PREDICTIVE AND GENERATIVE MODELING OF TIME SERIES EXTREMES

By

Asadullah Hill Galib

A DISSERTATION

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

Computer Science—Doctor of Philosophy

2024

ABSTRACT

The accurate modeling of extreme values in time series data is a critical yet challenging task that has garnered significant interest in recent years. The impact of extreme events on human and natural systems underscores the need for effective and reliable modeling methods. To address this need, we develop novel methods that combine extreme value theory (EVT) with deep neural networks (DNNs) for a range of time series modeling tasks, including forecasting, representation learning, and generative modeling. While integrating EVT into DNNs can provide a robust framework for understanding the behavior of extreme values, several challenges arise in effectively integrating EVT with deep learning architectures. Successfully addressing these challenges necessitates a comprehensive strategy that leverages the strengths of both methodologies. Thus, this thesis posits that integrating extreme value theory within deep learning frameworks offers a sound approach to modeling extreme values in time series data by enhancing forecasting accuracy, advancing representation learning, and improving generative capabilities.

This thesis introduces four novel deep learning frameworks: DeepExtrema, Self-Recover, SimEXT, and FIDE, which offer promising solutions for forecasting, imputation, representation learning, and generative modeling of extreme values in time series data. DeepExtrema focuses on integrating extreme value theory with deep learning formulation to improve the accuracy and reliability of extreme events forecasting. Self-Recover addresses data fusion challenges that arise from varying temporal coverage associated with long-term and random missing values of predictors. SimEXT explores how deep learning can be utilized to learn useful time series representations that effectively capture tail distributions for modeling extreme events. FIDE introduces a high-frequency inflation-based conditional diffusion model tailored towards preserving extreme value distributions within generative modeling. These frameworks are evaluated using real-world and synthetic datasets, demonstrating superior performance over existing state-of-the-art methods. The contributions of this research are significant in advancing the field of time series modeling and have practical implications across various domains, such as climate science, finance, and engineering.

This thesis is dedicated to my beloved parents **Abdul Mannan** and **Khaleda Akter** and my loving wife **Samima Aktar Shoshy**.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my Ph.D. advisor, Dr. Pang-Ning Tan, for his invaluable guidance, support, and encouragement throughout my research journey. His passion for the field of Computer Science has been a constant source of inspiration for me.

Besides, I would like to extend my heartfelt thanks to my Ph.D. guidance committee members (Dr. Jiayu Zhou, Dr. Sijia Liu, and Dr. Lifeng Luo) for their invaluable feedback and constructive criticism throughout my research. Their expertise and insights have greatly enriched my work and helped me refine my ideas.

I am also grateful to my lab mates at DMiner@MSU for their support and stimulating discussions. Their diverse perspectives, expertise, and willingness to help have been instrumental in shaping my ideas and approach to research.

In addition, this thesis is partially supported by the National Science Foundation under grant #IIS-2006633. I would also like to thank Dr. Pang-Ning Tan for providing assistantships during my Ph.D.

I am deeply grateful to my parents, Abdul Mannan and Khaleda Aktar, my brothers, Khaled Mosharraf Mukut and Akib Al Muyeed, for their unconditional love, support, and encouragement throughout my academic pursuits. Their guidance and prayers have been the driving force behind my success. I owe a special thanks to my wife, Samima Aktar, whose unwavering belief, patience, and understanding have been an unwavering source of strength during this challenging journey.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 RELATED WORK 11

CHAPTER 3 DEEPEXTREMA: A DEEP LEARNING APPROACH FOR FORECASTING BLOCK MAXIMA IN TIME SERIES DATA 19

CHAPTER 4 SELF-RECOVER: FORECASTING BLOCK MAXIMA IN TIME SERIES FROM PREDICTORS WITH DISPARATE TEMPORAL COVERAGE USING SELF-SUPERVISED LEARNING 36

