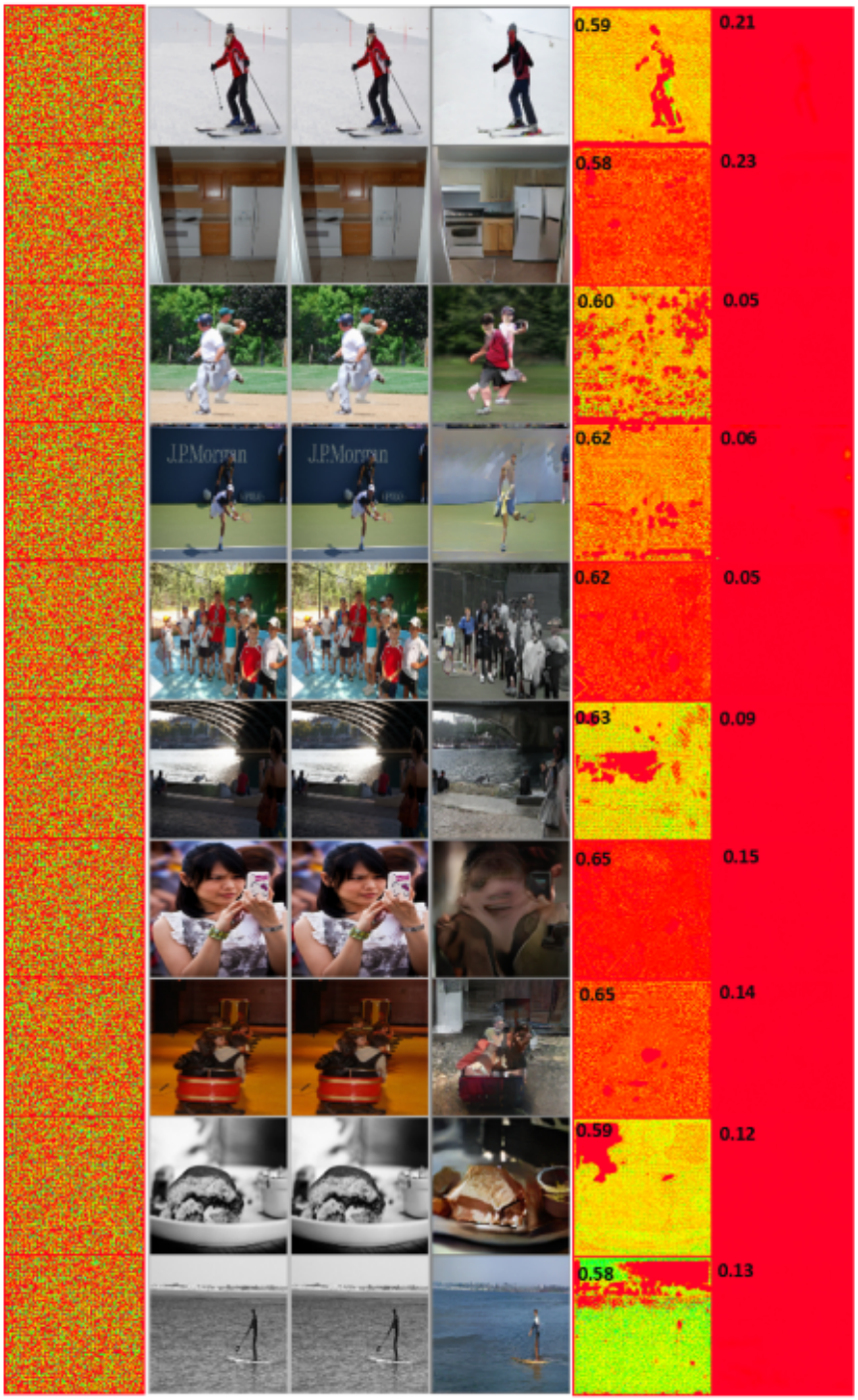


Figure B.6 Visualization of samples used for GM GauGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

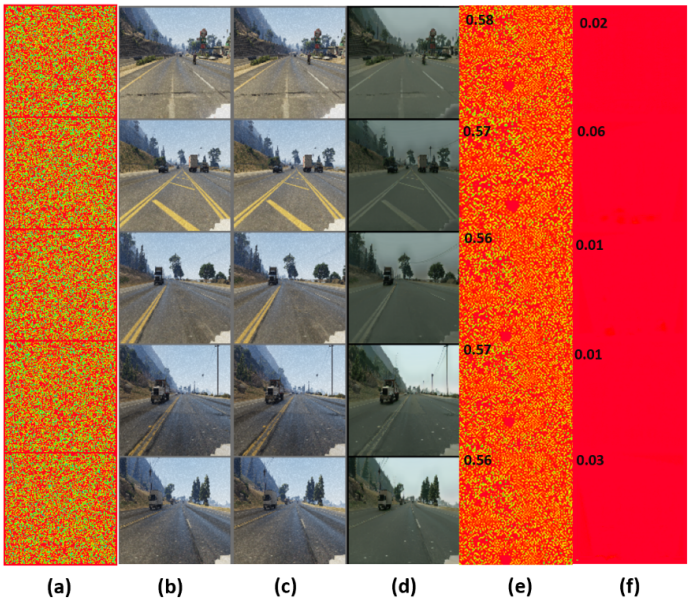


Figure B.7 Visualization of samples used for GM UNIT; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

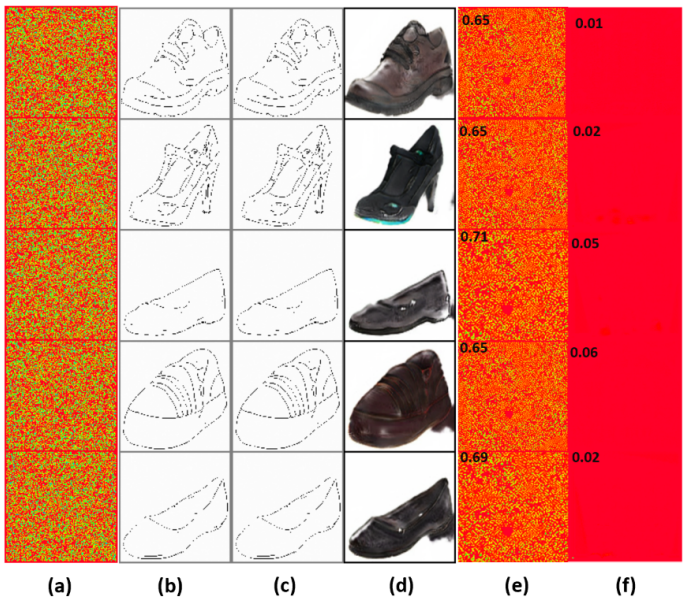


Figure B.8 Visualization of samples used for GM MUNIT; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

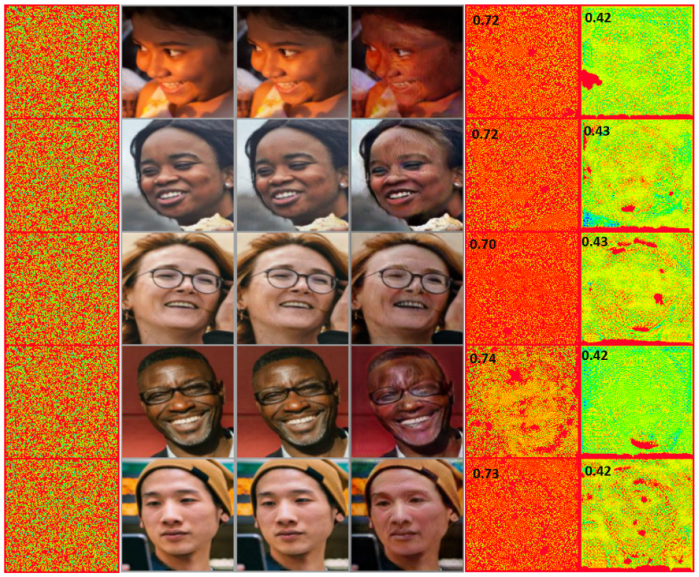


Figure B.9 Visualization of samples used for GM StarGANv2; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

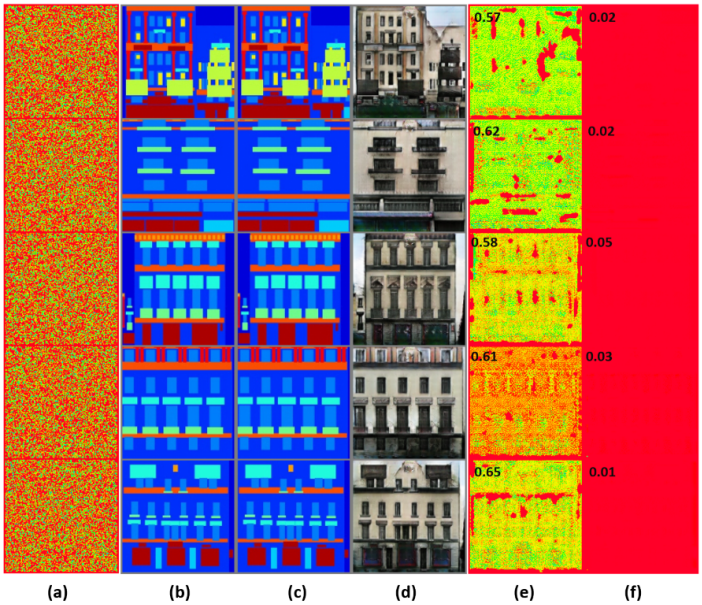


Figure B.10 Visualization of samples used for GM BicycleGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

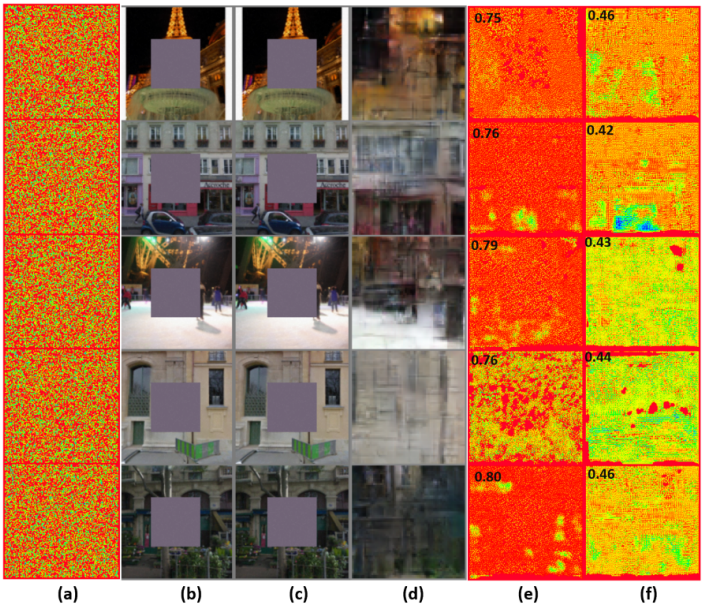


Figure B.11 Visualization of samples used for GM CONT_Encoder; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

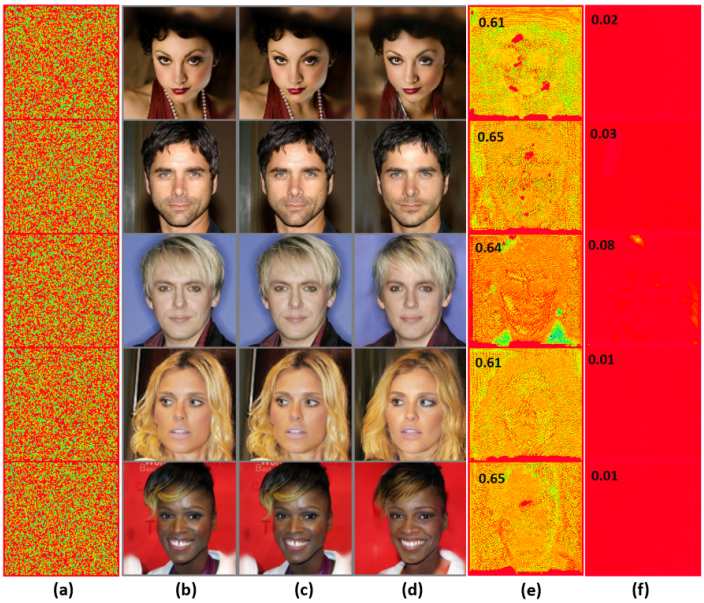


Figure B.12 Visualization of samples used for GM SEAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

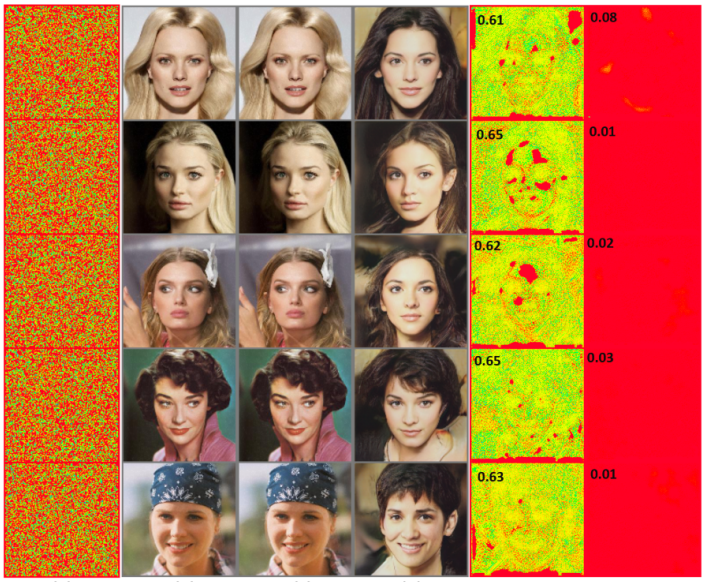


Figure B.13 Visualization of samples used for GM ALAE; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

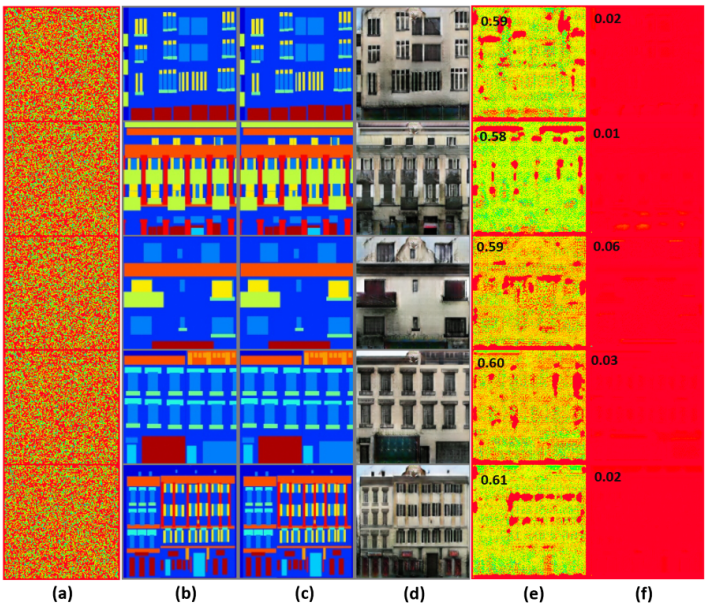


Figure B.14 Visualization of samples used for GM Pix2Pix; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

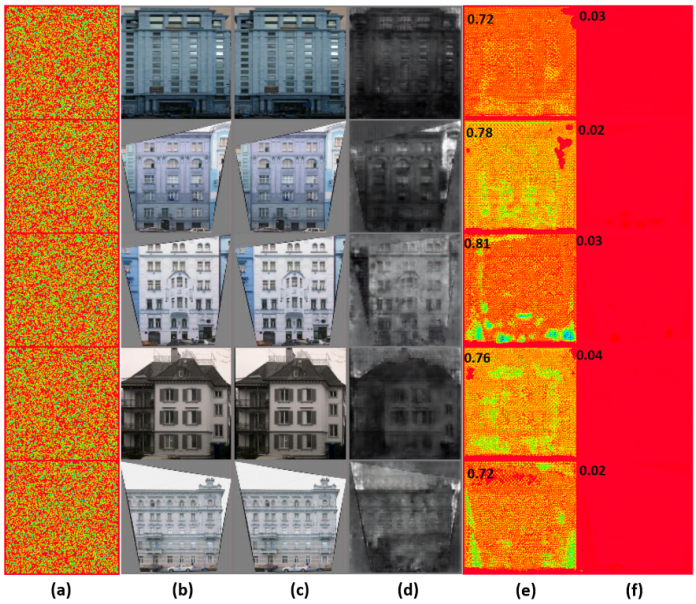


Figure B.15 Visualization of samples used for GM DualGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

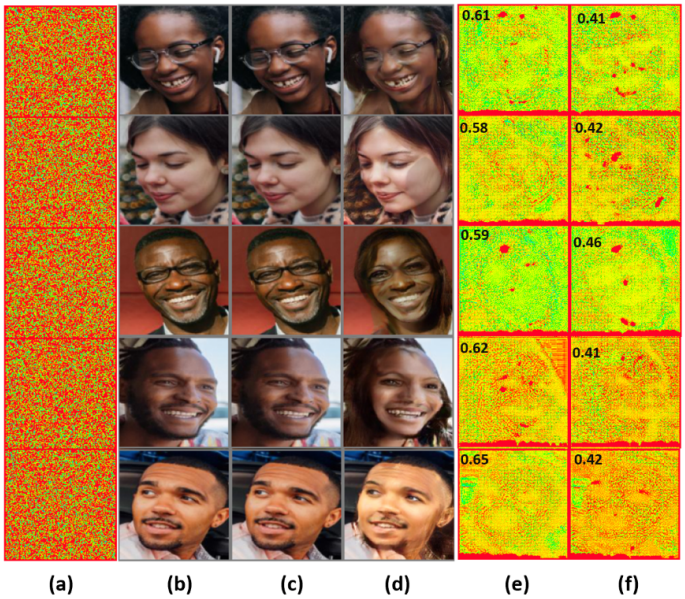


Figure B.16 Visualization of samples used for GM CouncilGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

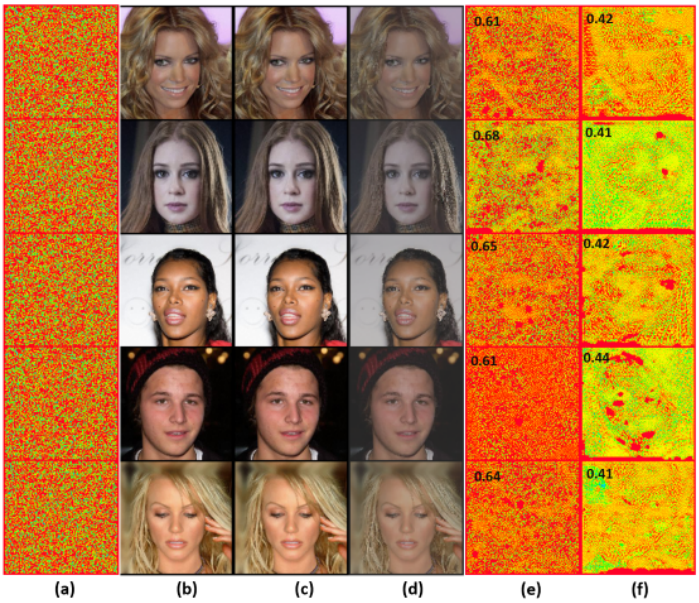


Figure B.17 Visualization of samples used for GM ESRGAN; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

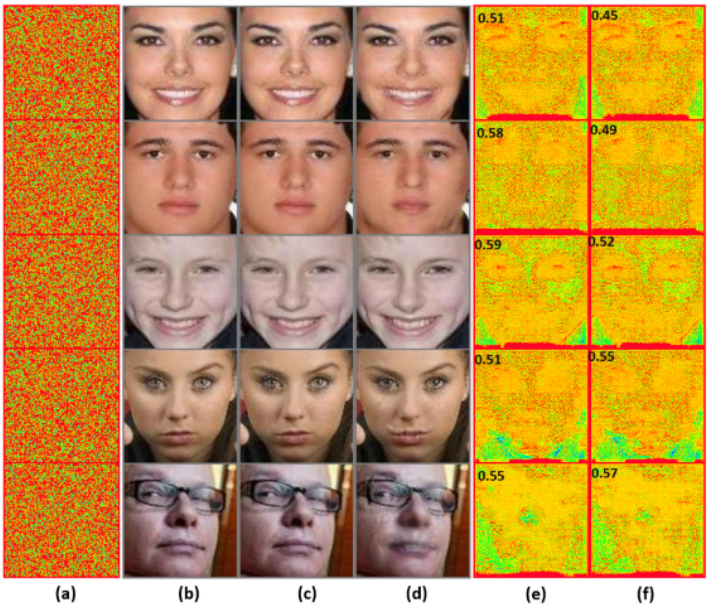


Figure B.18 Visualization of samples used for GM GANimation; (a) added template, (b) real images, (c) encrypted real images after adding a template, (d) manipulated images output by a GM, (e) recovered template from (c), and (f) recovered template from (d). Top left corner in last two columns shows the cosine similarity of the recovered template with the added template.

APPENDIX C

MALP APPENDIX

C.1 Implementation Details

Experimental Setup and Hyperparameters We train MaLP for 150, 000 iterations with a batch size of 4. For all of the networks, we use Adam optimizer except for the transformer which uses AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay $0.5e^{-5}$ and eps $1e^{-8}$. The learning rate is $1e^{-5}$ for all networks. The constraint weights are set as: $\lambda_1 = 100, \lambda_2 = 5, \lambda_3 = 4, \lambda_4 = 25, \lambda_5 = 25, \lambda_6 = 25, \lambda_7 = 50, \lambda_8 = 15, \lambda_9 = 20, \lambda_{10} = 50$. We use a template set size of 1 and template strength as 30% unless mentioned. All experiments are conducted on one NVIDIA K80 GPU.

Network Architecture. We show the network architecture of various components of MaLP in Fig. C.1. The shared network consists of 1 stem convolutional layer and 4 convolution blocks. Each convolution block consists of convolutional and batch normalization layers followed by ReLU activation. The output of the shared network is given to $\mathcal{E}_E$ and $\mathcal{E}_C$, both having the same architecture with 3 convolution blocks and 1 stem convolutional layer. We use the transformer $\mathcal{E}_T$ in the second branch of the framework where the ViT [79] architecture is adopted. The transformer consists of 6 encoder blocks, and a dropout of 0.1 is used. The features of the transformer are reshaped to the shape of the fakeness map *i.e.*$1 \times 128 \times 128$. Finally, we use a classifier C on the predicted fakeness maps to perform real *vs*.fake binary classification. The classifier has 8 convolution blocks, 1 stem convolutional layer, and 3 fully connected layers. We apply the ReLU activation between the layers.

GMs and dataset license information. We use a variety of face and generic GMs to show the effectiveness of MaLP. The information for all the GMs along with their training datasets, is shown in Tab. C.1. For many GMs used by [6], We use the test images released by [6]. for the remaining GMs, we would release the test images for fair comparison of generalization benchmark by the future works. We also show more visualization samples of the predicted fakeness maps by MaLP in Fig. C.2- Fig. C.5. All the fakeness maps are shown in "pink" cmap for better representation.

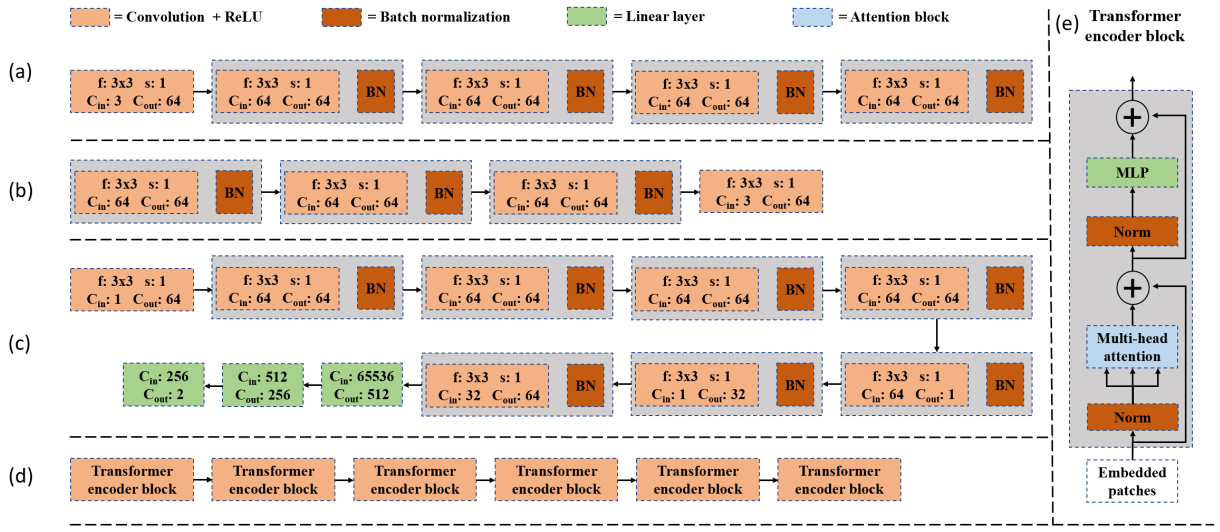


Figure C.1 Network architecture for different components of MaLP. (a) Shared network, (b) Encoder $\mathcal{E}_E$ and CNN network $\mathcal{E}_C$, (c) Classifier C, (d) Transformer $\mathcal{E}_T$, and (e) Transformer encoder block.