CHAPTER 5 SIMEXT: SELF-SUPERVISED REPRESENTATION LEARNING FOR EXTREME VALUES IN TIME SERIES 53

CHAPTER 6 FIDE: FREQUENCY-INFLATED CONDITIONAL DIFFUSION MODEL FOR EXTREME-AWARE TIME SERIES GENERATION 74

CHAPTER 7 CONCLUSIONS AND FUTURE WORK 98

BIBLIOGRAPHY 103

CHAPTER 1

INTRODUCTION

Time series is a sequence of observations recorded over time, where each observation depends on preceding ones, creating complex temporal interdependencies. Time series data is pervasive and comes in various forms, from stock prices and traffic patterns to climate data, disease outbreaks, and energy usage trends. Because time series data is so prevalent, capturing its dynamics has critical implications across numerous fields. For instance, in finance, time series analysis aids in stock market predictions; in energy, it informs load forecasting; in climate science, it models temperature and precipitation patterns. The broad applicability of time series data underscores its crucial role in scientific, economic, and social decision-making.

With the growing availability and significance of time series data in decision-making, accurate modeling underlying patterns, trends, and dynamics of time series is essential for strategic insights across diverse applications. Time series modeling includes several distinct tasks, such as forecasting, classification, generation, and anomaly detection. Numerous techniques have been developed to address these tasks, ranging from classical statistical approaches like GARCH (Generalized Autoregressive Conditional Heteroskedasticity) and ARIMA (Autoregressive Integrated Moving Average) to advanced machine learning techniques, broadly categorized into predictive and generative modeling paradigms. Predictive modeling focuses on learning mappings from input sequences to target variables, while generative modeling aims to capture the underlying data distribution for sequence generation. Recent advancements in deep learning have led to the development of sophisticated predictive modeling techniques for time series data, many of which are adapted from natural language processing. These techniques include Recurrent Neural Networks (RNNs) [1], Long Short-Term Memory networks (LSTMs) [2, 3, 4], and attention-based models (Informer [5], Autoformer [6]). Parallel developments in generative modeling have led to specialized time series adaptations including TimeGAN [7], TimeVAE [8], and Diffusion-TS [9] expanding the toolkit for time series modeling.

A key challenge in time series modeling is capturing extreme values, which are crucial due to

their significant impact on human and environmental systems. There are two primary approaches to defining extremes in time series: the **peak-over-threshold** approach, which considers values exceeding a specified threshold, and the **block maxima** approach, which defines extremes as the maximum within a given time window. Block maxima analysis offers insights into the worst-case scenario within time series, enabling a detailed understanding of maximum potential impacts. For instance, analyzing block maxima in hurricane data reveals the peak wind speeds of the hurricanes. Despite its importance, modeling block maxima poses significant challenges. Current time series modeling approaches primarily aim to minimize mean-square prediction error, focusing on predicting the conditional expectation rather than extreme values, thereby overlooking the critical tail distribution that characterizes extreme events.

To address this limitation, Extreme Value Theory (EVT) can be useful as it provides a robust statistical framework for modeling the distribution of extreme values. However, classical implementations of EVT struggle to capture the highly complex, nonlinear relationships inherent in time series data. This thesis hypothesizes that integrating EVT with advanced deep learning techniques can enhance the modeling of block maxima in time series by effectively capturing tail distributions while leveraging the representational power of neural networks. The primary challenge lies in developing an effective integration of EVT principles into deep learning architectures, a gap this research aims to address with significant implications for risk management and decision-making under uncertainty across multiple domains.

1.1 Research Challenges

Accurately forecasting extremes in time series is a complex and challenging task that demands specialized models and approaches, as advanced models like Transformer-based models struggle to predict extreme values with high accuracy. The difficulty of forecasting extremes is emphasized by Figure 1.1, which illustrates the limitations of existing models and techniques in predicting extreme events. The figure shows the results of three different settings for time series forecasting of a temperature dataset[1] using a Transformer-based model. The top row shows the performance

[1] https://www.narccap.ucar.edu/data/index.html

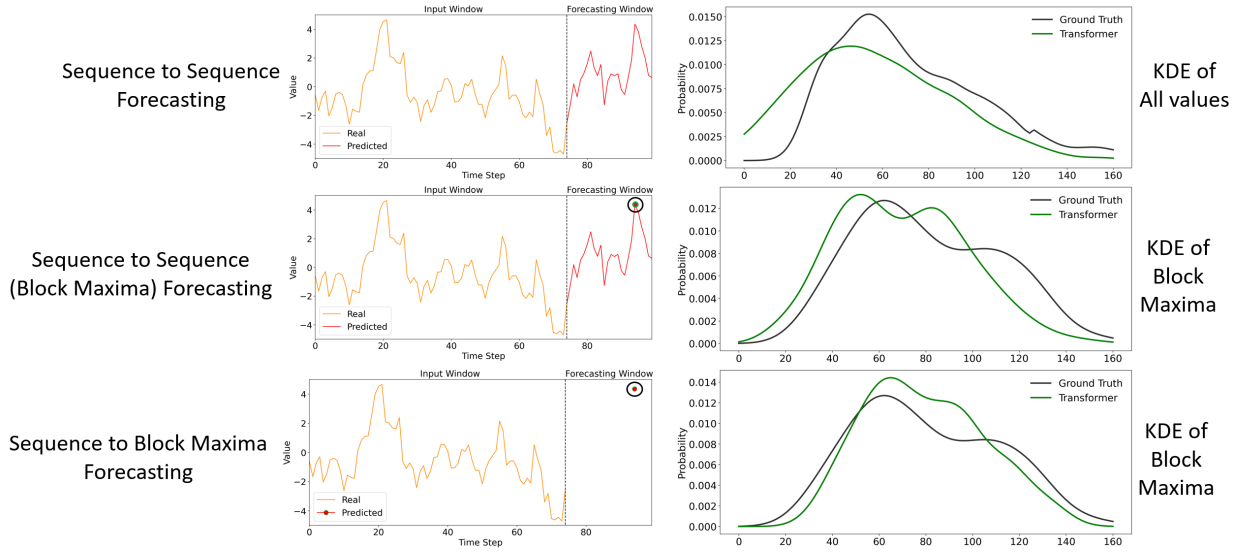


Figure 1.1 The figure illustrates the challenge of forecasting extremes, as even advanced models like Transformer struggle with accuracy. Directly predicting block maxima improves performance but still falls short of the ground truth distribution, as shown by Kernel Density Estimation (KDE). The left column shows a sample sequence, and the right column displays the KDE of all sequences in a temperature dataset.

of sequence-to-sequence forecasting, which does not perform well considering all values in the forecasting window, particularly for extreme values, as shown by the Kernel Density Estimation plots. The middle row depicts the results of sequence-to-sequence forecasting, where the model predicts all time steps in the forecasting window but only uses block maxima for optimization and evaluation. It also fails to perform well for extreme values. The bottom row shows the model's performance when directly predicting block maxima in the forecasting window, which is an improvement over the previous settings, but it still falls short of the ground truth distribution. This indicates that forecasting extreme events requires more sophisticated and specialized models, as accurately capturing these extremes remains a complex and challenging task, as shown in Figure 1.1.

The pursuit of accurate forecasting of extreme values in time series data presents several key research challenges. **First, the rarity of extreme values** poses a significant challenge in modeling, as these extreme values represent a small fraction of the dataset, often leading to an insufficient representation of extreme events during training. In some cases, extreme values may have been ignored as outliers during training to improve the generalization performance of the model. This scarcity can hinder a model's ability to generalize and accurately predict future extremes.