We also indicate the cosine similarity between the predicted and ground truth fakeness maps. We observe that the fakeness maps for encrypted images have minimal bright regions. However, for fake images, MaLP is able to localize the modified regions well, considering the modified attributes/GMs are unseen in training.

The face datasets include CelebA [200] and CelebA-HQ [157], both of which don't have any associated Institutional Review Board (IRB) approval. The authors for both datasets mention the availability of the dataset for non-commercial research purposes, which we strictly adhere to. For generic images datasets, we use Facades [306], COCO [30], Horse2Zebra [385], Summer2Winter [385], GTA2CITY [262], Edges2Shoes [144], Paris street-view [240] and Sketch-Photo [332] datasets. All the mentioned generic image datasets can be used for non-commercial research purposes, as mentioned by the authors, and we use the datasets for the same purposes.

Image Editing Degradations. We apply several image editing degradations to the test set to verify the robustness of MaLP. The details of these operations are listed below:

1. JPEG compression: We compress the image with the compression quality of 50%.
2. Blur: We apply the Gaussian blur with a filter size of 7×7.
3. Noise: We apply a Gaussian noise with zero mean and unit variance.

Table C.1 List of GMs along with their training datasets.

Dataset	GMs
CelebA [200]	STGAN [194], AttGAN [129], StarGAN [52], GANimation [248], CouncilGAN [232], ESRGAN [333], GDWCT [49]
CelebA-HQ [157]	SEAN [387], StarGAN-v2 [53], ALAE [245], DRGAN [304], ColorGAN [223],
Facades [306]	CycleGAN [385], BicycleGAN [386], Pix2Pix [144]
COCO [30]	GauGAN [238]
Horse2Zebra [385]	AutoGAN [371]
Summer2Winter [385]	DRIT [177]
GTA2CITY [262]	UNIT [195]
Edges2Shoes [144]	MUNIT [139]
Paris Street-view [232]	Cont_Enc [240]
Sketch-Photo [332]	DualGAN [357]

Table C.2 Ablation for localization loss.

Loss	CS ↑	PSNR ↑	SSIM ↑
CS	0.9356	22.16	0.7114
CS + L_2	0.9230	18.98	0.6614
CS + SSIM + L_2	0.9211	19.12	0.6816
CS + SSIM + L_1	0..8777	14.01	0.3712
CS + SSIM	**0.9394**	**23.020**	**0.7312**

4. Low-resolution: We resize the image to half the original resolution and restore it back to the original resolution using linear interpolation.

Potential Societal Impact The problem of manipulation localization is crucial from the perspective of media forensics. Localizing the fake regions not only helps in the detection of these fake media but, in the future, can also help recover the original image that the GM has manipulated. We also show that MaLP can be used as a discriminator to improve the quality of GMs. While this is an interesting application of MaLP, it can be a possibility that the GMs become more robust to our framework, decreasing the localization performance if the training of the GM is done from scratch.

C.2 Additional Experiments

Localization Loss. We show the importance of manipulation loss (defined in Eq. 8) in Sec. 4.6. We perform an ablation to formulate the loss of fakeness maps for manipulated images. As shown in Tab. C.2, we try experimenting with various loss functions *i.e.*cosine similarity (CS), L_1, L_2 and

Table C.3 Comparison with [141] using multiple GMs in training. MaLP is able to outperform [141] by training images manipulated by only STGAN.

Method	Training GMs	Cosine similarity ↑		
		AttGAN	StarGAN	StyleGAN
Hunag *et al*. [141]	STGAN + ICGAN + PGGAN + StyleGAN + StyleGAN2 + StarGAN + AttGAN	0.6940	0.8494	0.7479
MaLP	STGAN	**0.8557**	**0.8718**	**0.8255**

Table C.4 Performance of MaLP across different attribute modifications seen in training.

Method	Cosine similarity ↑					
	Bald	Bangs	Black Hair	Eyeglasses	Mustache	Smile
[141]	0.9014	0.8850	0.8817	0.9093	0.9152	0.8634
MaLP	**0.9478**	**0.9329**	**0.9367**	**0.9549**	**0.9470**	**0.9489**

structural similarity index measure (SSIM). Using just the CS loss results in better performance compared to combining it with L_1 or L_2 loss. We observe a huge deterioration in performance when using L_1 loss. This can be explained as PSNR and SSIM are directly related to mean squared error which is optimized by either an L_2 or SSIM loss. Finally, adopting an SSIM loss with CS loss results in a better performance as both of them are more related to the metrics, making it easier for MaLP to converge.

Comparison with Baseline. Due to the limited GPU memory, we conduct proactive training with one GM only because the GM needs to be loaded to the memory and used on the fly. On the other hand, passive methods can be trained on multiple GMs because the image generation processes are conducted offline. As shown in Tab. C.3, [141] trains on images manipulated by 7 different GMs, unlike MaLP, which is trained on images manipulated by only 1 GM. We show the performance on three GMs, which are seen for [141], but unseen for MaLP. MaLP performs better even though these GMs' images are not seen in training. Therefore, even though the training of MaLP is limited by 1 GM, it can achieve better generalization to other GMs proving the effectiveness of proactive schemes.

Multiple Attribute Modifications. Instead of training on bald attribute modification by STGAN, we train and test MaLP on multiple attribute modifications. These include bald, bangs, black hair,

Table C.5 Ablation study for transformer architecture.

Optimizer	Depth	Dropout	Cosine similarity↑	Accuracy↑
Adam	6	0.1	0.8839	0.9514
AdamW	1	0.0	0.8825	0.9647
AdamW	1	0.0	0.8826	0.9680
AdamW	3	0.0	0.8830	0.9705
AdamW	6	0.1	**0.8848**	**0.9856**

eyeglasses, mustache, and smile manipulation. We show the results in Tab. C.4. MaLP performs better for all the attribute modifications compared to the passive method [141]. We also observe an increase in cosine similarity compared to when MaLP is trained on only bald attribute modification. This is expected, as the more types of modifications MaLP sees in training, the better it learns to localize.

Transformer Architecture Ablation. We ablate various parameters of the transformer to select the best architecture for manipulation localization. We experiment with parameters that include optimizer, depth *i.e.*number of blocks, and dropout. We only use the transformer branch and switch off the CNN branch during training. The results are shown in Tab. C.5. We observe that the localization performance is almost the same when using the transformer to predict fakeness maps. However, the detection accuracy has a significant impact. Having dropout does increase the performance for detection and localization. Further, using the weighted Adam optimizer is more beneficial than using the vanilla Adam optimizer. Therefore, we adopt the architecture of the transformer with 6 blocks and optimize it with a weighted Adam optimizer. Finally, we also include the dropout to achieve the best performance for localization and detection.

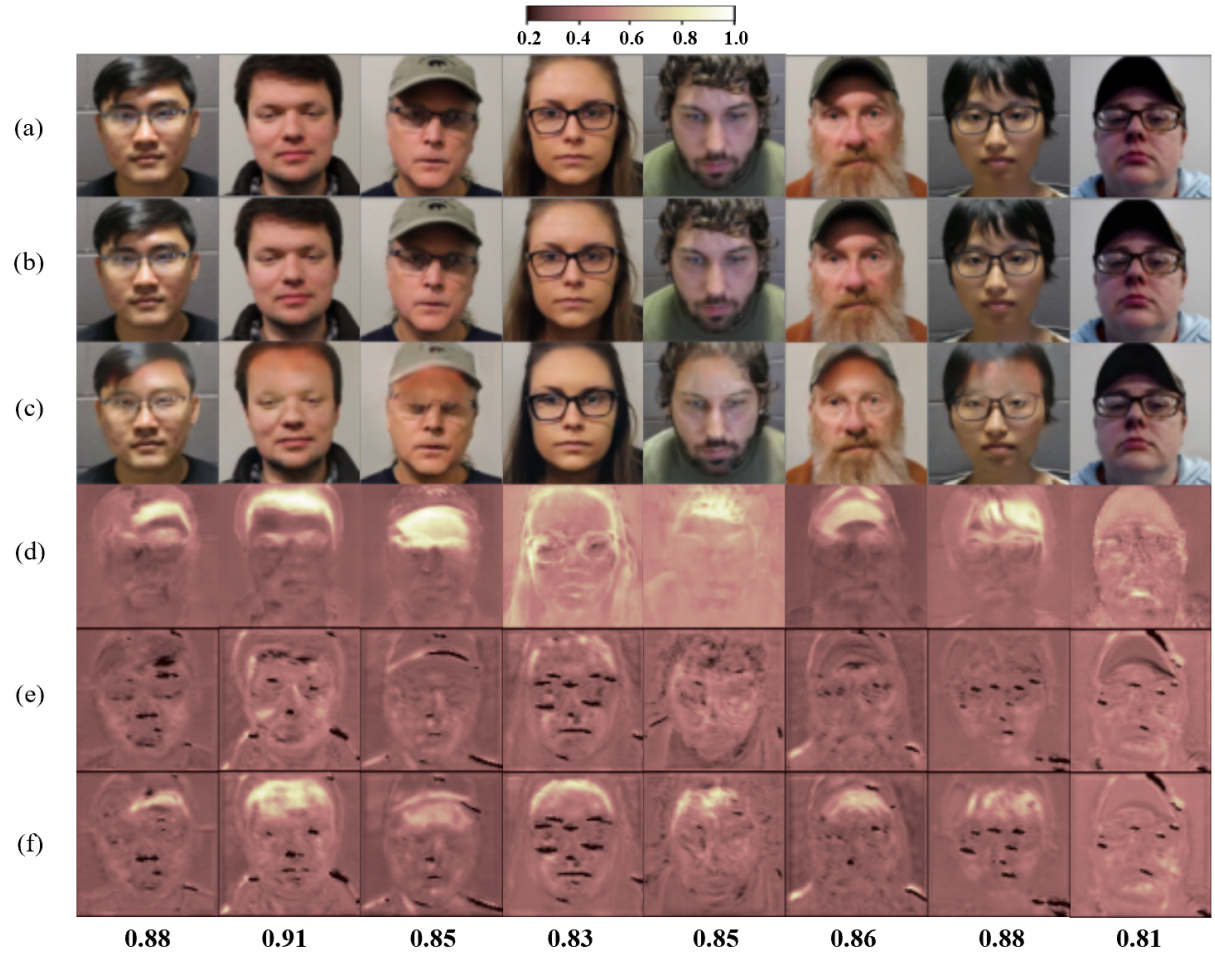


Figure C.2 Visualization of fakeness maps for different attribute modifications by STGAN. (a) Real image, (b) encrypted image, (c) manipulated image, (d) ground-truth $\boldsymbol{M}_{GT}$, (e) predicted fakeness map for encrypted images, and (f) predicted fakeness map for manipulated images. We also show the cosine similarity between the predicted and ground-truth fakeness map below (f). All face images come from SiWM-v2 data [115].

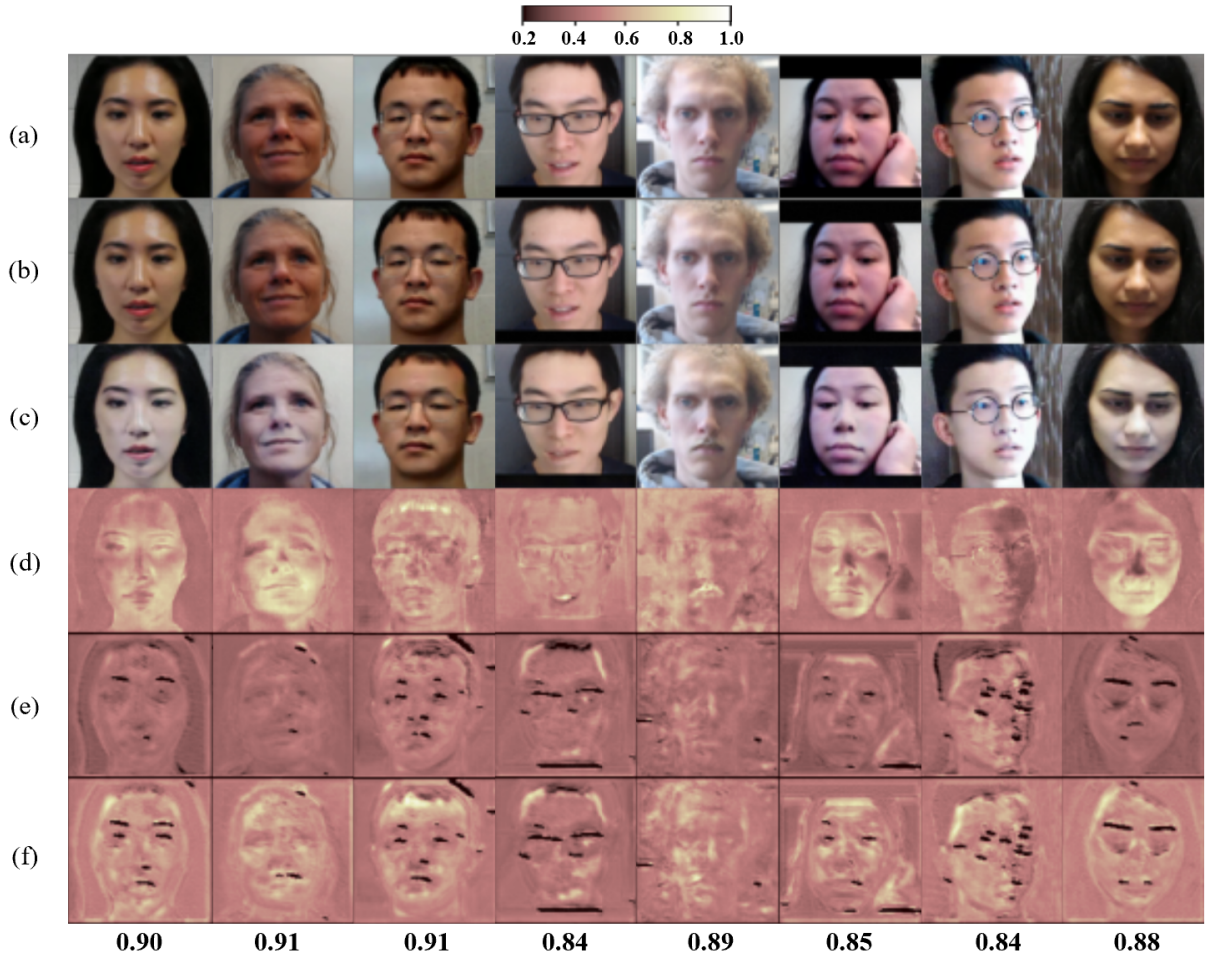


Figure C.3 Visualization of fakeness maps for different attribute modifications by STGAN. (a) Real image, (b) encrypted image, (c) manipulated image, (d) ground-truth $\boldsymbol{M}_{GT}$, (e) predicted fakeness map for encrypted images, and (f) predicted fakeness map for manipulated images. We also show the cosine similarity between the predicted and ground-truth fakeness map below (f). All face images come from SiWM-v2 data [115].

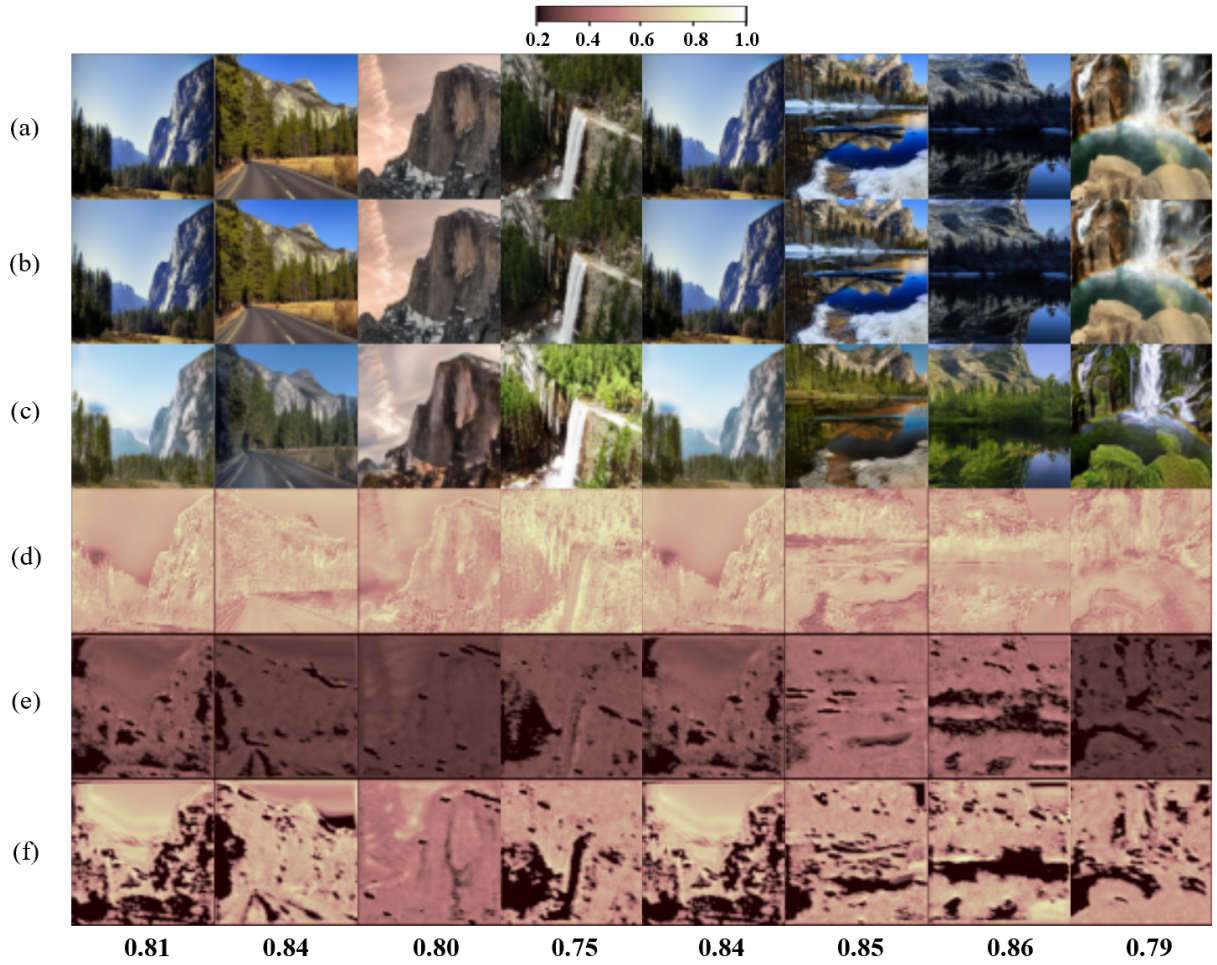


Figure C.4 Visualization of fakeness maps for manipulation by DRIT. (a) Real image, (b) encrypted image, (c) manipulated image, (d) ground-truth $\boldsymbol{M}_{GT}$, (e) predicted fakeness map for encrypted images, and (f) predicted fakeness map for manipulated images. We also show the cosine similarity between the predicted and ground-truth fakeness map below (f).

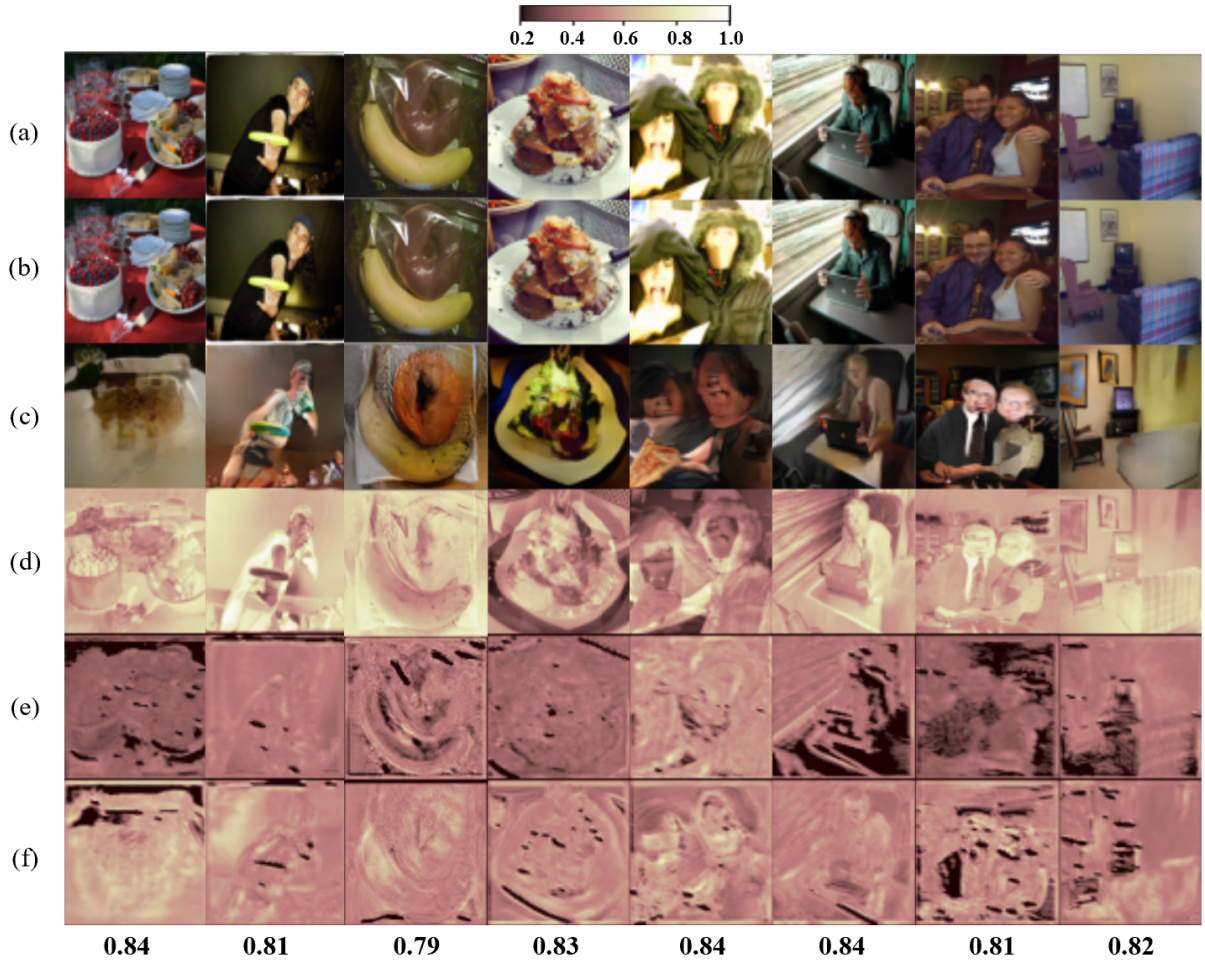


Figure C.5 Visualization of fakeness maps for manipulation by GauGAN. (a) Real image, (b) encrypted image, (c) manipulated image, (d) ground-truth $\boldsymbol{M}_{GT}$, (e) predicted fakeness map for encrypted images, and (f) predicted fakeness map for manipulated images. We also show the cosine similarity between the predicted and ground-truth fakeness map below (f).

APPENDIX D

PROBED APPENDIX

D.1 Proof of Lemma 1

We begin our proof by considering the image $\boldsymbol{i}$ as a column vector and the model as a linear regression model with learnable weights $\boldsymbol{w}_t$. The subscript of time t denotes that the weights change as one performs SGD updates.

SGD Steps. We first consider the gradient of weight ($\boldsymbol{w}_t$). The linear model uses SGD for training, therefore, $\boldsymbol{w}_t$ after t gradient steps is given by:

$$\boldsymbol{w}_t = \boldsymbol{w}_0 - \sum_{i=0}^{t} s_i \boldsymbol{g}_t = \boldsymbol{w}_0 - \sum_{i=0}^{t} s_i \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_t}, \tag{D.1}$$

where, for linear regression model with image $\boldsymbol{i}$, $\mathcal{L} = f(\boldsymbol{w}_t \boldsymbol{i} - z) = f(\eta)$. To estimate the gradient $\boldsymbol{w}_t$, we have,

$$\begin{aligned}
\boldsymbol{g}_t &= \frac{\partial \mathcal{L}(\boldsymbol{w}_t \boldsymbol{i} - z)}{\partial \boldsymbol{w}_t} \\
&= \frac{\partial \mathcal{L}(\boldsymbol{w}_t \boldsymbol{i} - z)}{\partial (\boldsymbol{w}_t \boldsymbol{i} - z)} \frac{\partial (\boldsymbol{w}_t \boldsymbol{i} - z)}{\partial \boldsymbol{w}_t} \\
&= \frac{\partial \mathcal{L}(\eta)}{\partial \eta} \boldsymbol{i} \\
\boldsymbol{g}_t &= \boldsymbol{i} \upsilon,
\end{aligned} \tag{D.2}$$

where $\upsilon = \frac{\partial \mathcal{L}(\eta)}{\partial \eta}$ is the gradient of the loss function wrt noise.

Optimal Weights. First, we will find the bound of the converged value $\boldsymbol{w}_\infty$ and the optimal value $\boldsymbol{w}_*$. If μ_w is mean of the learned weight, we have,

$$\begin{aligned}
\mathbb{E}\left(\|\boldsymbol{w}_\infty - \boldsymbol{w}_*\|_2^2\right) &= \mathbb{E}\left(\|\boldsymbol{w}_\infty - \mu_w + \mu_w - \boldsymbol{w}_*\|_2^2\right), \\
&= \mathbb{E}((\boldsymbol{w}_\infty - \mu_w)^T (\boldsymbol{w}_\infty - \mu_w)) + \mathbb{E}((\mu_w - \boldsymbol{w}_*)^T (\mu_w - \boldsymbol{w}_*)) \\
&+ 2\mathbb{E}((\boldsymbol{w}_\infty - \mu_w)^T (\mu_w - \boldsymbol{w}_*)), \\
&= \mathbb{E}((\boldsymbol{w}_\infty - \mu_w)^T (\boldsymbol{w}_\infty - \mu_w)) + \mathbb{E}((\mu_w - \boldsymbol{w}_*)^T (\mu_w - \boldsymbol{w}_*))
\end{aligned} \tag{D.3}$$

Using $\mathbb{E}(\boldsymbol{w}_\infty - \mu_w) = \mathbb{E}(\boldsymbol{w}_\infty) - \mu_w = \mu_w - \mu_w = 0$, we have

$$\implies \mathbb{E}\left(\|\boldsymbol{w}_\infty - \boldsymbol{w}_*\|_2^2\right) = Var(\boldsymbol{w}_\infty) + \mathbb{E}((\mu_w - \boldsymbol{w}_*)^T(\mu_w - \boldsymbol{w}_*)) \tag{D.4}$$

where $Var(\boldsymbol{w}) = \sum_j w_j^2$.

Gradient of Weight. Given the image vector $\boldsymbol{i}$, and noise η are statistically independent, the image and noise gradient υ defined in Eq. (D.2) are also statistically independent. We also assume that the distribution of image is normal Gaussian ($\mathbb{E}(\boldsymbol{i}) = 0$). Therefore, the expectation of the gradient $\boldsymbol{g}_t$ is given by,

$$\mathbb{E}(\boldsymbol{g}_t) = \mathbb{E}(\boldsymbol{i})\mathbb{E}(\upsilon) = 0, \tag{D.5}$$

Next, the variance of $\boldsymbol{g}_t$ is given as

$$Var(\boldsymbol{g}_t) = Var(\boldsymbol{i}\upsilon) = \mathbb{E}(\boldsymbol{i}^T\boldsymbol{i})[Var(\upsilon) + \mathbb{E}^2(\upsilon)] - \mathbb{E}(\boldsymbol{i})\mathbb{E}(\upsilon). \tag{D.6}$$

We assume that image pixels are normally distributed. This is common since the networks do a mean subtraction before inputting to the network. Thus, $\mathbb{E}(\boldsymbol{i}) = 0$. Hence, we have

$$Var(\boldsymbol{g}_t) = \mathbb{E}(\boldsymbol{i}^T\boldsymbol{i})Var(\upsilon). \tag{D.7}$$

Converged Weight. From Eq. (D.1), the expectation of the weight at time t is,

$$\begin{aligned}\mathbb{E}(\boldsymbol{w}_t) &= \mathbb{E}(\boldsymbol{w}_0) + \sum_{i=0}^{t} s_i\mathbb{E}(\boldsymbol{g}_j) \\ &= 0 \text{ (Using Eq. (D.5))}\end{aligned} \tag{D.8}$$

Therefore, for converged weight,

$$\begin{aligned}\mathbb{E}(\boldsymbol{w}_\infty) &= \lim_{t\to\infty} \mathbb{E}(\boldsymbol{w}_t), \\ \mathbb{E}(\boldsymbol{w}_\infty) &= \mathbb{E}(\mu_w) = 0.\end{aligned} \tag{D.9}$$

For variance, using Eq. (D.1) we have,

$$Var(\boldsymbol{w}_t) = Var(\boldsymbol{w}_0) + (\sum_{i}^{t} s_j^2) Var(\boldsymbol{g}_t).$$

Therefore, we have,

$$\begin{aligned} Var(\boldsymbol{w}_\infty) &= \lim_{t\to\infty}(Var(\boldsymbol{w}_t)) \\ &= Var(\boldsymbol{w}_0) + \Big(\lim_{t\to\infty}\sum_{i=}^{t} s_j^2\Big) Var(\boldsymbol{g}_t) \\ Var(\boldsymbol{w}_\infty) &= Var(\boldsymbol{w}_0) + \mathcal{S}' Var(\boldsymbol{g}_t). \end{aligned} \tag{D.10}$$

Substituting Eq. (D.7) in the above equation, we have

$$Var(\boldsymbol{w}_\infty) = Var(\boldsymbol{w}_0) + \mathcal{S}'\mathbb{E}(\boldsymbol{i}^T\boldsymbol{i}) Var(\upsilon), \tag{D.11}$$

Going back to Eq. (D.4), and substituting Eq. (D.8) and Eq. (D.10), we have,

$$\begin{aligned} &\mathbb{E}\left(\|\boldsymbol{w}_\infty - \boldsymbol{w}_*\|_2^2\right) = Var(\boldsymbol{w}_0) + \mathcal{S}'\mathbb{E}(\boldsymbol{i}^T\boldsymbol{i}) Var(\upsilon) + \mathbb{E}(||\boldsymbol{w}_*||^2) \\ \implies &\mathbb{E}\left(\|\boldsymbol{w}_\infty - \boldsymbol{w}_*\|_2^2\right) = c + \mathcal{S} Var(\upsilon) \end{aligned} \tag{D.12}$$

where c is independent of loss function $\mathcal{L}$ and $\mathcal{S} = \mathcal{S}'\mathbb{E}(\boldsymbol{i}^T\boldsymbol{i})$ is also another constant.

Lemma 1**.**

We assume that the regression error term $e = \boldsymbol{w}^T\boldsymbol{i} - \hat{y}$, is drawn from zero mean Gaussian with variance σ^2 as in [128]. So,

$$Var(\hat{e}) = Var(\boldsymbol{w}^T\boldsymbol{i} - \hat{y}) = \sigma^2. \tag{D.13}$$

For a passive detector with converged weights $\boldsymbol{w}_\infty$, we have,

$$\begin{aligned} \mathbb{E}\left(\|\boldsymbol{w}_\infty - \boldsymbol{w}_*\|_2^2\right) &= c + \mathcal{S} Var(\upsilon) \\ &= c + \mathcal{S} Var(e) \\ \implies \mathbb{E}\left(\|\boldsymbol{w}_\infty - \boldsymbol{w}_*\|_2^2\right) &= c + \mathcal{S}\sigma^2 \end{aligned} \tag{D.14}$$

Similarly, for a proactive detector with converged weights $\boldsymbol{w}'_\infty$, we have

$$\mathbb{E}\left(\left\|\boldsymbol{w}'_\infty - \boldsymbol{w}_*\right\|_2^2\right) = c + \mathcal{S} Var(\upsilon') \tag{D.15}$$

Assume that a proactive detector multiplies the input image vector $\boldsymbol{i}$ with a scalar template s. From Eq. (D.12), we write the loss term as,

$$\mathcal{L}^{'} = \frac{1}{2}\left(s\boldsymbol{w}^{T}\boldsymbol{i} - \hat{y}\right)^{2}$$
$$\implies \frac{\partial \mathcal{L}^{'}}{\partial \boldsymbol{w}} = (s\boldsymbol{w}^{T}\boldsymbol{i} - \hat{y})s\boldsymbol{i} \tag{D.16}$$

Taking the variance,

$$\begin{aligned} Var(\upsilon^{'}) = Var\left(\frac{\partial \mathcal{L}^{'}}{\partial \boldsymbol{w}}\right) &= Var((s\boldsymbol{w}^{T}\boldsymbol{i} - \hat{y})s\boldsymbol{i}) \\ &= Var(s(\hat{y}+e) - \hat{y})s^{2}Var(\boldsymbol{i}) \quad \text{, assuming } \mathbb{E}(\boldsymbol{i}) = 0 \\ &= Var(se + (s-1)\hat{y})s^{2}Var(\boldsymbol{i}) \\ &= (Var(se) + Var((s-1)\hat{y}))s^{2}Var(\boldsymbol{i}) \\ &= s^{2}Var(e)s^{2}Var(\boldsymbol{i}) \qquad \text{, assuming } Var(\hat{y}) = 0 \\ &\leq s^{2}Var(e)s^{2} \quad \text{, assuming } Var(\boldsymbol{i}) \leq 0.5\times(-1)^{2}+0.5\times 1^{2} = 1 \end{aligned} \tag{D.17}$$

$$\implies Var(\upsilon^{'}) \leq s^{4}\sigma^{2} \tag{D.18}$$

If the magnitude of the scalar template is bounded by 1 i.e., $s^{2} < 1$, we have

$$Var(\upsilon^{'}) < \sigma^{2}. \tag{D.19}$$

The above shows that the gradients in the proactive model has less noise than the passive model (a key for better convergence). Substituting above in Eq. (D.15), we have

$$\begin{aligned} \mathbb{E}\left(\left\|\boldsymbol{w}^{'}_{\infty} - \boldsymbol{w}_{*}\right\|_{2}^{2}\right) &= c + \mathcal{S}Var(\upsilon^{'}) \\ &< c + \mathcal{S}\sigma^{2} \\ &< c + \mathcal{S}Var(\upsilon) \end{aligned}$$
$$\implies \mathbb{E}\left(\left\|\boldsymbol{w}^{'}_{\infty} - \boldsymbol{w}_{*}\right\|_{2}^{2}\right) < \mathbb{E}\left(\|\boldsymbol{w}_{\infty} - \boldsymbol{w}_{*}\|_{2}^{2}\right). \tag{D.20}$$

The last inequality follows trivially from Eq. (D.14).

D.2 Proof of Theorem 1

From Lemma 1, we have,

$$\mathbb{E}\left(\left\|\boldsymbol{w}_{\infty}^{'}-\boldsymbol{w}_{*}\right\|_2^2\right) < \mathbb{E}\left(\|\boldsymbol{w}_{\infty}-\boldsymbol{w}_{*}\|_2^2\right)$$

$$\implies Var(\boldsymbol{w}_{\infty}^{`}) < Var(\boldsymbol{w}_{\infty})$$

$$\implies \mathbb{E}(|\boldsymbol{w}_{\infty}^{`T}\boldsymbol{i}-y|) < \mathbb{E}(|\boldsymbol{w}_{\infty}^{T}\boldsymbol{i}-y|)$$

$$\implies \mathbb{E}(\hat{y}^{`}-y) < \mathbb{E}(\hat{y}-y) \tag{D.21}$$

Since the proactive detector has a better bounding box prediction,

$$\implies \mathbb{E}(IoU_{2D}^{'}) > \mathbb{E}(IoU_{2D}) \tag{D.22}$$

Since AP is a non-decreasing function of IoU_{2D}, we have,

$$AP^{`} \geq AP. \tag{D.23}$$

An important point to note is that the non-decreasing nature does not keep the inequality strict. In other words, we agree that the final AP from passive and pro-active schemes could be equal. However, our experience says that IoU improvements, especially close to 1, lead to significant AP improvements. Current SoTA detectors already achieve decent IoU; hence, even a slight improvement in IoU improves the AP score.

D.3 Implementation Details

We now include more details of our method here.

Network Architecture. The network architecture of encoder $\mathcal{E}$ and decoder $\mathcal{D}$ network used for PrObeD is shown in Fig. D.1. Both networks consist of 2 stem convolution layers and 13 blocks, each block containing convolutional, batch normalization, and ReLU activation layers. The images are given as input to the encoder network to output the template, which is multiplied by the input images to make them encrypted. The encrypted images are then passed to the decoder network to recover the template. Finally, we input encrypted images to different object detectors to perform detection.

Dataset license information. We use benchmark datasets for GOD and COD. The authors for MS-COCO [190] dataset specify that the annotations in this dataset, along with this website, belong to the COCO Consortium and are licensed under a Creative Commons Attribution 4.0 License. The COD10K dataset is available for non-commercial purposes only [81]. The CAMO data is published under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License [176]. Finally, the NC4K dataset is available to use for non-commercial purposes.

Experimental Setup and Hyperparameters. PrObeD is trained in an end-to-end manner for all the object detectors, with training iterations similar to the pretrained object detector. For both encoder and decoder networks, we use Adam optimizer with a learning rate of $1e^{-5}$. We use different weights of $[\lambda_{OBJ}, \lambda_E, \lambda_D]$ for different object detectors. We use [7,10,10] for Faster-RCNN, [50, 1.25, 4.25] for YOLOv5, [50, 7.5, 7.5] for DeTR and [10, 0.1, 0.1] for DGNet. All experiments are conducted on one NVIDIA A100 GPU.

D.4 Additional Experiments

Train COD detector DGNet more. Similar to the GOD detector, we train the COD detector DGNet for more iterations, similar to after applying PrObeD. The results are shown in Tab. D.1. We see a similar behavior as seen in GOD detectors; the performance improves after training for more iterations, but only up to a certain extent. PrObeD is able to improve performance by a larger

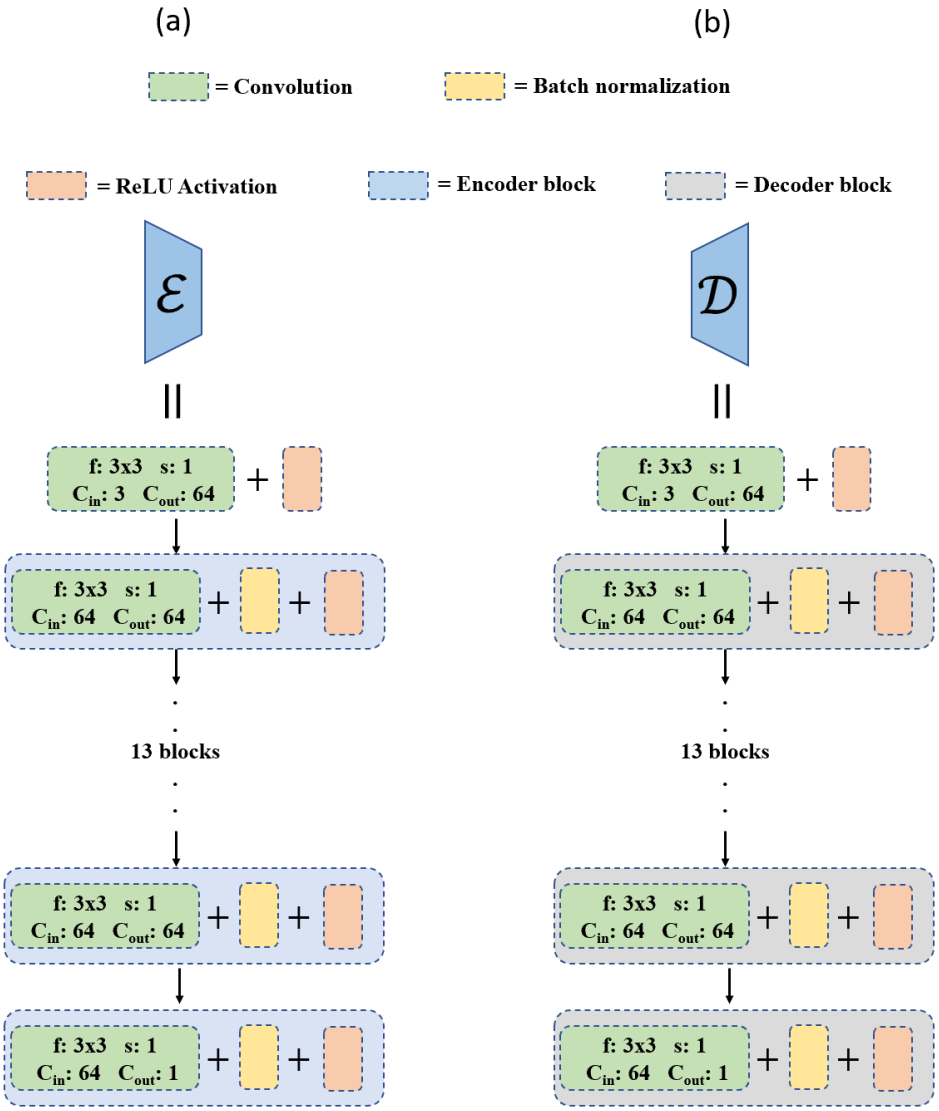


Figure D.1 **Architecture** for encoder and decoder network.

margin, showing the effectiveness of the proactive schemes.

COD loss. Our loss design is inspired by the prior proactive works [7, 6], which estimate the learnable template by applying a cosine similarity loss. The authors experiment with various loss types, showing the effectiveness of the cosine similarity loss design. However, COD is analogous to the segmentation task, which generally adopts a loss design of cross-entropy loss with dice loss, which might be beneficial for COD. We perform an ablation by applying cross-entropy loss with dice loss for COD. The results are shown in Tab. D.2. We see that our proactive wrapper is not benefiting by removing the cosine similarity loss, proving the study of the prior proactive works.

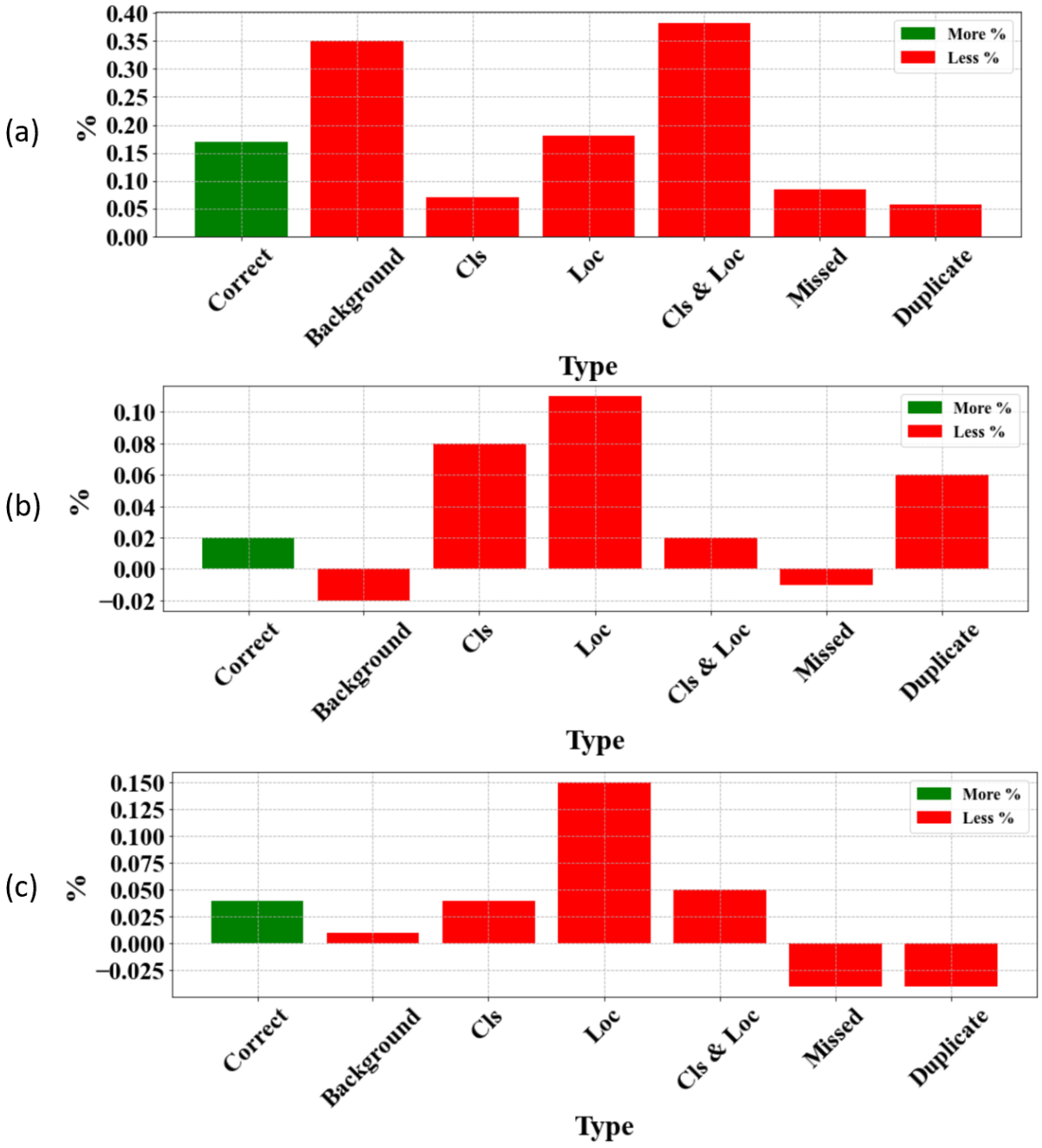


Figure D.2 **Error analysis** for (a) Faster-RCNN, (b) YOLOv5, and (c) DeTR. PrObeD is able to improve the number of correct predictions and reduce most errors.

Error analysis. Following [23], there can be a number of errors that deteriorate the performance of the object detector. These are:

1. Classification error (Cls): Localized correctly but classified incorrectly.

Table D.1 **Ablation of training iterations** on DGNet for more iterations similar to after applying PrObeD.

Method	Iter	$E_m\uparrow$	$S_m\uparrow$	$wF_\beta\uparrow$	MAE$\downarrow$	$E_m\uparrow$	$S_m\uparrow$	$wF_\beta\uparrow$	MAE$\downarrow$	$E_m\uparrow$	$S_m\uparrow$	$wF_\beta\uparrow$	MAE$\downarrow$
		CAMO				COD10K				NC4K			
DGNet[149]	1×	0.859	0.791	0.681	0.079	0.833	0.776	0.603	0.046	0.876	0.815	0.710	0.059
DGNet[149]	2×	0.861	0.791	0.682	0.080	0.832	0.778	0.606	0.045	0.875	0.814	0.711	0.059
+ PrObeD	2×	**0.871**	**0.797**	**0.702**	**0.071**	**0.869**	**0.803**	**0.661**	**0.037**	**0.900**	**0.838**	**0.755**	**0.049**

Table D.2 Ablation of dice loss with cross-entropy (CE) loss *vs*. cosine similarity.

Method	CAMO				COD10K				NC4K			
	$E_m\uparrow$	$S_m\uparrow$	$wF_\beta\uparrow$	MAE$\downarrow$	$E_m\uparrow$	$S_m\uparrow$	$wF_\beta\uparrow$	MAE$\downarrow$	$E_m\uparrow$	$S_m\uparrow$	$wF_\beta\uparrow$	MAE$\downarrow$
Dice + CE loss	0.831	0.782	0.688	0.084	0.810	0.795	0.646	0.045	0.874	0.817	0.721	0.060
Cosine similarity	**0.871**	**0.797**	**0.702**	**0.071**	**0.869**	**0.803**	**0.661**	**0.037**	**0.900**	**0.838**	**0.755**	**0.049**

2. Localization error (Loc): Classified correctly but localized incorrectly.

3. Both Classification and Localization error (Cls & Loc): Classified and localized incorrectly.

4. Duplicate detection error (Duplicate): Would be correct if not for a higher scoring detection.

5. Background error (Background): Detected background as foreground.

6. Missed target error (Missed): All undetected targets *i.e*.false negatives, which are not already covered by classification or localization errors.

Fig. D.2 shows the error analysis for three object detectors, namely, Faster-RCNN, YOLOv5, and DeTR. PrObeD improves the number of correct predictions of all three detectors, especially for Faster-RCNN, where the number of correct predictions increases by around 17%. For DeTR and YOLOv5, the improvement is less, which is evident from the less increase in correct predictions. The major improvement for all three detectors comes from classification and localization-related errors. All these errors decrease after PrObeD is applied to all the detectors. Further, Faster-RCNN, being an old detector, makes a lot of background errors, which are reduced by a significant margin after applying PrObeD. The gain is not much for DeTR and YOLOv5, which tend to make fewer background errors. Finally, one-stage detectors suffer mostly from the problem of duplicate detection, which is remedied by the PrObeD.

D.5 Potential Negative Societal Impact

PrObeD utilizes a proactive scheme to benefit object detection. Our approach can be considered a benign adversarial attack on object detectors. However, with a change in the objective function, PrObeD could also be used as an adversarial attack to deteriorate the performance of different object detectors. This might pose a threat to object detectors, whether used for GOD or COD, and some forms of adversarial training might be required to prevent the threat of adversarial attacks.

APPENDIX E

PROMARK APPENDIX

E.1 Multiple Watermark Configurations

We investigate the application of dual watermarks, each positioned on opposing sides of the image. This exploration raises a pivotal query: “Is the spatial positioning of watermarks critical to the performance?" To answer this, we ablate four distinct watermark configurations. As shown in Tab. E.1, there is a consistent performance across all watermark placements (left, right, top, bottom), thereby substantiating the spatial robustness of PrObeD in watermark positioning.

E.2 Watermark Robustness

We test our method against 14 different degradations (blur, various noises, fog, etc.), by adopting the evaluation protocol detailed in the RoSteALS [27]. We use 50 watermarked training images from LSUN dataset and use unconditional LDM with a strength of 30%. The average attribution accuracy for training and generated images across all 14 attacks is $90.21 \pm 7.63\%$ and $89.51 \pm 8.18\%$, as compared to 95.12% without any degradation, showing the robustness of our approach to multiple forms of watermark attack.

E.3 Possibility of Concept Leakage

We present multiple results where we attribute the images generated using non-watermarked data, for example via random latent code and conditional generation. We detect no retention of the watermark after noising or in random latent codes, with watermark detection accuracy of 50.56% (chance 50%) after noising for ≥ 900 timestamps or in random latent codes. The LDM generates an image from noise through inversion, and the watermark is added during this GenAI model inference process. Our decoder is employed independently to identify the concept. To prove this, we evaluate our model in Table 4 (main paper) for two more baselines, using held-out images (1) with no watermark encryption, and (2) encrypted with a different concept’s watermark. ProMark is able to attain an attribution accuracy of 94.32% and 94.01% respectively when evaluated with ground-truth concept watermark for both baselines compared to 95.60% reported for watermarked held-out data. Therefore, when inverting generating images that encrypt no watermark, or encrypt

Table E.1 Multi-concept attribution performance across different configurations.

Configuration		Attribution Accuracy (%) ↑		
Secret 1	Secret 2	Secret 1	Secret 2	Combined
Left	Right	95.61	93.31	90.12
Right	Left	95.52	93.35	90.19
Top	Bottom	95.66	93.70	90.01
Bottom	Top	95.02	93.46	90.73

incorrect watermark, the correct concept watermark is encrypted.

E.4 Computational Efficiency

We demonstrate the computation efficiency of ProMark during inference (running watermark decoder to perform causal attribution), which costs 5.6ms on one A100 GPU. Training with watermarked data adds negligible cost to generative model training. This is comparable to running inference on CLIP, or ALADIN to perform correlation based attribution (28.32 ms) but the additional cost of the embedding search is 87.91 ms for a dataset of 20K LSUN training images. ProMark therefore offers the advantage of both efficiency and causality for training data attribution. We will add this to the paper.

E.5 Additional Watermark Strength Analysis

Our research introduces a new paradigm in concept attribution for images classified under multiple concepts. We show the analysis of PSNR variation with watermark strength for the case of multi-concept attribution. The results are shown in Fig. E.1. Our findings indicate that, compared to single watermark cases, the PSNR for multi-concept images is marginally higher at equivalent watermark strengths. However, as expected, an increase in watermark strength generally leads to a decrease in PSNR.

Furthermore, we have visualized images from different datasets to showcase the extent of degradation caused by varying watermark strengths. As discussed in Sec. 4.5, the performance of our method improves with increased watermark strength. Nevertheless, this increase in strength leads to a decline in image quality, evidenced by the emergence of bubble-like artifacts in the images, as shown in Fig. E.2 (the watermark strength ranges from 0.1 to 1.0).

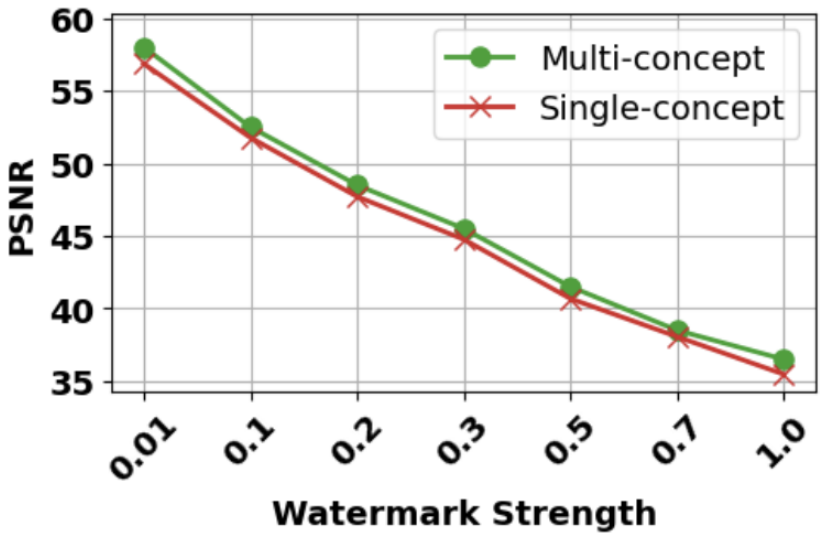


Figure E.1 PSNR *vs*.watermark strength for single vs multi-concept attribution.

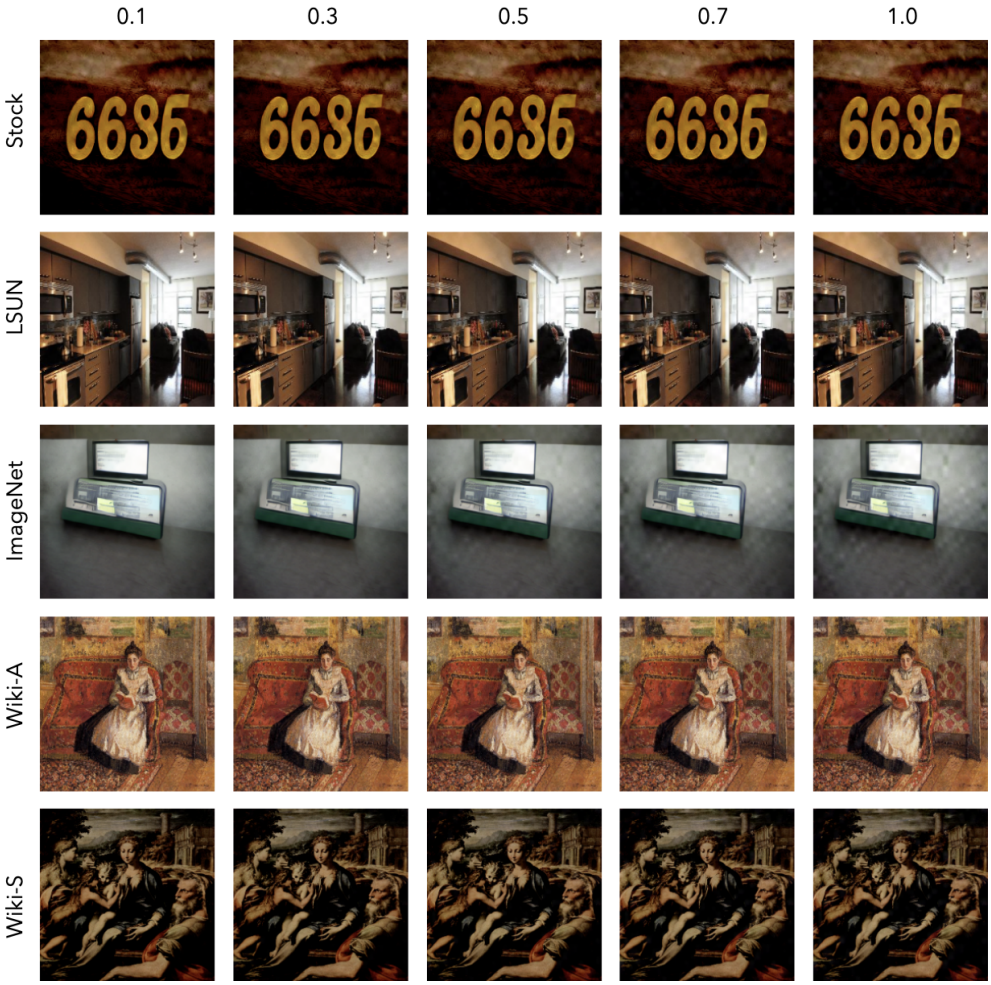


Figure E.2 Noise Strength visualization for different watermark strength.

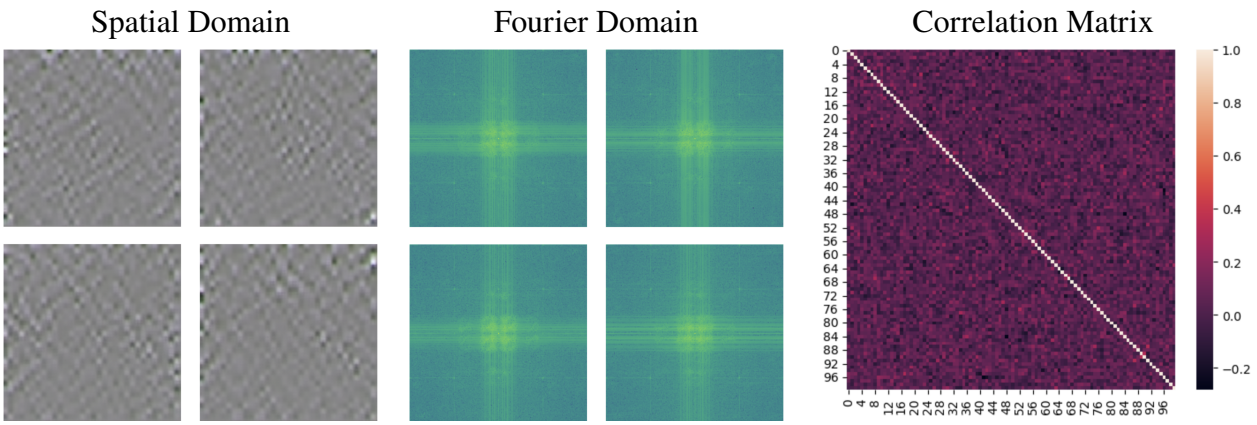


Figure E.3 Watermark Visualization: Spatial domain, Fourier domain and inter-watermark cosine similarity for 100 watermarks.

E.6 Watermark Discussion

We visualize some sample watermarks in both, spatial and frequency domain in Fig. E.3. These watermarks are converted from bit-sequences to spatial domain as described in Sec. 3.4. Visually, the watermarks appear indistinguishable from one another in both domains. Yet, their orthogonality is clearly demonstrated through the cosine similarity matrix, which we used to analyze 100 different watermarks. This matrix reveals that the inter-watermark cosine similarity is consistently close to zero, decisively indicating the orthogonal nature of these watermarks.

E.7 Implementation Details

We train PrObeD with LDM for $15K$ iterations with a batch size of 32, using 8 NVIDIA A100 GPUs for each experiment. We use the default parameters for optimizers as used in the official repository of [264]. The learning rate is set at $3.2e^{-5}$ for training LDM.

We further show the architecture for the generic decoder used for comparing against pretrained secret decoder shown in Fig. E.4. The generic decoder consists of 2 stem convolution layers and 10 convolution blocks. Each block consists of convolutional and batch normalization layers followed by ReLU activation.

E.8 More Sampled Images

We use multiple datasets for evaluating PrObeD. We sample images from the trained LDM for every class. We show some of the train and sampled images for the corresponding classes

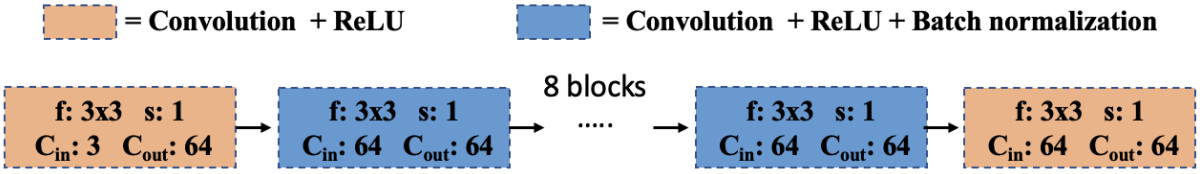


Figure E.4 Generic decoder architecture.

for different datasets in Figs. E.5 to E.8. We argue that PrObeD is able to perform attribution to different types of concepts, *i.e.*image templates (Fig. E.5), image style (Fig. E.8), style and content (Fig. E.6), and ownership (Fig. E.7). Therefore, proactive based causal methods perform attribution not only on the style or motif of the image as done by correlation based works, but also performs attribution to a variety of concepts proving it's generalizability.

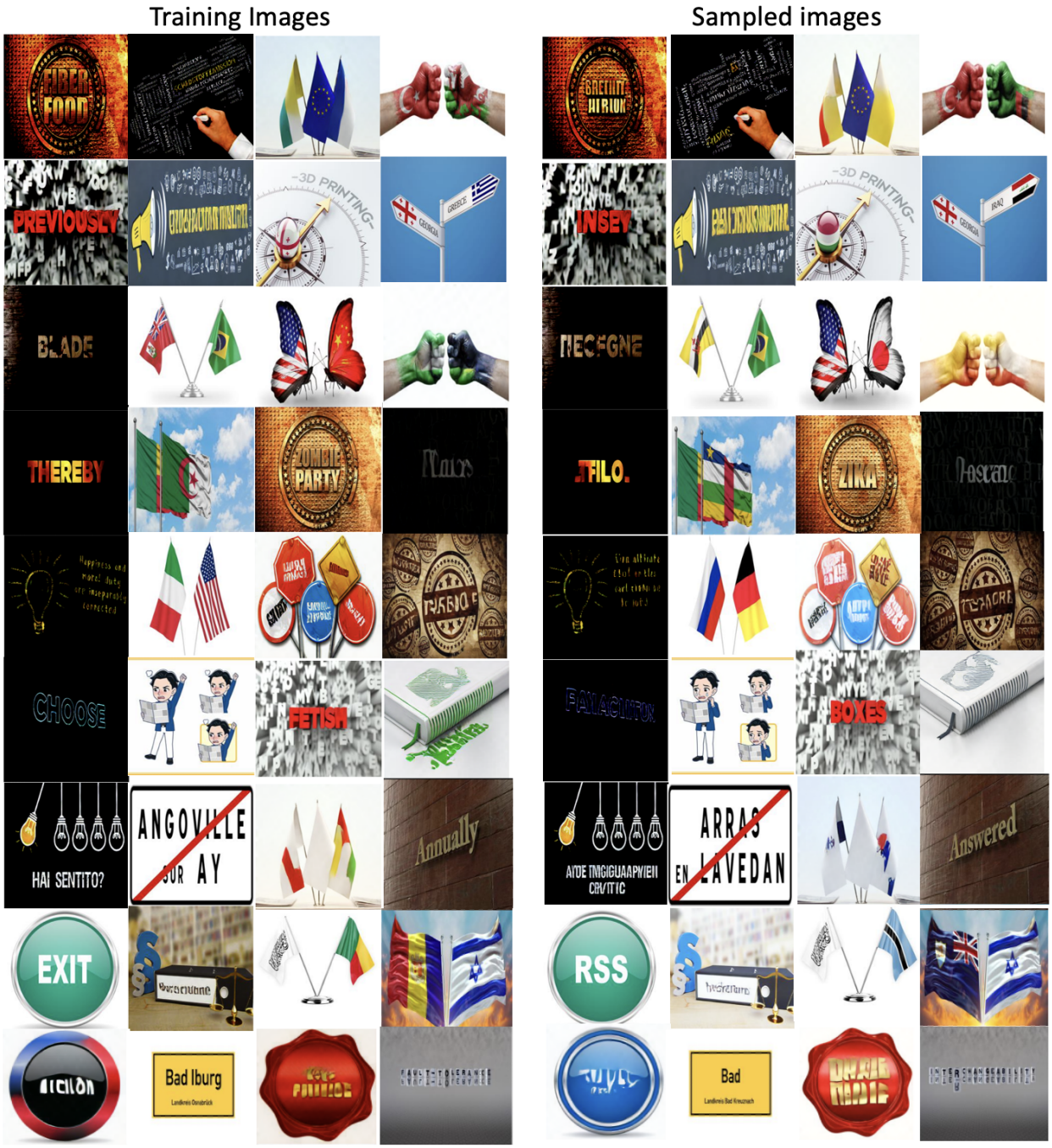


Figure E.5 Training and sampled images for stock dataset.

Figure E.6 Training and sampled images for BAM dataset.

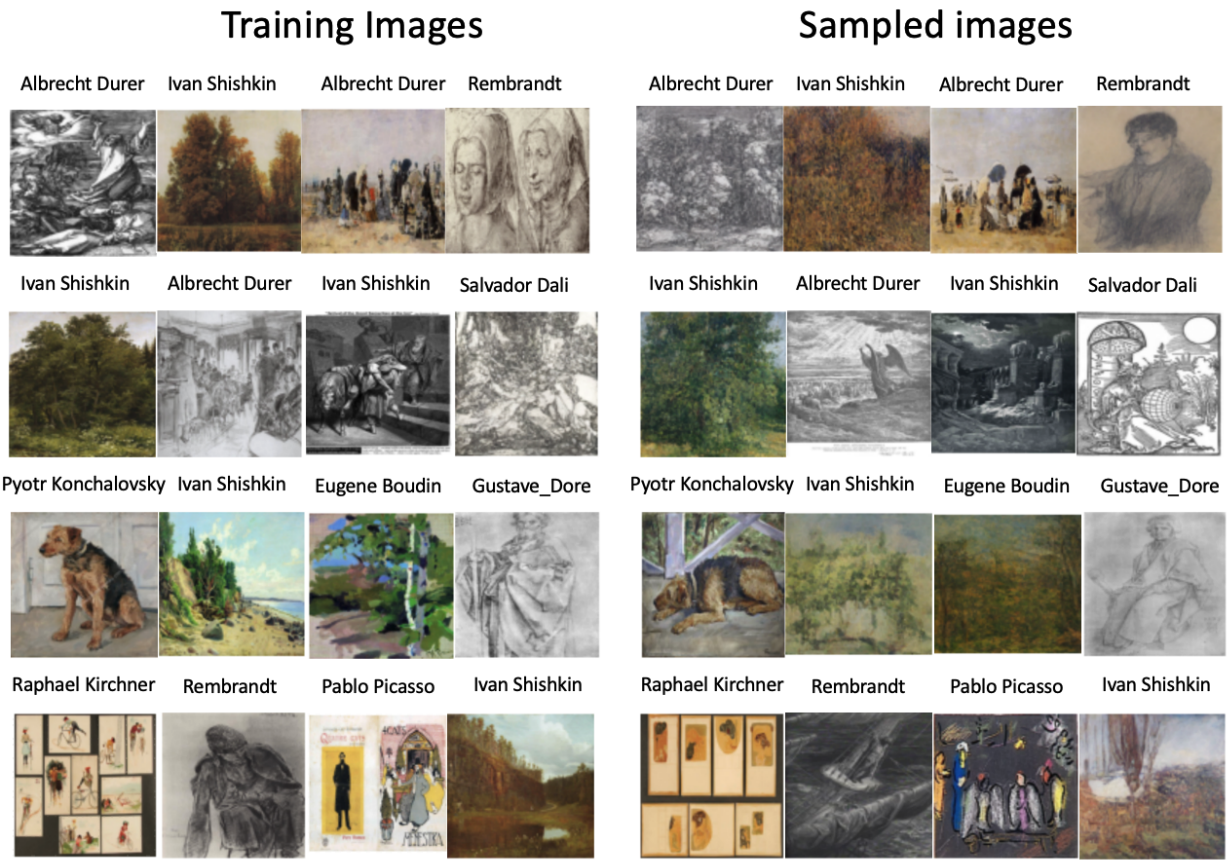


Figure E.7 Training and sampled images for wiki-a dataset.

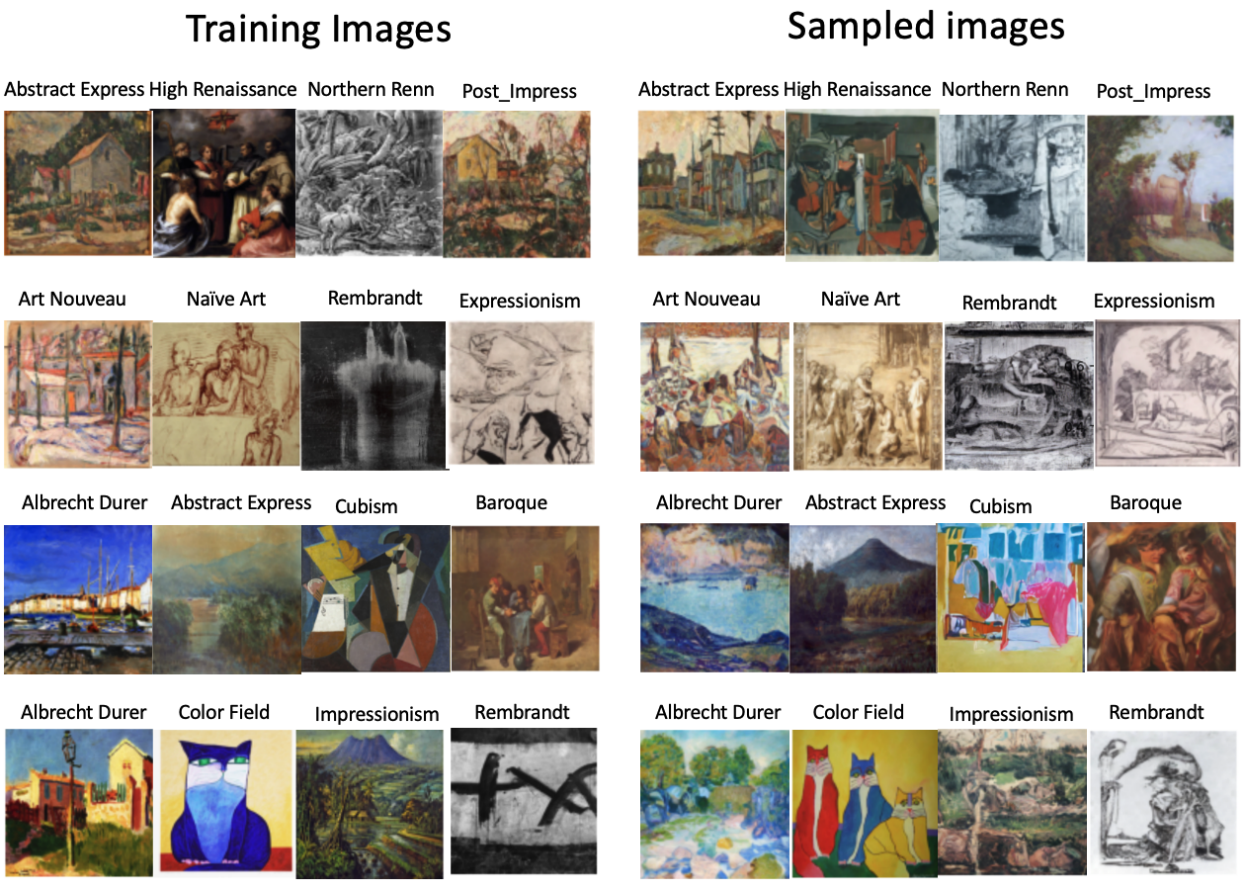


Figure E.8 Training and sampled images for wiki-s dataset.

APPENDIX F

CUSTOMMARK APPENDIX

F.1 Additional Experiments

Components Ablation. Tab. F.1 presents a comprehensive ablation study to analyze the contribution of individual components in CustomMark to its overall performance. The complete implementation of CustomMark achieves the highest performance across all metrics, with bit accuracy at 96.10%, attribution accuracy at 91.83%, clip score at 0.80, and csd score at 0.77. These results highlight the framework's ability to maintain robust attribution while preserving image quality. The performance drop observed when specific components are removed demonstrates the critical role each plays in the model's functionality.

The removal of the concept encoder results in a significant drop in performance, with bit accuracy and attribution accuracy reduced to 81.21% and 65.19%, respectively. This highlights the encoder's essential role in embedding bi secret information effectively. Similarly, disabling the mapper reduces bit accuracy to 93.10% and attribution accuracy to 87.11%, indicating its importance in maintaining precise attribution. The absence of attention finetuning from LDM moderately impacts the bit accuracy and attribution accuracy. However, qualitative performance is greatly reduced with csd score falling to 0.65, showcasing its role in style matching of clean and watermarked generated images during training.

The removal of regularization loss leads to minor performance degradation for attribution, but it impacts the qualitative metrics like the csd score, which drops to 0.71, demonstrating its role in ensuring consistency during watermark embedding, even tough it's only for initial iterations. Notably, the exclusion of style loss has the most detrimental effect on attribution accuracy, which falls dramatically to 40.16%, emphasizing its importance in preserving stylistic fidelity during the watermarking process. These results collectively validate the carefully designed architecture of CustomMark, where each component contributes significantly in achieving both robust attribution and high-quality image generation.

Sequential Learning Analysis. Fig. F.1 demonstrates the performance of individual concepts

Table F.1 Ablation study of various components of CustomMark for 10 concepts in training. [KEYS: att.:attention, reg. Regularization].

Changed	Bit Acc. (%)↑	Attribution Acc. (%) ↑	CLIP Score ↑	CSD Score ↑
CustomMark	**96.10**	**91.83**	**0.80**	**0.77**
– Concept Encoder	81.21	65.19	0.65	0.61
– Mapper	93.10	87.11	0.79	0.78
– Att. Finetune	95.16	90.88	0.71	0.65
– Reg. Loss	95.31	90.12	0.77	0.71
– Style Loss	75.10	40.16	0.66	0.62

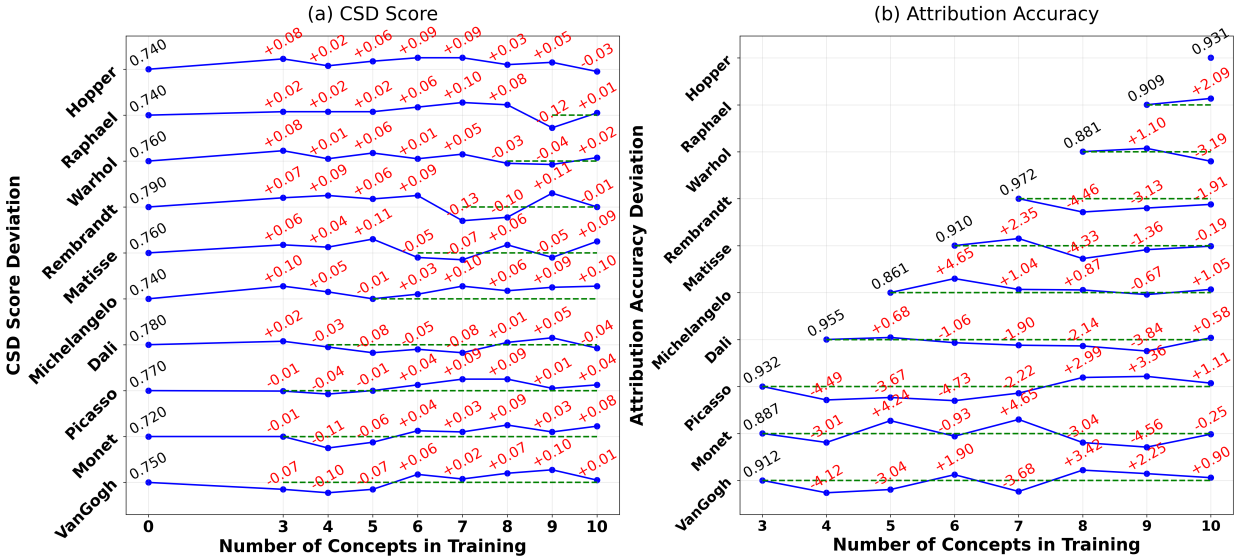


Figure F.1 Performance variation of individual concepts during sequential learning.

during sequential learning with CustomMark, evaluated through CSD score deviation and attribution accuracy as new concepts are added. The graphs illustrate how CustomMark maintains robust performance while adapting to an increasing number of concepts, showcasing its scalability and efficiency.

In the CSD score deviation plot(Fig. F.1(a)), the deviation remains minimal across most concepts, even as the number of concepts increases from 3 to 10. For instance, Hopper and Raphael exhibit only slight increases in deviation (+0.08 and +0.10, respectively) when additional concepts are introduced. This indicates that CustomMark effectively preserves stylistic fidelity for previously learned concepts while integrating new ones. Further, the CSD score before and after attribution remains almost similar. It decreases a little bit in start when the concept is introduced, but it gradually recovers to the original score. Notably, the deviation remains consistently low for

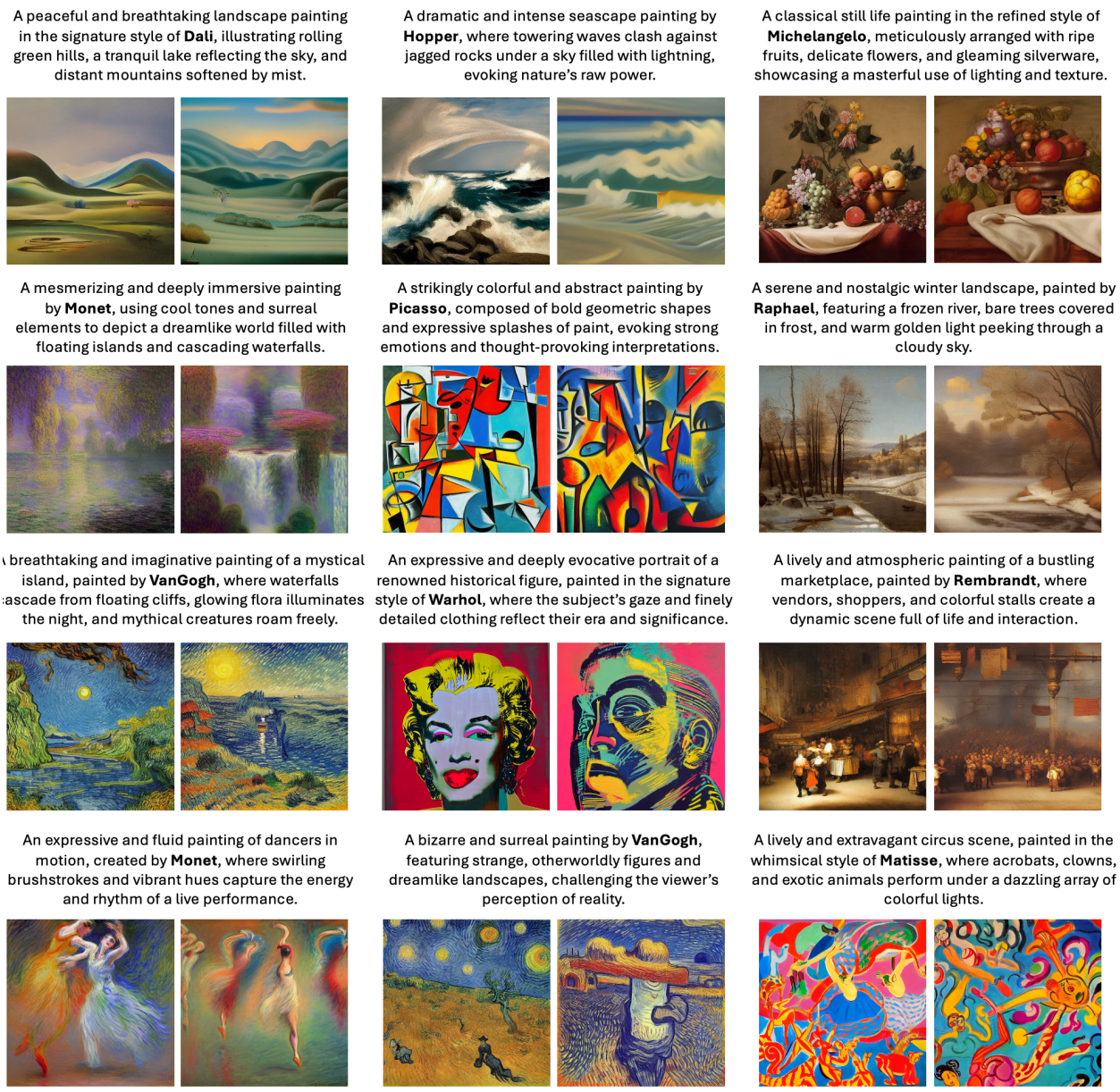
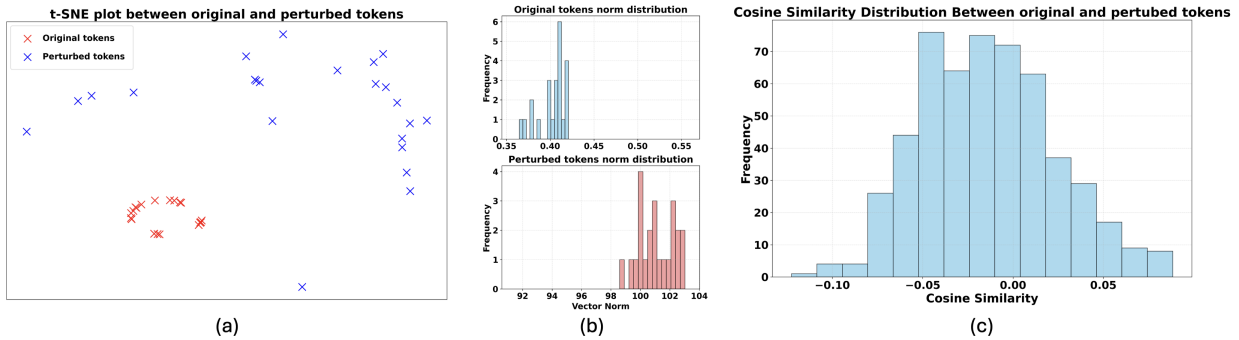


Figure F.2 Generated clean (left) and watermarked (right) images pairs for artists as concepts sampled using big and complicated prompts.

concepts like Picasso and Monet, further validating the robustness of the model.

The attribution accuracy plot (Fig. F.1(b)) highlights CustomMark's strong adaptability, with consistent attribution for new concepts added to training while maintaining high performance for earlier ones. This demonstrates that CustomMark's sequential learning approach effectively balances the retention of previously learned attributions with the incorporation of new ones, keeping in mind that CustomMark requires only about 10% additional training iterations per concept. These



(a) (b) (c)

Figure F.3 Analysis of original and perturbed tokens by (a) t-SNE plot, (b) norm distribution, and (c) distribution of cosine similarity between the two sets of embeddings.

results underline the practical viability of CustomMark in dynamic, real-world scenarios where the set of concepts evolves over time.

Complex Prompts. Fig. F.2 demonstrates the effectiveness of using complex and detailed prompts to generate images that accurately match the artistic styles of renowned painters. Each pair of images—one clean and one watermarked—illustrates that even though a long and complex prompt, CustomMark was able to insert the corresponding watermark onto the generated images as long as the concept token was perturbed.

Despite the complexity of the prompts, the generated images successfully capture the signature style of artists such as Dali, Monet, Van Gogh, Picasso, and Warhol. The results showcase precise interpretations of surreal, impressionistic, cubist, and other artistic movements, reinforcing the ability of GenAI model to replicate stylistic nuances into the watermarked images.

Analysis of Token Embedding. Fig. F.3 illustrates the analysis of original and perturbed tokens through t-SNE plots, norm distributions, and cosine similarity distributions. In the t-SNE plot (Fig. F.3(a)), the original tokens (red) and perturbed tokens (blue) demonstrate a clear separation, signifying that the perturbed tokens effectively diverge from their original counterparts. This divergence is critical for embedding unique watermarks and facilitating robust attribution. The norm distributions (Fig. F.3(b)) show that original tokens are centered very close to the norm 0 and exhibit a narrower range of vector norms, while perturbed tokens ahve high norms close to 100, and display a wider spread. This indicates that perturbations introduce divergence of the norm as compared to the original tokens and promotes controlled variability to the token space, contributing

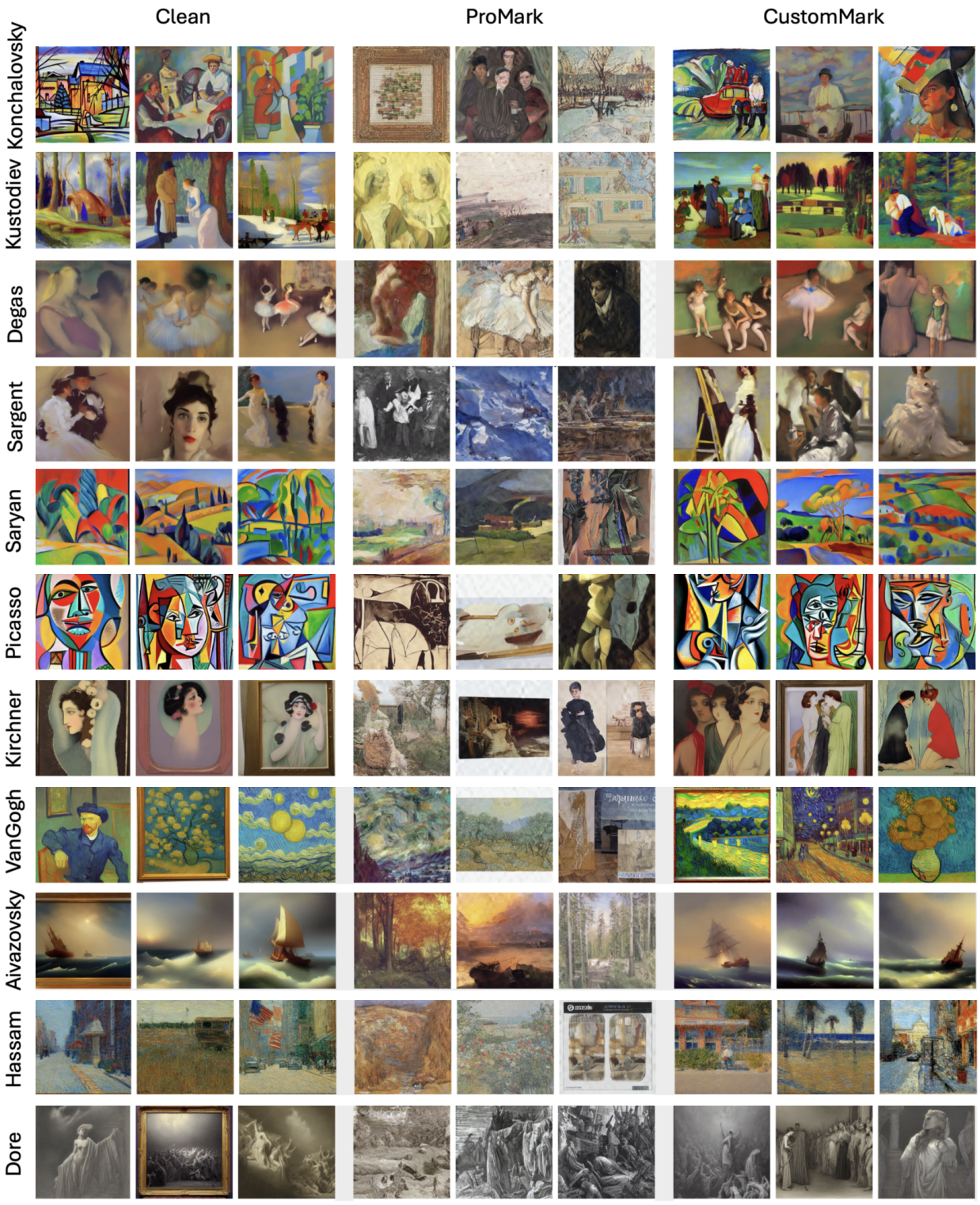


Figure F.4 Comparison with ProMark on WikiArt dataset.

to their distinctiveness. The cosine similarity distribution (Fig. F.3(c)) reveals that the similarity between original and perturbed tokens clusters around zero, highlighting that the perturbations

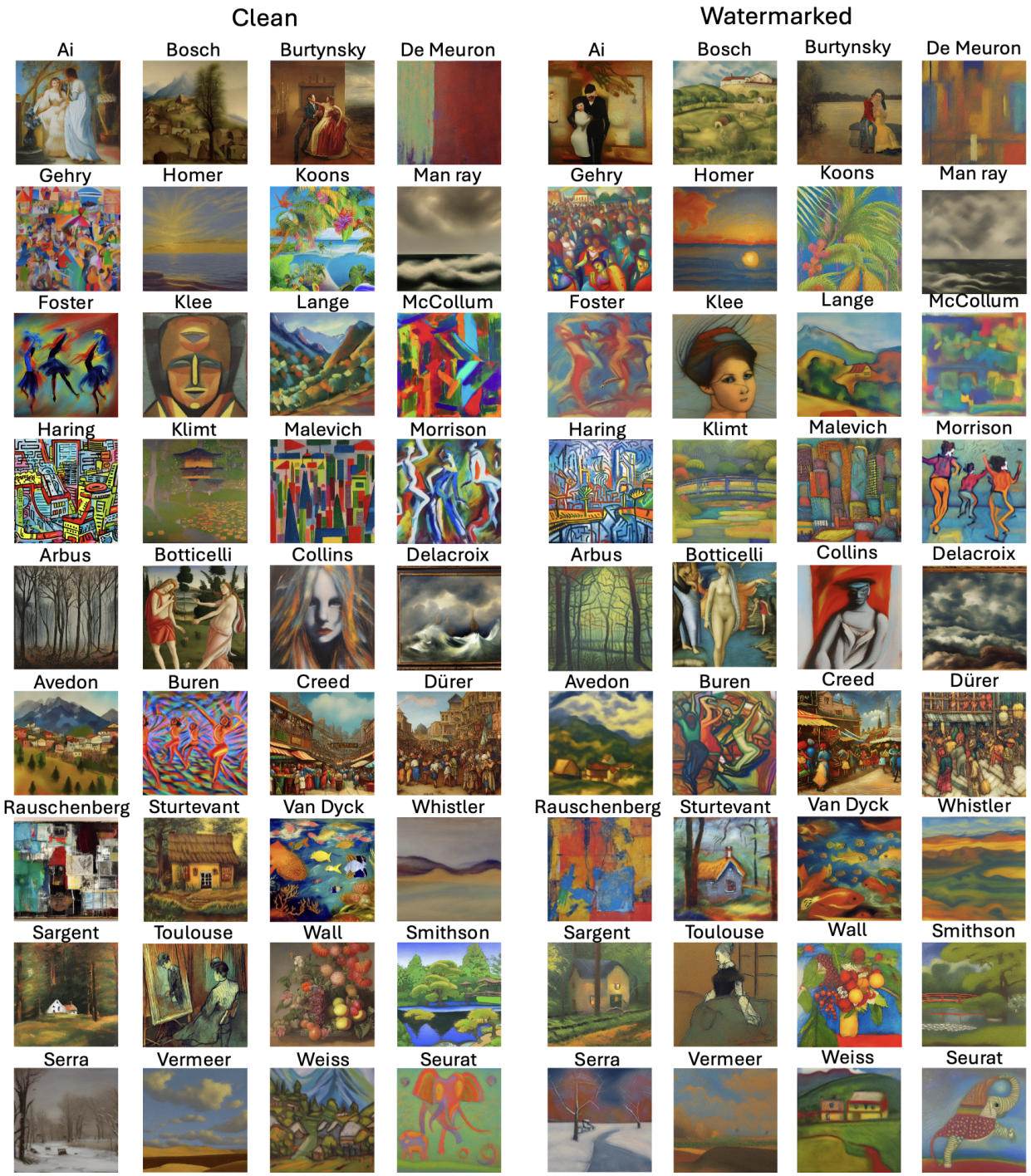


Figure F.5 Generated clean and watermarked images for artists as concepts sample by model trained for attributing 200 artists.

maintain minimal overlap with the original token directions—a necessary condition for ensuring effective attribution.

In our proposed approach, we apply the regularization loss during the initial iterations of

training. The regularization ensures that the perturbed tokens start with a meaningful deviation from the original tokens, setting a strong foundation for subsequent learning. To analyze this further, we don't switch off the regularization loss. We observe that continuing the regularization loss throughout the training process leads to the original and perturbed tokens becoming overly similar, undermining the ability to embed distinguishable watermarks and impairing attribution accuracy. With this approach, the model achieves a secret accuracy of 56.14% and an attribution accuracy of 1.54%, Therefore, we strategically switch off the regularization loss after the initial 200 iterations to allow the perturbed tokens to diverge as they want. This maintains the separation between original and perturbed tokens, ensuring that the model can generate robust watermarks while preserving the quality of attribution.

F.2 More Watermarked Samples

Fig. F.4 provides a comparative analysis between clean images, ProMark [4], and CustomMark on the WikiArt dataset, showcasing their performance in attribution while preserving artistic styles across a range of renowned artists from WikiArt dataset. CustomMark demonstrates superior style adaptation compared to ProMark, consistently maintaining the unique stylistic elements and visual fidelity of the original artworks. For artists such as Degas, Picasso, and Van Gogh, CustomMark effectively replicates the signature brushstrokes, color palettes, and composition techniques, resulting in outputs that remain faithful to their distinctive styles. In contrast, ProMark introduces noticeable bubble like artifacts and style distortions that detract from the visual coherence of the images. Similarly, for detailed and intricate works by artists like Sargent and Dore, CustomMark preserves the depth and intricacy, while ProMark struggles with fidelity, leading to degradation in fine details.

Fig. F.5 illustrates examples of clean and watermarked images for artists used as concepts, sampled from a model trained on 200 artists. Unlike Fig. F.4, which focused on the WikiArt dataset and showcased the performance of CustomMark for 23 artists, this figure demonstrates the scalability of the method when extended to a much larger and diverse set of artistic concepts. Across a wide range of styles, from Bosch and Klimt's classic depictions to Koons and Haring's contemporary designs, the watermarked images retain the stylistic essence of the clean images

while embedding imperceptible watermarks. Notably, the approach performs consistently well across different styles, capturing subtle details in works by artists such as Dürer, Toulouse, and Vermeer without introducing artifacts.

This comparison highlights CustomMark's ability to adapt seamlessly to various artistic styles, ensuring high-quality outputs that respect the original artistic intent, even when dealing with hundreds of distinct artistic styles. Its flexibility and fidelity make it a reliable solution for scenarios requiring robust watermarking without compromising on artistic integrity.

F.3 Limitations

While CustomMark offers an efficient solution for concept attribution, it has some limitations. First, it relies on the explicit mention of concepts in prompts, making attribution challenging when an artist's style is indirectly referenced or subtly embedded in the generated image. CustomMark finds it challenging to embed large bit sequences due to the mapper network being too simple for mapping bit sequence to noise perturbation. A sophisticated mapper network might address this issue. Additionally, CustomMark has not been tested on multi-concept scenarios, such as prompts combining multiple artists or blending diverse styles, leaving its robustness in such cases unexplored. Another limitation of CustomMark is its reliance on generated data for training. If the original GenAI model fails to adequately capture an artist's unique style or nuances, the improved model with attribution capabilities may struggle to accurately reflect or attribute that style in the generated images. These limitations highlight areas for future improvement to enhance the system's versatility and robustness.

F.4 Potential Social Impact

The potential social impact of CustomMark lies in its ability to foster a collaborative and transparent relationship between AI model developers and the artists. By introducing attribution capabilities, this algorithm empowers artists to gain recognition for the influence of their styles on AI-generated content, promoting a sense of agency and fairness. Unlike adversarial strategies that often pit creators against AI systems, CustomMark provides a constructive mechanism to bridge this divide, offering a signal for transparency without compromising creativity. By focusing on

attribution and transparency, CustomMark aims to support a harmonious integration of AI into the creative landscape, minimizing potential societal harm and building trust between artists and AI systems.

F.5 Implementation Details

Artist Lists. The list in Tab. F.2 presents a comprehensive compilation of 200 artists, which serves as the foundation for our attribution experiments. For experiments requiring a specific number of artists (top-k), we systematically select the top-k artists based on their numerical ranking in the table. This approach ensures consistency and reproducibility across various experimental setups. An ablation study is conducted by varying k as discussed in the main paper, with artists chosen accordingly. The scalability and robustness of the attribution methodology are assessed under a range of configurations, from smaller subsets of artists to the full set of 200 artists. Furthermore, we extend our evaluation beyond 200 artists by leveraging $1,000$ classes from ImageNet as additional concepts, demonstrating the scalability and adaptability of our approach.

Table F.2 Comprehensive List of 200 Artists.

Artist 1	**Artist 2**	**Artist 3**	**Artist 4**	**Artist 5**
1. Claude Monet	2. Pablo Picasso	3. Vincent van Gogh	4. Michelangelo Buonarroti	5. Raphael Sanzio
6. Rembrandt van Rijn	7. Salvador Dalí	8. Henri Matisse	9. Andy Warhol	10. Edward Hopper
11. Frida Kahlo	12. Edgar Degas	13. Paul Cézanne	14. Jackson Pollock	15. Edvard Munch
16. Gustav Klimt	17. Paul Gauguin	18. Pierre-Auguste Renoir	19. Johannes Vermeer	20. Caravaggio
21. Jan van Eyck	22. Édouard Manet	23. Georgia O'Keeffe	24. Francisco Goya	25. Albrecht Dürer
26. Sandro Botticelli	27. Titian	28. Diego Velázquez	29. Giotto di Bondone	30. El Greco
31. Peter Paul Rubens	32. Caspar David Friedrich	33. Wassily Kandinsky	34. Marc Chagall	35. Eugène Delacroix
36. Piet Mondrian	37. Roy Lichtenstein	38. Joan Miró	39. Hieronymus Bosch	40. Jean-Michel Basquiat
41. Gustave Courbet	42. Thomas Gainsborough	43. Jean-Auguste-Dominique Ingres	44. Élisabeth Vigée Le Brun	45. Artemisia Gentileschi

Table F.2 (cont'd)

Artist 1	**Artist 2**	**Artist 3**	**Artist 4**	**Artist 5**
46. Camille Pissarro	47. Georges Seurat	48. Diego Rivera	49. Henri de Toulouse-Lautrec	50. Édouard Vuillard
51. Berthe Morisot	52. Mary Cassatt	53. James Abbott McNeill Whistler	54. John Singer Sargent	55. William Blake
56. David Hockney	57. Keith Haring	58. Jasper Johns	59. Alfred Sisley	60. Jean-Baptiste-Camille Corot
61. Winslow Homer	62. Grant Wood	63. Paul Klee	64. Yayoi Kusama	65. Egon Schiele
66. Amedeo Modigliani	67. Fernand Léger	68. Giorgio de Chirico	69. Henri Rousseau	70. Max Ernst
71. Kazimir Malevich	72. Mark Rothko	73. René Magritte	74. Alphonse Mucha	75. Francis Bacon
76. Marcel Duchamp	77. Leonardo da Vinci	78. Lucian Freud	79. Anselm Kiefer	80. Joseph Beuys
81. Bridget Riley	82. Anish Kapoor	83. Damien Hirst	84. Tracey Emin	85. Ai Weiwei
86. Gerhard Richter	87. Jeff Koons	88. Takashi Murakami	89. Zhang Xiaogang	90. Jenny Saville
91. Kara Walker	92. Yoko Ono	93. Cindy Sherman	94. Louise Bourgeois	95. Barbara Kruger
96. Richard Serra	97. Donald Judd	98. Sol LeWitt	99. Frank Stella	100. Ellsworth Kelly
101. Robert Rauschenberg	102. Claes Oldenburg	103. Paolo Veronese	104. Pieter Bruegel	105. Anthony van Dyck
106. J.M.W. Turner	107. John Constable	108. John Everett Millais	109. Dante Gabriel Rossetti	110. Edward Burne-Jones
111. David Alfaro Siqueiros	112. Rufino Tamayo	113. Victor Vasarely	114. Kurt Schwitters	115. Andy Goldsworthy
116. Richard Long	117. Robert Smithson	118. Christo Javacheff	119. Walter Gropius	120. Robert Venturi
121. Jean Nouvel	122. Daniel Libeskind	123. Richard Rogers	124. Renzo Piano	125. Norman Foster
126. Bjarke Ingels	127. Frank Gehry	128. Santiago Calatrava	129. Toyo Ito	130. Frank Lloyd Wright
131. Alvar Aalto	132. Dominique Perrault	133. Luis Barragán	134. James Stirling	135. Peter Zumthor
136. Kazuyo Sejima	137. Kengo Kuma	138. Jacques Herzog	139. Pierre de Meuron	140. César Pelli
141. Christian de Portzamparc	142. Stefano Boeri	143. Wang Shu	144. Olafur Eliasson	145. Thomas Hirschhorn
146. Felix Gonzalez-Torres	147. Gilbert	148. Ugo Rondinone	149. Paul McCarthy	150. Cory Arcangel

Table F.2 (cont'd)

Artist 1	**Artist 2**	**Artist 3**	**Artist 4**	**Artist 5**
151. Elaine Sturtevant	152. Marcel Broodthaers	153. Maurizio Cattelan	154. Rirkrit Tiravanija	155. Allan McCollum
156. Glenn Ligon	157. Peter Fischli	158. David Weiss	159. Peter Doig	160. Thomas Schütte
161. Neo Rauch	162. Marlene Dumas	163. Felix Gonzalez-Torres	164. Lorna Simpson	165. Byrne Morrison
166. Glenn Martin	167. Dan Collins	168. Matthew Barney	169. Peter Hujar	170. Shirin Neshat
171. Thomas Demand	172. Alexander McQueen	173. Catherine Opie	174. Wolfgang Tillmans	175. Martin Creed
176. Olafur Eliasson	177. James Turrell	178. Bill Viola	179. Andreas Gursky	180. Lewis Baltz
181. Cindy Sherman	182. Man Ray	183. Bruce Nauman	184. Sol LeWitt	185. Richard Hamilton
186. James Rosenquist	187. Nam June Paik	188. Vito Acconci	189. Susan Rothenberg	190. Lawrence Weiner
191. Daniel Buren	192. Robert Gober	193. Adrian Piper	194. Katharina Fritsch	195. Christian Marclay
196. Richard Avedon	197. Jeff Wall	198. Edward Burtynsky	199. Julius Lange	200. Diane Arbus

Distortion Applied for Robustness Evaluation. For robustness evaluation in Fig. 8 (main paper), we apply several post-processing distortions. These augmentations are applied by following [88]. Below are the details:

1. **Color Jitter**: For the color jitter augmentation, we modified several aspects of the images. The brightness factor, contrast factor, and saturation were adjusted to a value of 0.3, while the hue factor was set to 0.1 to introduce controlled variations in the image colors.

2. **Crop and Resize**: For the crop and resize augmentation, we randomly extracted 384×384 blocks from the original 512×512 images and resized these blocks to 256×256, simulating different framing conditions.

3. **Gaussian Blur**: We applied Gaussian blur with a kernel size of (3,3) and a sigma value of (2.0, 2.0) to simulate soft-focus effects in the images.

4. **Gaussian Noise**: To introduce random noise, Gaussian noise was added to the images with a standard deviation of 0.05, creating a more realistic representation of noisy environments.

5. **JPEG compression**: We used a quality setting of 50 to simulate compression artifacts often encountered in real-world image data.

6. **Rotation**: This augmentation was randomly applied to the images within a range of 0 to 180 degrees to account for changes in orientation during training.

7. **Sharpness**: For the sharpness augmentation, we set the intensity to 1, enhancing the clarity of certain features within the images.

Architecture Details. We use several networks for designing CustomMark, which include concept encoder, secret mapper, and secret decoder. For concept encoder, a U-Net-inspired network designed for processing and transforming 1D sequential data is adopted. Initially, a fully-connected layer is maps the bit sequence to a feature vector which is concatenated with the token embedding. This is given as input to the encoder-decoder framework of U-Net to output the perturbed token embedding.

The mapper network is a feature transformation module designed to encode input indices into high-dimensional representations using an embedding-based approach. It employs a learnable embedding layer that maps input indices (*e.g.*16) to vectors in a higher-dimensional space (*e.g.*64). The embeddings are initialized orthogonally and scaled to maintain a unit standard deviation. During the forward pass, the network retrieves embeddings for all possible input indices, weights them element-wise based on the input tensor, and sums these weighted embeddings along the input dimension. The result is normalized by the square root of batch size and biased by adding 1, producing a robust high-dimensional representation for each input bit sequence.

Finally, we use the EfficientNet-B3 [295] architecture as its core backbone for secret decoder. The network is initialized with pre-trained weights from the ImageNet dataset for robust feature extraction. The final classifier layer of EfficientNet is replaced with a fully connected layer that outputs the predicted bit sequence.

Prompt Details. Following [88], we use various prompts for sampling clean and watermarked images which are used to train CustomMark. The collection of prompts is different, depending on the concept we attribute. We replace the "[name]" with the corresponding concept token. Below are the details:

1. Artists as concepts:
   - *"a painting, art by [name]"*
   - *"a rendering, art by [name]"*
   - *"a cropped painting, art by [name]"*
   - *"the painting, art by [name]"*
   - *"a clean painting, art by [name]"*
   - *"a dirty painting, art by [name]"*
   - *"a dark painting, art by [name]"*
   - *"a picture, art by [name]"*
   - *"a cool painting, art by [name]"*
   - *"a close-up painting, art by [name]"*
   - *"a bright painting, art by [name]"*
   - *"a cropped painting, art by [name]"*
   - *"a good painting, art by [name]"*
   - *"a close-up painting, art by [name]"*
   - *"a rendition, art by [name]"*
   - *"a nice painting, art by [name]"*
   - *"a small painting, art by [name]"*
   - *"a weird painting, art by [name]"*
   - *"a large painting, art by [name]"*

- *“A serene landscape painting in the style of [name]”*
- *“A bustling cityscape in the style of [name]”*
- *“A painting of a cozy cottage in the woods in the style of [name]”*
- *“A vibrant underwater scene in the style of [name]”*
- *“A whimsical painting of a flying elephant in the style of [name]”*
- *“A still life painting featuring fruit and flowers in the style of [name]”*
- *“A portrait of a famous historical figure in the style of [name]”*
- *“A painting of a dreamy night sky in the style of [name]”*
- *“A colorful abstract painting in the style of [name]”*
- *“A street scene from Paris in the style of [name]”*
- *“A depiction of a beautiful sunset over the ocean in the style of [name]”*
- *“A painting of a peaceful mountain village in the style of [name]”*
- *“An energetic painting of dancers in motion in the style of [name]”*
- *“A painting of a snow-covered winter scene in the style of [name]”*
- *“A painting of a tropical paradise in the style of [name]”*
- *“A painting of a magical forest filled with fantastical creatures in the style of [name]”*
- *“A painting of a dramatic stormy seascape in the style of [name]”*
- *“A portrait of a majestic lion in the style of [name]”*
- *“A painting of a romantic scene between two lovers in the style of [name]”*
- *“A painting of a serene Japanese garden in the style of [name]”*
- *“A painting of a bustling marketplace in the style of [name]”*
- *“A painting of a tranquil river scene in the style of [name]”*
- *“A painting of a fiery volcano eruption in the style of [name]”*
- *“A painting of a futuristic cityscape in the style of [name]”*

- *“A painting of a whimsical circus scene in the style of [name]”*
- *“A painting of a mysterious moonlit forest in the style of [name]”*
- *“A painting of a dramatic desert landscape in the style of [name]”*
- *“A portrait of a regal peacock in the style of [name]”*
- *“A painting of a mystical island in the style of [name]”*
- *“A painting of a lively carnival scene in the style of [name]”*

2. ImageNet classes as concepts:

- *“a photo of a [name]”*
- *“a rendering of a [name]”*
- *“a cropped photo of the [name]”*
- *“the photo of a [name]”*
- *“a photo of a clean [name]”*
- *“a photo of a dirty [name]”*
- *“a dark photo of the [name]”*
- *“a photo of my [name]”*
- *“a photo of the cool [name]”*
- *“a close-up photo of a [name]”*
- *“a bright photo of the [name]”*
- *“a cropped photo of a [name]”*
- *“a photo of the [name]”*
- *“a good photo of the [name]”*
- *“a photo of one [name]”*
- *“a close-up photo of the [name]”*
- *“a rendition of the [name]”*

– *“a photo of the clean [name]”*

– *“a rendition of a [name]”*

– *“a photo of a nice [name]”*

– *“a good photo of a [name]”*

– *“a photo of the nice [name]”*

– *“a photo of the small [name]”*

– *“a photo of the weird [name]”*

– *“a photo of the large [name]”*

– *“a photo of a cool [name]”*

– *“a photo of a small [name]”*

– *“a photo of a [name] playing sports”*

– *“a rendering of a [name] at a concert”*

– *“a cropped photo of the [name] cooking dinner”*

– *“the photo of a [name] at the beach”*

– *“a photo of a clean [name] participating in a marathon”*

– *“a photo of a dirty [name] after a mud run”*

– *“a dark photo of the [name] exploring a cave”*

– *“a photo of my [name] at graduation”*

– *“a photo of the cool [name] performing on stage”*

– *“a close-up photo of a [name] reading a book”*

– *“a bright photo of the [name] at a theme park”*

– *“a cropped photo of a [name] hiking in the mountains”*

– *“a photo of the [name] painting a mural”*

– *“a good photo of the [name] at a party”*

- *“a photo of one [name] playing an instrument”*
- *“a close-up photo of the [name] giving a speech”*
- *“a rendition of the [name] during a workout”*
- *“a photo of the clean [name] gardening”*
- *“a rendition of a [name] dancing in the rain”*
- *“a photo of a nice [name] volunteering at a charity event”*
- *“a photo of a [name] surfing a giant wave”*
- *“a rendering of a [name] skydiving over a scenic landscape”*
- *“a cropped photo of the [name] riding a rollercoaster”*
- *“the photo of a [name] rock climbing a steep cliff”*
- *“a photo of a clean [name] practicing yoga in a peaceful garden”*
- *“a photo of a dirty [name] participating in a paintball match”*
- *“a dark photo of the [name] stargazing at a remote location”*
- *“a photo of my [name] crossing the finish line at a race”*
- *“a photo of the cool [name] breakdancing in a crowded street”*
- *“a close-up photo of a [name] blowing out candles on a birthday cake”*
- *“a bright photo of the [name] flying a kite on a sunny day”*
- *“a cropped photo of a [name] ice-skating in a winter wonderland”*
- *“a photo of the [name] directing a short film”*
- *“a good photo of the [name] participating in a flash mob”*
- *“a photo of one [name] skateboarding in an urban park”*
- *“a close-up photo of the [name] solving a Rubik’s cube”*
- *“a rendition of the [name] fire dancing at a beach party”*
- *“a photo of the clean [name] planting a tree in a community park”*

- *"a rendition of a [name] performing a magic trick on stage"*
- *"a photo of a nice [name] rescuing a kitten from a tree"*

APPENDIX G

PIVOT APPENDIX

G.1 Additional Experiments.

LoRa in backbone vs head. The ablation study shown in Tab. G.1 evaluates the effectiveness of LoRA when applied to different parts of the baseline detector. When the LoRA layers application is shifted from the backbone to the head, the performance of both detectors, TSN and MViTv2, significantly decreases. Specifically, for TSN, the top-1 accuracy drops to 24.21% and the top-5 accuracy to 53.30%, while MViTv2 experiences a decline to 35.11% and 63.09% for top-1 and top-5 accuracy, respectively. This indicates that the backbone plays a critical role in extracting meaningful spatial-temporal features in video detection tasks, and adapting LoRA to the head limits its capacity to leverage these features effectively. These results highlight that LoRA's effectiveness is highly dependent on its application to critical regions of the model, particularly the backbone in this case, where it can better capture the temporal dynamics and spatial features necessary for improving action recognition.

Template Learning. Template learning plays a pivotal role in PiVoT by providing universal adaptability and efficiency in detector performance. When the framework transitions from frame-dependent templates to universal templates, a substantial degradation in accuracy is seen. For TSN, universal templates achieve a top-1 accuracy of 32.88% compared to 51.37% for frame-dependent templates, with a similar trend in top-5 accuracy (61.31% versus 78.71%). A similar degradation is seen in MViTv2, where universal templates result degraded performance. This demonstrates that while universal templates aim to generalize across frames, they cannot match the level of frame-specific optimization provided by PiVoT's default template approach.

Further, replacing learned templates with fixed templates also degrades performance. For TSN, top-1 accuracy falls to 26.19% and top-5 accuracy to 52.01%, while for MViTv2, top-1 accuracy decreases to 60.12% and top-5 accuracy to 86.11%. These results emphasize the necessity of dynamic, learned templates in PiVoT, as fixed templates fail to adapt to the variations in temporal dynamics and action-specific nuances inherent in video sequences. Overall, the original template

Table G.1 Ablation study of LoRA and template learning for PiVoT.

Changed	From→To	TSN [324] (%)↑		MViTv2 [182] (%)↑	
		Top-1	Top-5	Top-1	Top-5
PiVoT	-	**51.37**	**78.71**	**68.81**	**91.63**
LoRA	Backbone→Head	24.21	53.30	35.11	63.09
Template	Frame depend→Universal	32.88	61.31	57.70	85.14
	Learn→Fixed	26.19	52.01	60.12	86.11

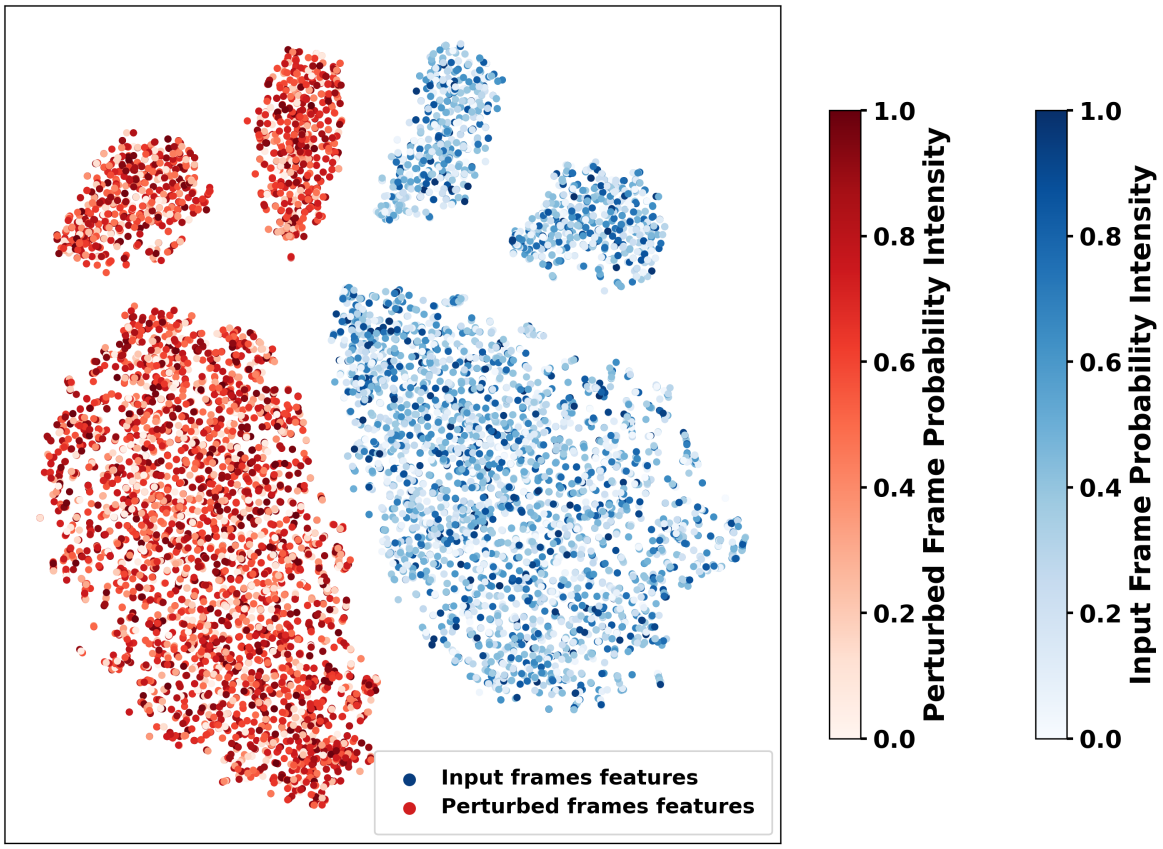
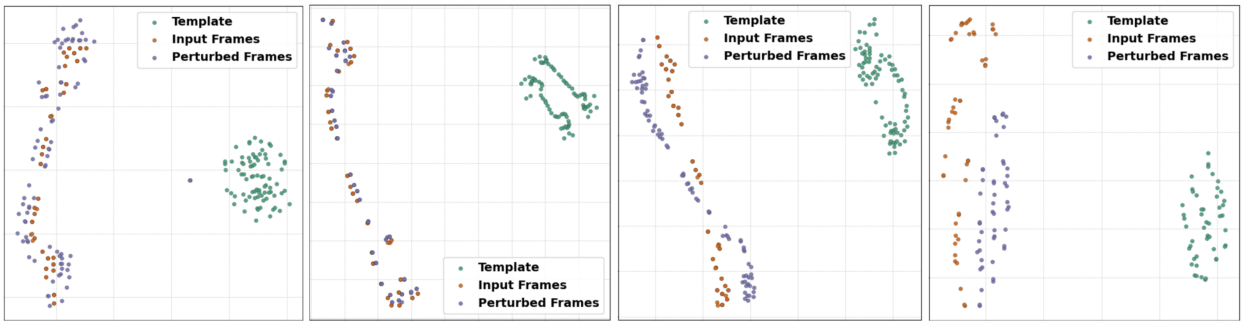


Figure G.1 Backbone feature distribution with color intensity varied by detector head logits confidence. Lighter color means detector is less confident and vice -versa.

learning mechanism in PiVoT proves to be critical for achieving superior performance compared to alternative template designs.

G.2 Template Analysis.

Fig. G.1 demonstrates the backbone feature distribution of input frames and perturbed frames when provided separately to the respective trained TSM detector. Perturbed frames, created by adding input frames with the generated template, exhibit a distinct separation in feature space



(a) TSN (b) TSM (c) AVA (d) MViTv2

Figure G.2 tSNE plot for all four detectors for input frames, perturbed frames, and estimated templates.

compared to the original input frames. The color intensity variation, corresponding to the detector logits, reveals a higher confidence for perturbed frames. This indicates that the template enhances the model's ability to extract discriminative features, leading to more confident predictions by the detector. The addition of templates aligns features more effectively with the task requirements, showcasing the utility of template-based enhancement for video-based tasks.

We further analyze the template enhancement in the t-SNE plots in Fig. G.2 which demonstrate that, at the frame level, the input and perturbed frames exhibit minimal differences in distribution, indicating that the addition of the template does not significantly alter the original frame content. However, when viewed at the feature level in Fig. G.1, there is a marked distinction between the input and perturbed frame feature distributions. This highlights that while the perturbation introduced by the template is subtle at the pixel level, it has a significant impact on the feature representations extracted by the detector.

This distinction underscores the effectiveness of the templates in enhancing task-relevant features. By subtly modifying the input frames, the templates guide the detector's feature space towards better alignment with the underlying action-specific semantics. This leads to enriched feature representations that improve the detector's performance without compromising the natural temporal and spatial consistency of the input frames. In essence, the templates act as an implicit augmentation mechanism, creating a more expressive and discriminative feature space for accurate action recognition and detection.

G.3 Limitations

The proposed PiVoT wrapper, while effective in enhancing video-based detectors, has certain limitations that needs discussion. First, the method requires training the wrapper specifically with each architecture, limiting its potential as a true plug-and-play solution. Developing a training-free implementation could significantly improve its ease of adoption across diverse models. Second, while performance gains are consistently observed across various tasks and datasets, the magnitude of these gains cannot be guaranteed. This variability comes from the differing architectures and dataset characteristics, which may influence the effectiveness of the wrapper. Another limitation lies in the visibility and influence of the templates. Currently, the templates have substantial freedom to enhance task-specific performance, but this poses challenges when perturbed videos need to be publicly shared or used outside controlled environments. Making the templates imperceptible would allow broader adoption to detectors, which might not have our wrapper installed. This means that if the invisible templates are already embedded in the video, the need for having PiVoT on every copy of detector will be eliminated. We leave all these useful directions for our future works.

G.4 Potential Social Impact

Video analysis tasks have diverse applications in the health industry, sports, entertainment, and surveillance, where accuracy is critical. For instance, in healthcare, fall detection systems in homes rely on accurate video-based monitoring to ensure timely assistance for patients. Similarly, in sports and entertainment, analyzing player movements with precision enhances performance evaluation and strategy development. Surveillance systems, which often operate in real-time, require high accuracy to detect anomalies effectively.

PiVoT, a template-based approach, offers a practical solution by significantly improving the accuracy of video-based detectors without substantially increasing the size or complexity of the system. This efficiency ensures that existing systems can be enhanced with minimal computational overhead, making the solution scalable for deployment in resource-constrained environments. By enabling high performance without large architectural modifications, the technique broadens accessibility and applicability, addressing the growing demand for robust, efficient, and accurate

video analysis across diverse real-world scenarios.

G.5 Implementation details

We provide the implementation details of our methods focusing on detector details, where the LoRA layers ar applied in the backbone, and the architecture details of our framework.

Detector Details. For all the detectors, we use the default config files as provided by the MMACTION2 toolbox [58]. Below are the names of the config files used for our experiments:

1. TSN: tsn_imagenet-pretrained-r50_8xb32-1x1x8-50e_sthv2-rgb and tsn_imagenet-pretrained-r50_8xb32-1x1x8-100e_kinetics400-rgb
2. TSM: tsm_imagenet-pretrained-r50_8xb16-1x1x8-50e_sthv2-rgb and tsm_imagenet-pretrained-r50_8xb16-1x1x8-50e_kinetics400-rgb
3. MViTv2: mvit-small-p244_k400-pre_16xb16-u16-100e_sthv2-rgb and mvit-small-p244_32xb16-16x4x1-200e_kinetics400-rgb
4. SlowFast: slowfast_kinetics400-pretrained-r50_8xb16-4x16x1-20e_ava21-rgb

The models are first trained on the respective dataset to reproduce the reported performance. This trained model is then further used with our PiVoT wrapper.

LoRA Application We use different backbones for different detectors. TSN and TSM use ResNet-50, SlowFast uses 3D ResNet-50, and MViTv2 uses multi-scale ViT backbone.

In this ResNet-50 backbone, LoRA is selectively applied to the convolutional layers within residual layer 3 and layer 4 of the network. Specifically, LoRA is integrated into the BasicBlock and Bottleneck modules for these layers. For the BasicBlock, LoRA is applied at the first and second convolutional layers (conv1 and conv2), adapting the channel dimensions through low-rank matrices. Similarly, in the Bottleneck module, LoRA is applied at each of the three convolutional layers (conv1, conv2, and conv3), modifying the input or output channels to enhance feature adaptation. By limiting LoRA to these deeper layers, the model focuses on refining high-level feature representations without overburdening the earlier stages of the network.

In the 3D ResNet-50 SlowFast LoRA network, LoRA is applied selectively to specific 3D convolutional layers within both the slow and fast pathways. Specifically, LoRA is applied to the conv1 layer and multiple convolutional layers across all ResNet stages (layer1, layer2, layer3, and layer4). This includes the main convolutional layers within each block, such as conv1, conv2, and conv3 in the bottleneck layers. This selective application focuses on enhancing the representational capacity of key layers without modifying the entire model.

Finally for MViTv2, LoRA is applied within the multi-scale attention mechanism to the query and key projections. Specifically, LoRA introduces two low-rank projection layers for the input features, reducing their dimensionality to a smaller rank r. These reduced representations are then projected back to the original dimensions using corresponding projection layers before being added to the standard query and key projections. This enables LoRA to enhance the adaptability of the attention mechanism while minimizing the additional parameter overhead. These LoRA adaptations are applied at each attention block across all transformer layers of the MViT model, making them integral to improving the model's representational capacity and flexibility.

Architecture Details. We employ a 3D attention-based U-Net network [338, 56, 143] to estimate templates from the input frames. LoRA layers are integrated into the detector's backbone, as previously described. The entire framework is trained end-to-end, with the detector initialized using a pretrained model.

APPENDIX H

REVERSE ENGINEERING OF GENERATIVE MODELS APPENDIX

H.1 Test sets for evaluation

The experiments described in the text were performed on four different test sets, each set containing twelve different GMs for the leave out testing. For test sets, we follow the distribution of GMs as follows: six GANs, two VAEs, two ARs, one NF and one AA model. We select this distribution because of the number of GMs of each type in our dataset which has 81 GANs, 13 VAEs, 11 ARs, 5 NFs and 6 AAs. The sets considered are shown in Tab. H.1.

H.2 Ground truth for GMs

We collected a fake face dataset of 116 GMs, each of them with $1,000$ generated images. We also collect the ground truth hyperparameters for network architecture and loss function types. Tab. H.2 shows the ground truth representation of the network architecture where different hyperparameters are of different data types. Therefore, we apply min-max normalization for the continuous type parameters to make all values in the range of $[0, 1]$. For multi-class and binary labels, we further show the feature value for different labels in Tab. H.3. Note that some parameters share the same values but with different meanings. For example, F14 and F15 represent skip connection and down-sampling respectively. Tab. H.4 shows the ground truth representation of the loss function types used to train each GM where all these values are binary indicating whether the particular loss type was used or not.

H.3 Network architecture

Fig. H.5 shows the network architecture used in different experiments. For GM parsing, our FEN has two stem convolution layers and 15 convolution blocks with each block having convolution, batch normalization and ReLU activation to estimate the fingerprint. The encoder in the PN has five convolution blocks with each block having convolution, pooling and ReLU activation. This is followed by two fully connected layers to output a 512 dimension feature vector which is further given as input to multiple branches to output different predictions. For continuous type parameters,

Table H.1 Test sets used for evaluation. Each set contains six GANs, two VAEs, two ARs, one AA and one NF.

GM	Set 1	Set 2	Set 3	Set 4
GM 1	ADV_FACES	AAE	BICYCLE_GAN	GFLM
GM 2	BETA_B	ADAGAN_C	BIGGAN_512	IMAGE_GPT
GM 3	BETA_TCVAE	BEGAN	CRGAN_C	LSGAN
GM 4	BIGGAN_128	BETA_H	FACTOR_VAE	MADE
GM 5	DAGAN_C	BIGGAN_256	FGSM	PIX2PIX
GM 6	DRGAN	COCOGAN	ICRGAN_C	PROG_GAN
GM 7	FGAN	CRAMERGAN	LOGAN	RSGAN_REG
GM 8	PIXEL_CNN	DEEPFOOL	MUNIT	SEAN
GM 9	PIXEL_CNN++	DRIT	PIXEL_SNAIL	STYLE_GAN
GM 10	RSGAN_HALF	FAST_PIXEL	STARGAN_2	SURVAE_FLOW_NONPOOL
GM 11	STARGAN	FVBN	SURVAE_FLOW_MAXPOOL	WGAN_DRA
GM 12	VAEGAN	SRFLOW	VAE_FIELD	YLG

we use two fully connected layers to output a 9-D network architecture. For discrete type parameters and loss function parameters, we use separate classifiers with three fully connected layers for every parameter to perform multi-class or binary classification.

For the deepfake detection task, we change the architecture of our FEN network as current deepfake manipulation detection requires much deeper networks. Thus, our FEN architecture has two stem convolution layers and 29 convolution blocks to estimate the fingerprint. For further classification, we use a shallow network of five convolution blocks followed by two fully connected layers.

For the image attribution task, we use the same FEN as used in model parsing, and a shallow network of two convolution blocks and two fully connected layers to perform multi-class classification.

Table H.2 Ground truth feature vector used for prediction of network architecture for all GMs. F1: # layers, F2: # convolutional layers, F3: # fully connected layers, F4: # pooling layers, F5: # normalization layers, F6: #filters, F7: # blocks, F8:# layers per block, F9: # parameters, F10: normalization type, F11: non-linearity type in last layer, F12: nonlinearity type in blocks, F13: up-sampling type, F14: skip connection, F15: downsampling.

GM	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
AAE	9	0	7	0	2	0	0	0	1593378	0	1	0	0	1	0
ACGAN	18	10	1	0	7	2307	5	3	4276739	0	1	1	0	1	0
ADAGAN_C	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0

Table H.2 (cont'd.)

GM	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
ADAGAN_P	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
ADV_FACES	45	23	1	1	20	2627	4	6	30000000	1	1	1	0	1	0
ALAE	33	25	8	0	0	4094	3	8	50200000	1	2	2	1	0	1
BEGAN	10	9	1	0	0	515	2	4	7278472	0	1	0	0	0	0
BETA_B	7	4	3	0	0	99	1	3	469173	3	3	1	0	1	1
BETA_H	7	4	3	0	0	99	1	3	469173	3	3	1	0	1	1
BETA_TCVAE	7	4	3	0	0	99	1	3	469173	3	3	1	0	1	1
BGAN	8	0	5	0	3	0	2	3	1757412	0	1	2	0	0	0
BICYCLE_GAN	25	14	1	0	10	4483	2	10	23680256	0	1	1	0	0	0
BIGGAN_128	63	21	1	0	41	6123	6	10	50400000	0	1	1	1	1	1
BIGGAN_256	75	25	1	0	49	7215	6	12	55900000	0	1	1	1	1	1
BIGGAN_512	87	29	1	0	57	8365	6	14	56200000	0	1	1	1	1	1
CADGAN	8	4	1	0	3	451	3	2	3812355	0	1	1	0	1	1
CCGAN	22	12	0	0	10	3203	2	9	29257731	0	1	1	1	1	1
CGAN	8	0	5	0	3	0	2	3	1757412	0	1	2	0	0	0
COCO_GAN	19	9	1	0	9	2883	3	4	50000000	0	1	1	0	0	0
COGAN	9	5	0	0	4	259	2	2	1126790	0	1	2	0	1	1
COLOUR_GAN	19	10	0	0	9	2435	2	9	19422404	0	1	1	0	1	1
CONT_ENC	19	11	0	0	8	5987	2	8	40401187	0	1	2	0	1	1
CONTRAGAN	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
COUNCIL_GAN	62	30	3	0	29	6214	2	10	69616944	1	1	1	0	1	0
CRAMER_GAN	9	4	1	0	4	454	2	3	9681284	0	1	1	0	1	0
CRGAN_C	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
CRGAN_P	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
CYCLEGAN	47	24	0	0	23	2947	4	9	11378179	1	1	1	1	1	1
DAGAN_C	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
DAGAN_P	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
DCGAN	9	4	1	0	4	454	2	3	9681284	0	1	1	0	1	0
DEEPFOOL	95	92	1	2	0	7236	4	10	22000000	2	0	1	1	0	0
DFCVAE	45	22	2	0	21	4227	4	7	2546234	0	3	2	0	0	1
DISCOGAN	21	12	0	0	9	3459	2	9	29241731	1	1	2	1	1	1
DRGAN	44	28	1	1	14	4481	3	8	18885068	0	1	0	0	1	1

Table H.2 (cont'd.)

GM	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
DRIT	19	10	0	0	9	1793	4	3	9564170	1	1	1	1	1	1
DUALGAN	25	14	1	0	10	4483	2	10	23680256	0	1	1	0	0	0
EBGAN	6	3	1	0	2	195	2	2	738433	0	1	2	0	0	1
ESRGAN	66	66	0	0	0	4547	5	4	7012163	2	2	2	1	0	0
FACTOR_VAE	7	4	3	0	0	99	1	3	469173	3	3	1	0	1	1
Fast pixel	17	9	0	0	8	768	2	8	4600000	0	3	0	0	1	0
FFGAN	39	19	1	1	19	3261	0	0	50000000	0	1	1	1	1	1
FGAN	5	0	3	0	2	0	2	2	2256401	0	3	1	0	1	0
FGAN_KL	5	0	3	0	2	0	2	2	2256401	0	3	1	0	1	0
FGAN_NEYMAN	5	0	3	0	2	0	2	2	2256401	0	3	1	0	1	0
FGAN_PEARSON	5	0	3	0	2	0	2	2	2256401	0	3	1	0	1	0
FGSM	95	92	1	2	0	7236	4	10	22000000	2	0	1	1	0	0
FPGAN	23	12	0	0	11	2179	2	6	53192576	1	1	1	0	0	1
FSGAN	37	20	0	1	16	2863	4	8	94669184	0	0	1	1	1	1
FVBN	28	0	28	0	0	0	1	1	307721	2	3	0	0	1	0
GAN_ANIME	25	18	0	0	7	2179	4	6	8467854	1	1	1	0	1	1
Gated_pixel_cnn	32	32	0	0	0	5433	3	10	3364161	2	3	2	1	1	0
GDWCT	79	27	40	1	11	5699	2	4	51965832	1	1	1	0	0	1
GFLM	95	92	1	2	0	7236	4	10	22000000	2	0	1	1	0	0
GGAN	8	4	1	0	3	451	3	2	3812355	0	1	1	0	1	1
ICRGAN_C	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
ICRGAN_P	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
Image_GPT	59	42	0	0	17	4673	7	8	401489	0	3	2	1	1	1
INFOGAN	7	3	1	0	3	195	2	2	1049985	0	1	2	0	0	1
LAPGAN	11	6	5	0	0	262	4	2	2182857	2	1	1	1	1	0
Lmconv	105	60	10	35	0	7156	15	5	46000000	2	3	0	1	1	1
LOGAN	35	14	13	1	7	4131	9	3	9416196	0	1	1	0	1	0
LSGAN	9	5	0	0	4	1923	2	4	23909265	0	1	1	0	0	0
MADE	2	0	2	0	0	0	1	2	12552784	2	3	0	0	1	0
MAGAN	9	5	0	0	4	963	2	3	11140934	0	1	1	0	1	0
MEMGAN	14	7	1	0	6	1155	3	4	4128515	0	1	1	0	1	0
MMD_GAN	9	4	1	0	4	454	2	3	9681284	0	1	1	0	1	0

Table H.2 (cont'd.)

GM	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
MRGAN	9	4	1	0	4	451	3	2	15038350	0	1	1	0	1	0
MSG_STYLE_GAN	33	25	8	0	0	4094	3	8	50200000	1	2	2	1	0	1
MUNIT	18	15	0	0	3	3715	2	6	10305035	1	0	1	1	1	1
NADE	1	0	1	0	0	0	1	1	785284	2	3	0	0	1	0
OCFGAN	9	4	1	0	4	454	2	3	9681284	0	1	1	0	1	0
PGD	95	92	1	2	0	7236	4	10	22000000	2	0	1	1	0	0
PIX2PIX	29	16	0	0	13	5507	2	13	54404099	1	1	2	1	1	1
PixelCNN	17	9	0	0	8	768	2	8	4600000	0	3	0	0	1	0
PixelCNN++	105	60	10	35	0	7156	15	5	46000000	2	3	0	1	1	1
PIXELDA	27	14	1	0	12	835	4	6	483715	0	1	1	1	0	0
PixelSnail	90	90	0	0	0	4051	3	10	40000000	2	0	3	0	1	0
PROG_GAN	26	25	1	0	0	4600	3	8	46200000	0	3	3	0	0	1
RGAN	7	3	1	0	3	195	2	2	1049985	0	1	2	0	0	1
RSGAN_HALF	8	4	1	0	3	899	3	2	13129731	0	1	1	0	1	0
RSGAN_QUAR	8	4	1	0	3	451	3	2	3812355	0	1	1	0	1	0
RSGAN_REG	8	4	1	0	3	1795	3	2	48279555	0	1	1	0	1	0
RSGAN_RES_BOT	15	7	1	0	7	963	3	4	758467	0	1	1	1	1	0
RSGAN_RES_HALF	15	7	1	0	7	1155	3	4	1201411	0	1	1	1	1	0
RSGAN_RES_QUAR	15	7	1	0	7	579	3	4	367235	0	1	1	1	1	0
RSGAN_RES_REG	15	7	1	0	7	2307	3	4	4270595	0	1	1	1	1	0
SAGAN	11	6	1	0	4	139	2	4	16665286	0	1	2	0	0	0
SEAN	19	16	0	0	0	5062	2	7	266907367	3	1	1	0	1	0
SEMANTIC	23	12	0	0	11	2179	2	6	53192576	1	1	1	0	0	1
SGAN	7	3	1	0	3	195	2	2	1049985	0	1	2	0	0	1
SNGAN	23	11	1	0	11	3871	4	5	10000000	0	1	1	0	1	0
SOFT_GAN	8	0	5	0	3	0	2	3	1757412	0	1	2	0	0	0
SRFLOW	66	66	0	0	2	4547	5	4	7012163	2	2	0	1	0	0
SRRNET	74	36	1	0	37	2819	4	16	4069955	0	1	1	0	1	1
STANDARD_VAE	7	4	3	0	0	99	1	3	469173	3	3	1	0	1	1
STARGAN	23	12	0	0	11	2179	2	6	53192576	1	1	1	0	0	1
STARGAN_2	67	26	12	4	25	4188	4	12	94008488	1	2	2	0	0	1
STGAN	19	10	0	0	9	2953	2	5	25000000	0	1	2	1	1	1

Table H.2 (cont'd.)

GM	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
STYLEGAN	33	25	8	0	0	4094	3	8	50200000	1	2	2	1	0	1
STYLEGAN_2	33	25	8	0	0	4094	3	8	59000000	1	2	2	1	0	1
STYLEGAN2_ADA	33	25	8	0	0	4094	3	8	59000000	1	2	2	1	0	1
SURVAE_FLOW_MAXPOOL	95	90	0	5	0	6542	2	20	25000000	2	0	0	0	0	0
SURVAE_FLOW_NONPOOL	90	90	0	0	0	6542	2	20	25000000	2	0	0	0	0	0
TPGAN	45	31	2	1	11	5275	0	0	27233200	0	3	3	0	1	1
UGAN	9	4	1	0	4	771	2	3	4850692	0	3	1	0	1	0
UNIT	43	22	0	0	21	4739	4	8	13131779	1	1	1	1	1	1
VAE_field	6	0	6	0	0	0	1	3	300304	2	3	0	0	0	0
VAE_flow	14	0	14	0	0	0	2	4	760448	2	3	0	0	0	0
VAEGAN	17	7	2	0	8	867	2	6	26396740	0	1	1	0	0	1
VDVAE	48	42	0	6	0	3502	3	13	41000000	2	0	2	1	1	1
WGAN	9	5	0	0	4	1923	2	4	23909265	0	1	1	0	0	0
WGAN_DRA	18	10	1	0	7	2307	5	3	4276739	0	1	1	0	1	0
WGAN_WC	18	10	1	0	7	2307	5	3	4276739	0	1	1	0	1	0
WGANGP	9	5	0	0	4	1923	2	4	23905841	0	1	1	0	0	0
YLG	33	20	1	2	10	5155	5	5	42078852	0	1	1	1	1	1

Table H.3 Feature value for different labels of multi-class and binary features.

Feature	Label	Value
Normalization type	0	Batch Normalization
	1	Instance Normalization
	2	Adaptive Instance Normalization
	3	No Normalization
Non-linearity type in last layer	0	ReLU
	1	Tanh
	2	Leaky_ReLU
	3	Sigmoid
Non-linearity type in blocks	0	ELU
	1	ReLU
	2	Leaky_ReLU
	3	Sigmoid
Upsampling type	0	Nearest Neighbour
	1	Deconvolution
Skip connection and downsampling	0	Feature used
	1	Feature not used

Table H.4 Ground truth feature vector used for prediction of loss type for all GMs.

GM	L_1	L_2	MSE	MMD	LS	WGAN	KL	Adversarial	Hinge	CE
AAE	1	0	0	0	0	0	0	0	0	1
ACGAN	1	0	0	0	0	0	0	0	0	1
ADAGAN_C	0	0	0	0	1	0	0	0	0	1
ADAGAN_P	0	0	0	0	1	0	0	0	0	0
ADV_FACES	1	0	1	0	1	0	0	0	0	0
ALAE	0	0	1	0	1	0	0	0	0	0
BEGAN	1	0	0	0	0	0	0	0	0	0
BETA_B	0	0	0	0	0	0	1	0	0	1
BETA_H	0	0	0	0	0	0	1	0	0	1
BETA_TCVAE	1	0	0	0	0	0	1	0	0	1
BGAN	0	0	0	0	1	0	0	0	0	1
BICYCLE_GAN	1	0	1	0	0	0	1	0	0	0
BIGGAN_128	1	0	0	0	0	0	0	0	0	0
BIGGAN_256	1	0	0	0	0	0	0	0	0	0
BIGGAN_512	1	0	0	0	0	0	0	0	0	0
CADGAN	0	0	0	1	0	0	0	0	0	0
CCGAN	0	0	0	0	1	0	0	1	0	0
CGAN	0	0	1	0	1	0	0	0	0	0

Table H.4 (cont'd).

GM	L_1	L_2	MSE	MMD	LS	WGAN	KL	Adversarial	Hinge	CE
COCO_GAN	1	1	0	0	0	1	0	0	1	0
COGAN	0	0	0	0	1	0	0	0	0	0
COLOUR_GAN	1	0	0	0	1	0	0	0	0	0
CONT_ENC	0	1	0	0	1	0	0	0	0	0
CONTRAGAN	1	0	0	0	0	0	0	1	0	1
COUNCIL_GAN	1	0	1	0	1	0	0	0	0	0
CRAMER_GAN	0	0	0	0	0	1	0	0	0	0
CRGAN_C	1	1	0	0	0	0	0	0	0	1
CRGAN_P	1	1	0	0	0	0	0	0	0	0
CYCLEGAN	1	0	0	0	1	0	0	0	0	0
DAGAN_C	1	0	0	0	0	0	0	0	0	1
DAGAN_P	1	0	0	0	0	0	0	0	0	0
DCGAN	0	0	0	0	0	0	0	0	0	1
DEEPFOOL	1	1	0	0	0	0	0	0	0	0
DFCVAE	0	1	0	0	0	0	1	0	0	1
DISCOGAN	1	0	0	0	1	0	0	0	0	0
DRGAN	0	0	0	0	1	0	0	0	0	1
DRIT	1	0	0	0	1	0	0	0	0	1
DUALGAN	1	0	0	0	0	1	0	0	0	0
EBGAN	0	1	0	0	1	0	0	1	1	0
ESRGAN	1	0	0	0	1	0	0	0	0	0
FACTOR_VAE	1	0	0	0	0	0	1	0	0	1
Fast pixel	0	0	0	0	0	0	0	0	0	1
FFGAN	1	1	0	0	1	0	0	0	0	1
FGAN	0	0	0	0	1	0	0	1	0	0
FGAN_KL	1	0	0	0	0	0	0	0	0	0
FGAN_NEYMAN	0	1	0	0	0	0	0	0	0	0
FGAN_PEARSON	0	0	1	0	0	0	0	0	1	0
FGSM	0	0	0	0	1	0	0	0	0	0
FPGAN	1	1	0	0	1	0	0	0	0	1
FSGAN	1	0	0	0	1	0	0	0	0	1
FVBN	0	0	0	0	0	0	0	0	0	1
GAN_ANIME	1	1	0	0	0	1	0	0	1	0
Gated_pixel_cnn	0	0	0	0	0	0	0	0	0	1
GDWCT	1	0	1	0	0	0	0	0	1	0
GFLM	0	0	1	0	0	0	0	0	0	1

Table H.4 (cont'd).

GM	L_1	L_2	MSE	MMD	LS	WGAN	KL	Adversarial	Hinge	CE
GGAN	1	0	0	0	0	0	0	0	0	0
ICRGAN_C	1	1	0	0	0	0	0	0	0	1
ICRGAN_P	1	1	0	0	0	0	0	0	0	0
Image_GPT	0	0	0	0	0	0	0	0	0	1
INFOGAN	0	0	1	0	1	0	0	0	0	1
LAPGAN	0	0	0	0	1	0	0	0	0	0
Lmconv	0	0	0	0	0	0	0	0	0	1
LOGAN	1	1	0	0	0	0	0	1	0	0
LSGAN	0	0	1	0	0	0	0	0	1	0
MADE	0	0	0	0	0	0	0	0	0	1
MAGAN	0	0	1	0	0	0	0	0	0	0
MEMGAN	0	0	0	0	1	0	0	0	0	0
MMD_GAN	1	0	0	1	0	0	0	0	0	0
MRGAN	0	0	1	0	1	0	0	0	0	0
MSG_STYLE_GAN	0	0	0	0	1	0	0	0	0	0
MUNIT	1	0	0	0	1	0	0	0	0	0
NADE	0	0	0	0	0	0	0	0	0	1
OCFGAN	0	0	0	1	0	0	0	0	1	0
PGD	1	1	0	0	0	0	0	0	0	0
PIX2PIX	1	0	0	0	1	0	0	0	0	0
PixelCNN	0	0	0	0	0	0	0	0	0	1
PixelCNN++	0	0	0	0	0	0	0	0	0	1
PIXELDA	0	0	0	0	1	0	0	0	1	1
PixelSnail	0	0	0	0	0	0	0	0	0	1
PROG_GAN	0	0	0	0	0	1	0	0	1	0
RGAN	0	0	0	0	0	1	0	0	0	0
RSGAN_HALF	0	0	0	0	0	0	0	0	0	1
RSGAN_QUAR	0	0	0	0	0	0	0	0	0	1
RSGAN_REG	0	0	0	0	0	0	0	0	0	1
RSGAN_RES_BOT	0	0	0	0	0	0	0	0	0	1
RSGAN_RES_HALF	0	0	0	0	0	0	0	0	0	1
RSGAN_RES_QUAR	0	0	0	0	0	0	0	0	0	1
RSGAN_RES_REG	0	0	0	0	0	0	0	0	0	1
SAGAN	0	0	0	0	1	0	0	0	0	0
SEAN	1	0	0	0	1	0	0	0	0	0
SEMANTIC	0	1	0	0	1	0	0	0	0	0

Table H.4 (cont'd).

GM	L_1	L_2	MSE	MMD	LS	WGAN	KL	Adversarial	Hinge	CE
SGAN	0	0	0	0	1	0	0	0	0	1
SNGAN	0	0	0	0	1	0	0	1	0	0
SOFT_GAN	0	0	0	0	1	0	0	0	0	0
SRFLOW	1	0	0	0	0	0	0	0	0	1
SRRNET	0	1	1	0	1	0	0	0	0	1
STANDARD_VAE	0	0	0	0	0	0	1	0	0	1
STARGAN	1	0	0	0	1	0	0	0	0	1
STARGAN_2	1	0	0	0	1	0	0	0	0	0
STGAN	1	0	0	0	1	1	0	0	0	0
STYLEGAN	0	1	0	0	0	1	0	0	0	0
STYLEGAN_2	0	1	0	0	1	0	0	0	1	0
STYLEGAN2_ADA	0	1	0	1	1	0	0	0	1	0
SURVAE_FLOW_MAXPOOL	0	0	0	0	0	0	1	0	0	1
SURVAE_FLOW_NONPOOL	0	0	0	0	0	0	1	0	0	1
TPGAN	1	0	0	0	0	1	0	0	0	0
UGAN	0	0	0	0	1	0	0	0	0	0
UNIT	0	0	0	0	1	0	1	0	0	0
VAE_field	0	0	0	0	0	0	1	0	0	1
VAE_flow	0	0	0	0	0	0	1	0	0	1
VAEGAN	1	0	0	0	1	0	1	0	0	0
VDVAE	0	0	0	0	0	0	1	0	0	1
WGAN	0	0	0	0	0	1	0	0	0	0
WGAN_DRA	0	0	1	0	0	1	0	0	0	0
WGAN_WC	0	0	0	0	0	1	0	0	0	0
WGANGP	0	1	0	0	0	1	0	0	0	0
YLG	0	0	0	0	0	1	0	0	0	0

Table H.5 Ground truth feature vector used for prediction of network architecture for evaluation on diffusion models. F1: # layers, F2: # convolutional layers, F3: # fully connected layers, F4: # pooling layers, F5: # normalization layers, F6: #filters, F7: # blocks, F8:# layers per block, F9: # parameters, F10: normalization type, F11: non-linearity type in last layer, F12: nonlinearity type in blocks, F13: up-sampling type, F14: skip connection, F15: downsampling.

GM	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
ADM	134	122	12	0	0	5000	8	12	554000000	1	1	1	1	1	1
ADM-G	134	122	12	0	0	5000	8	12	600000000	1	1	1	1	1	1
DDPM	134	122	12	0	0	5000	8	12	554000000	1	1	1	1	1	1
DDIM	134	122	12	0	0	5000	8	12	554000000	1	1	1	1	1	1
LDM	134	122	12	0	0	5000	8	12	554000000	1	1	1	1	1	1
Stable-Diffusion	94	84	10	0	0	5000	8	12	552000000	1	1	1	1	1	1
GLIDE-Diffusion	90	80	10	0	0	5000	8	12	270000000	1	1	1	1	1	1

Table H.6 Ground truth feature vector used for prediction of loss type for evaluation on diffusion models.

GM	L_1	L_2	MSE	MMD	LS	WGAN	KL	Adversarial	Hinge	CE
ADM	0	0	1	0	0	0	0	0	0	1
ADM-G	0	0	1	0	0	0	0	0	0	0
DDPM	0	0	1	0	0	0	0	0	0	0
DDIM	0	0	1	0	0	0	0	0	0	1
LDM	1	0	1	0	0	0	0	0	0	0
Stable-Diffusion	0	0	1	1	0	0	0	0	0	0
GLIDE-Diffusion	0	0	1	1	0	0	0	0	0	1

Table H.7 Test sets used for evaluation on diffusion models.

GM	Set 1	Set 2	Set 3	Set 4
GM 1	ADM	DDPM	Stable-diffusion	GLIDE-Diffusion
GM 2	ADM-G	DDIM	ADM-G	DDIM
GM 3	DDPM	LDM	GLIDE-Diffusion	LDM

Table H.8 Test sets used for coordinated misinformation attacks.

Type	GM 1	GM 2	GM 3	GM 4	GM 5	GM 6	GM 7	GM 8	GM 9	GM 10	GM 11	GM 12	GM 13	GM 14	GM 15
Seen GMs	BETA_B	GAN_ANIME	RGAN	DRIT	PIX2PIX	UNIT	SAGAN	DFCVAE	LOGAN	DAGAN_C	SRRNET	LSGAN	BIGGAN_128	RSGAN_HALF	BICYCLE_GAN
Unseen GMs	DRGAN	RSGAN_REG	MAGAN	MADE	ALAE	ACGAN	WGAN	TPGAN	LAPGAN	BETA_TCVAE	BGAN	FFGAN	CRGAN_C	FGAN	STARGAN

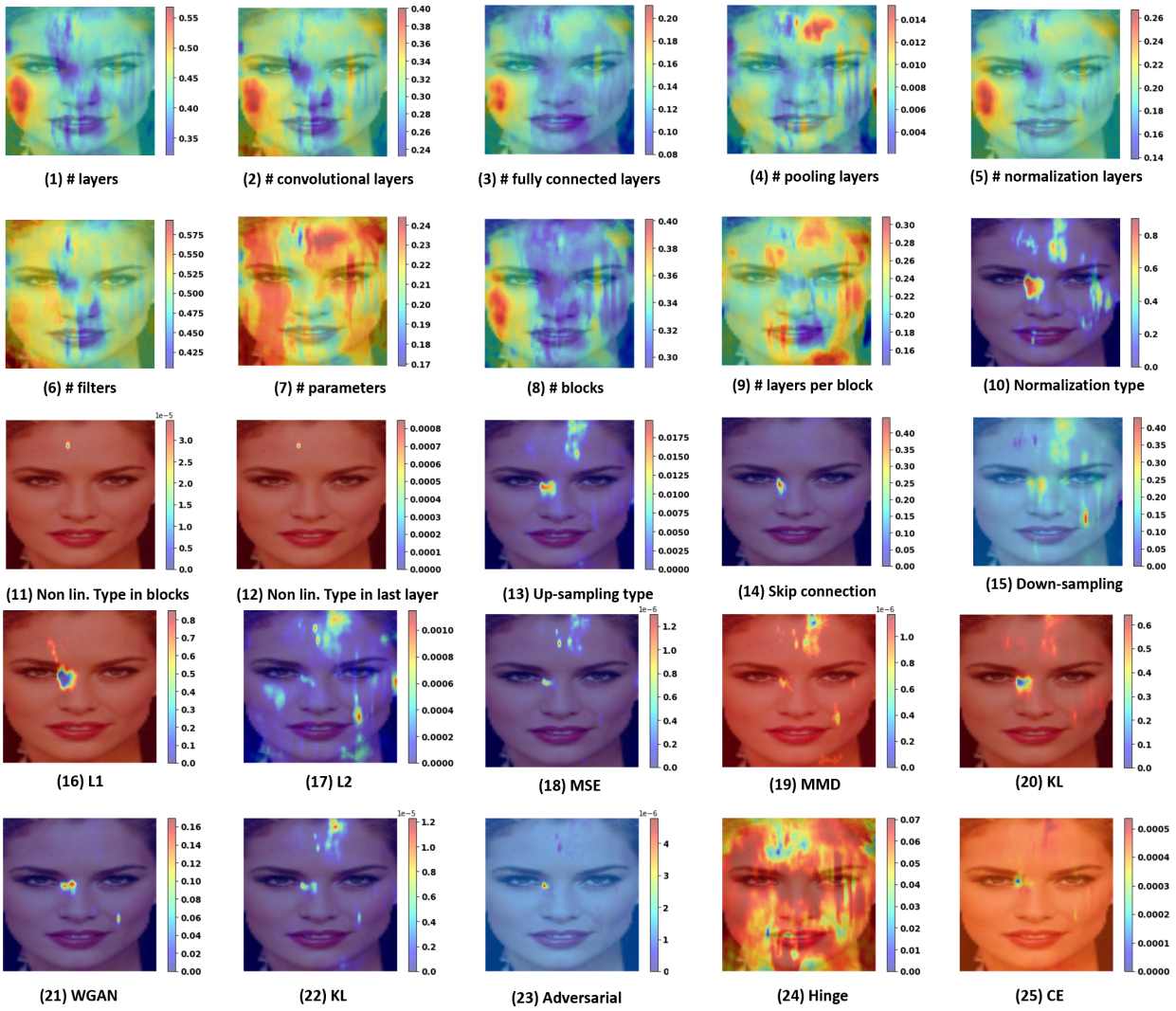


Figure H.1 Feature heatmap for each feature in network architecture and loss function predicted feature vector for face data. Each heatmap provides the importance of the region in the estimation of the respective parameter.

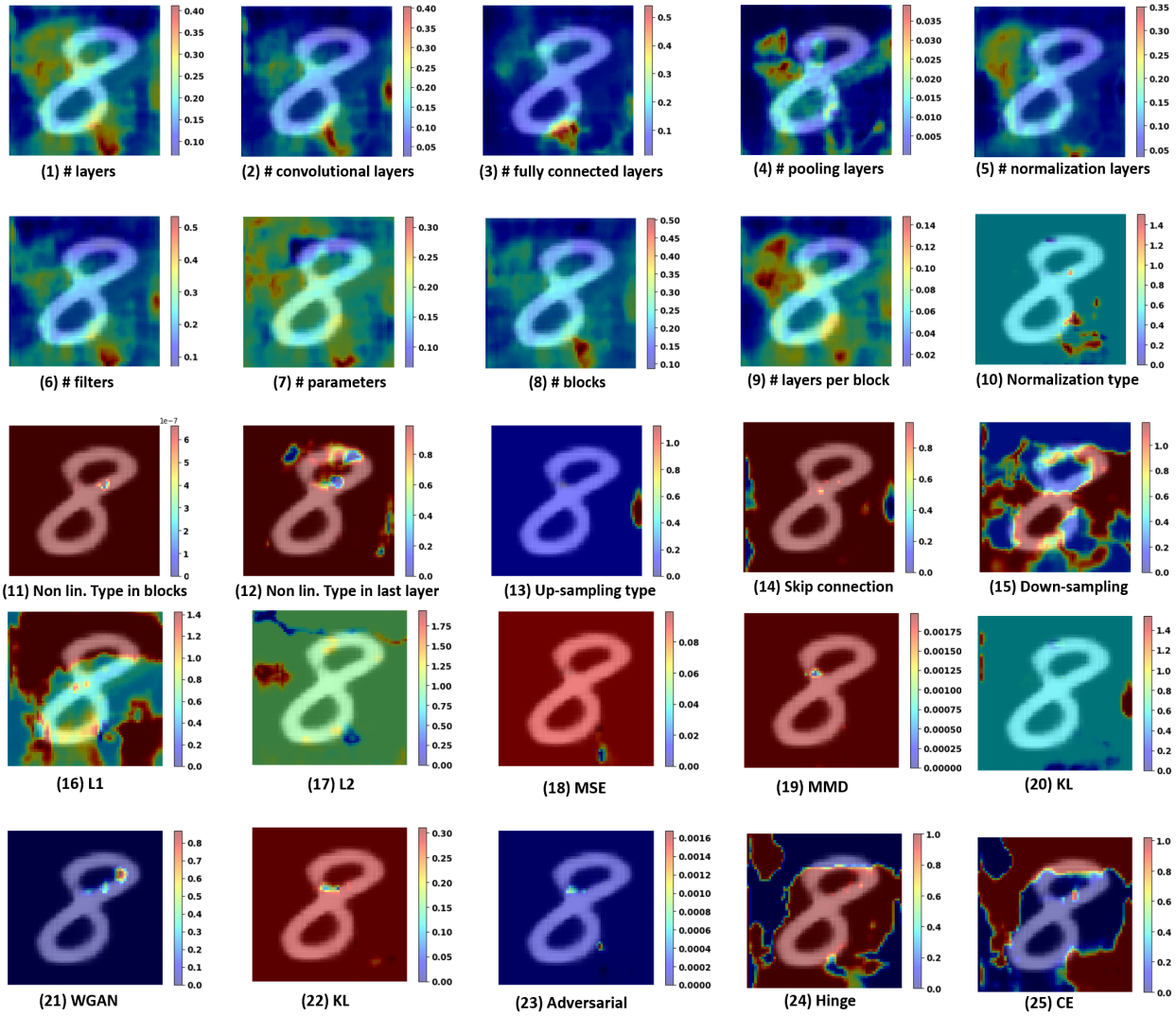


Figure H.2 Feature heatmap for each feature in network architecture and loss function predicted feature vector for MNIST data.

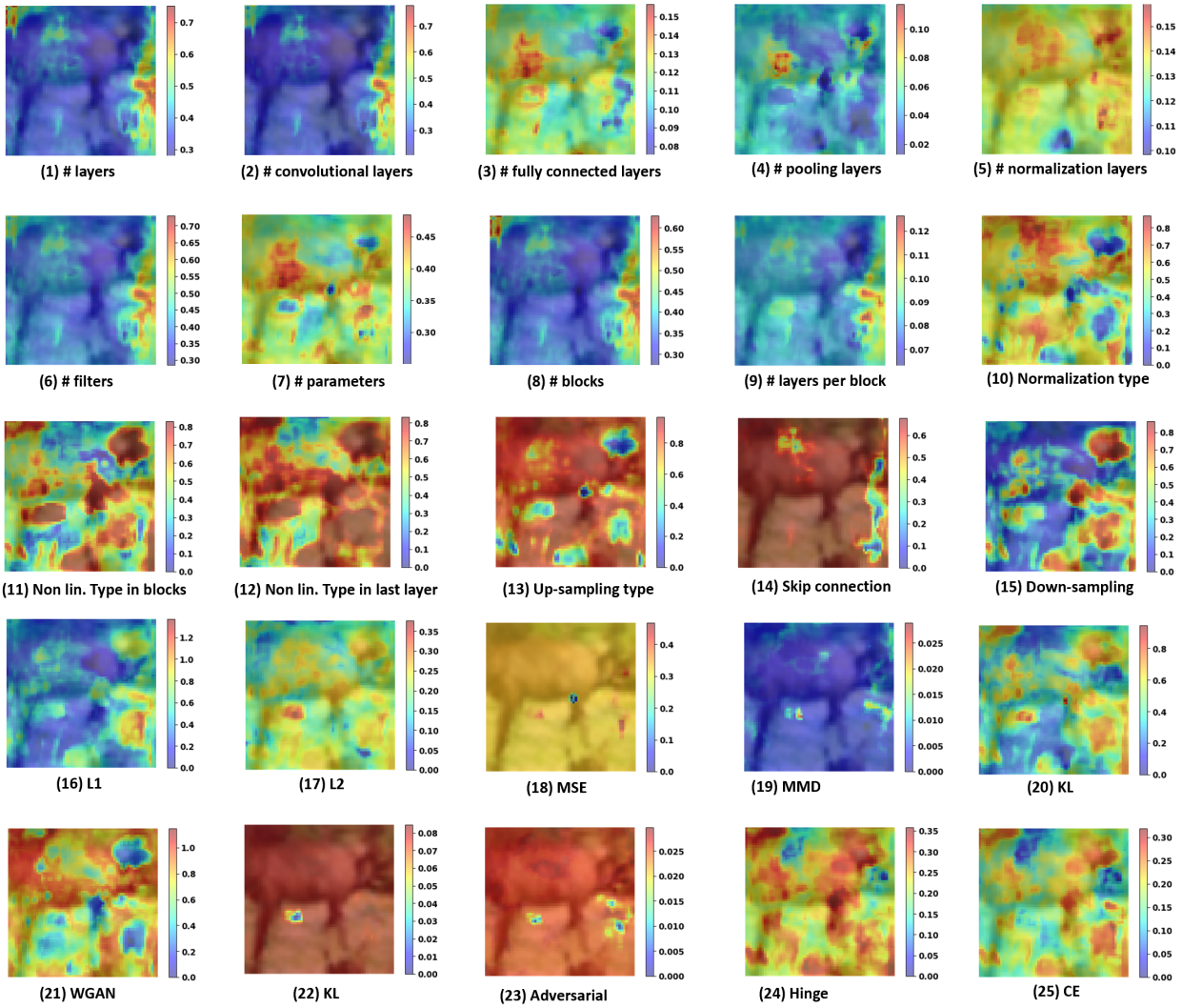


Figure H.3 Feature heatmap for each feature in network architecture and loss function predicted feature vector for CIFAR data.

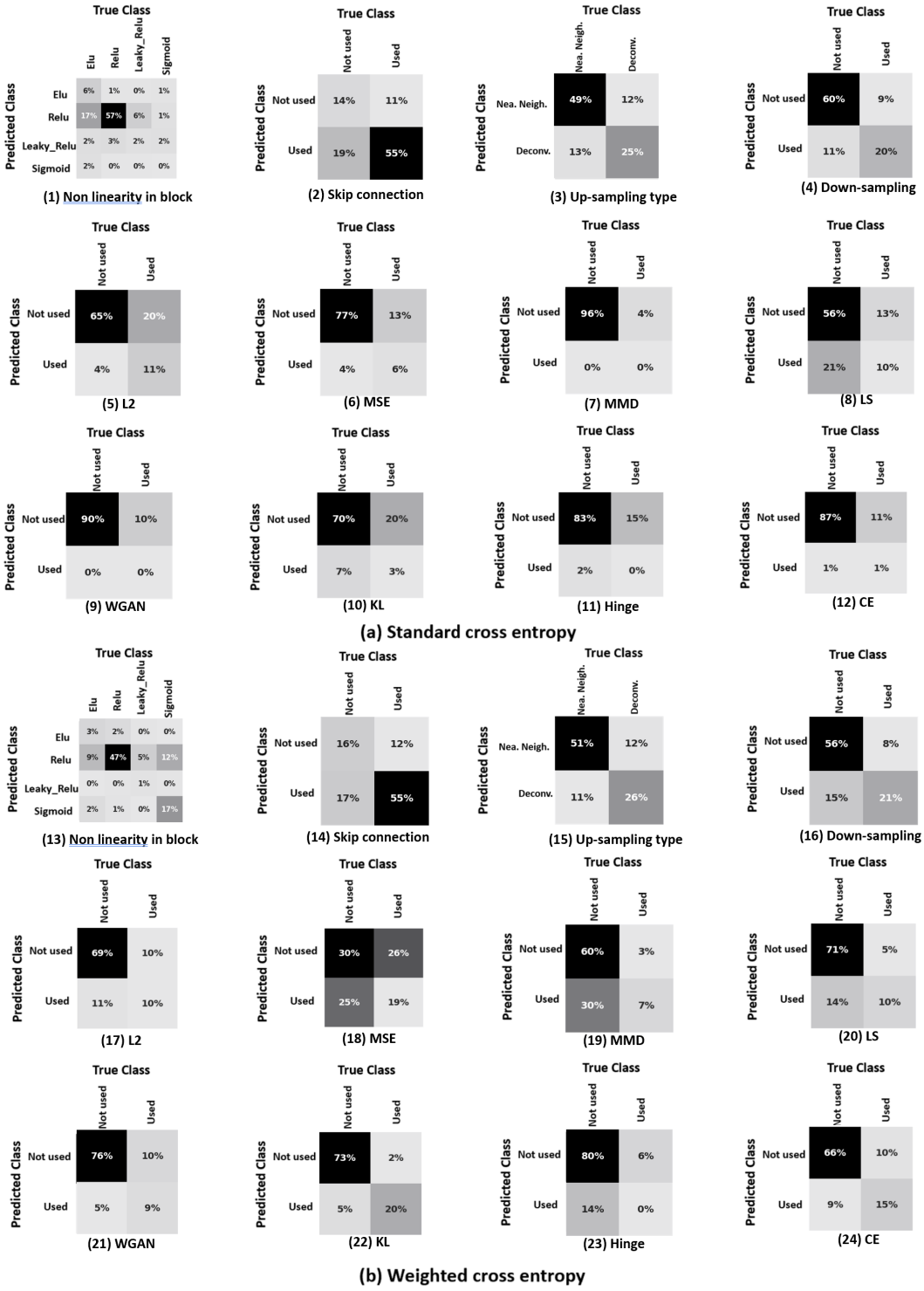


Figure H.4 Confusion matrix in the estimation of remaining parameters which were not shown in paper for network architecture and loss function. (1)-(12): Standard cross-entropy and (12)-(24): Weighted cross entropy. Weighted cross entropy handles imbalance of data much better than the standard cross entropy which usually predicts one class.

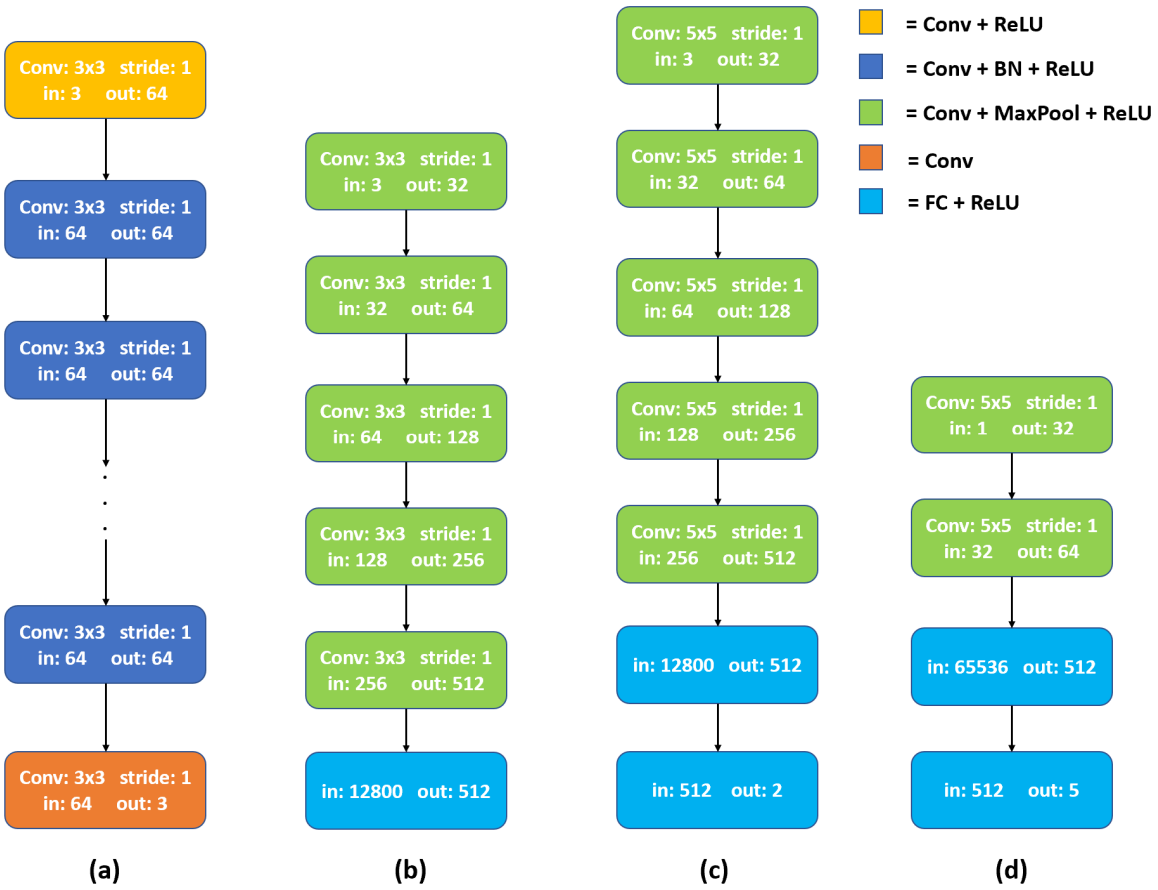


Figure H.5 Network architecture for various components of our method. (a) FEN (b) Mean and instance parser in PN (c) Shallow network for deepfake detection (d) Shallow network for image attribution.

H.4 Feature heatmaps

Every hyperparameter defined for network architecture and loss function type prediction may depend on certain region of the input image. To find out which region of the input image our model is looking at to predict each hyperparameter, we mask out 5×5 region from the input image. For the continuous type parameters, we compute the L_1 error between every predicted hyperparameter and its ground truth. This value of error will tell us how important is this 5 region in the input image to predict a particular hyperparameter. The higher the value of this error, the higher is the importance of that region in the prediction of the corresponding hyperparameter. For discete type parameters in network architecture and loss function, we estimate the probability of the ground truth label for every parameter. We subtract this probability from one to estimate the heatmap of the respective feature. Important regions will not affect the probability of the ground truth label

for a particular feature. To obtain a stable heatmap, we do the above experiment on 100 randomly chosen images across the different GMs and then calculate the average heatmap.

Fig. H.1, Fig. H.2 and Fig. H.3 show the feature heatmaps for every hyperparameter of network architecture and loss type feature vector for Face, MNIST and CIFAR data respectively. For each hyperparmater, there are certain regions of the input that are more important than others. Each type of data has different type of heatmaps indicating different regions of importance. For face and CIFAR, these regions lie mostly in the central part but for MNIST, many of the features depend on the regions closer to edges. There are also some similarities between these heatmaps for a particular type of data. This can indicate the similarity of these hyperparameters.