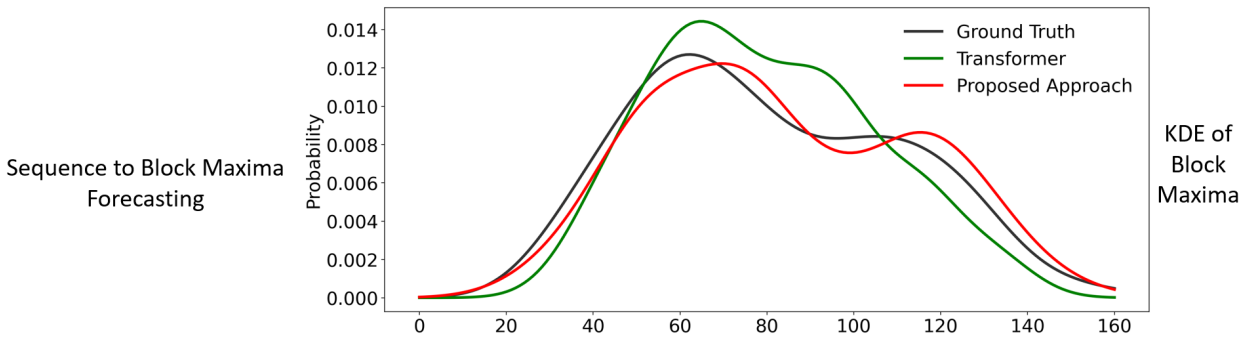


Figure 1.2 The figure highlights that incorporation of GEV into deep learning through our proposed approach performs well compared to ground truth distribution of extreme values.

Second, regularizing deep learning models is crucial, especially with limited data. Deep learning models typically require large amounts of data to achieve optimal performance, and a scarcity of data can result in overfitting, where the model excels on training data but fails to generalize to new, unseen instances. In cases of limited data, regularization techniques become essential to mitigate overfitting risks. In this context, Extreme Value Theory (EVT) could offer a solution, as EVT provides a statistically sound approach to characterizing the limiting distribution of extreme values. Integrating EVT into deep learning models not only aids in regularization but also enhances the model's ability to capture extreme behaviors by establishing constraints based on the underlying statistical properties of extremes.

Third, modeling extreme values in time series data requires a specialized focus on the tail distribution, as opposed to conventional existing deep learning methods, which prioritize modeling the conditional mean of the target variable by minimizing the mean-square prediction error [10]. The rarity of extreme values also necessitates a distinctive modeling approach that emphasizes the tail distribution rather than the overall distribution of the dataset. A robust representation learning approach tailored to the tail distribution or directly modeling the underlying distribution of extreme values is essential for accurately capturing and modeling these phenomena. This approach must effectively address the unique characteristics of the tail distribution, ensuring that the model can capture the critical information inherent in the distribution of extreme values and enhance its predictive performance for future extremes.

Fourth, effective data fusion is essential for enhancing the modeling of extreme events.

To enhance the forecasting of extreme values, relying solely on historical observations is often inadequate. This is particularly true when dealing with non-stationary time series, where historical data may not provide a robust foundation for accurately predicting block maxima. In such cases, incorporating domain-driven process-based models can effectively supplement historical data and improve forecast accuracy. For example, when forecasting climate extremes, integrating predictors from global climate models (GCMs) can provide valuable insights into potential future extremes that historical data alone might overlook. However, the integration of these diverse data sources introduces significant technical and conceptual challenges. Key issues include handling missing or inconsistent data, addressing varying temporal coverage, and accounting for different sources of uncertainty. Specifically, for time series extremes, merging outputs from process-based models with historical observations can be complicated by disparities in temporal coverage among different data sources. Addressing these challenges is critical to ensure the reliability and accuracy of extreme event predictions, thereby enhancing the effectiveness of decision-making and risk management strategies.

Fifth, enhancing generative modeling techniques for extreme values is critical for effective decision-making across various fields. The generative modeling of extreme values in time series plays a critical role in informed decision-making across various domains, such as climate science, energy consumption, and disease outbreak detection. Effective generative modeling of these extremes enables the understanding of the underlying data distribution, facilitates data augmentation, and enhances uncertainty estimations. These capabilities are crucial for developing robust risk management strategies and improving disaster preparedness measures. Despite the increasing interest in applying generative models to time series analysis [11, 12], their efficacy in accurately preserving the distribution of extreme values has been insufficiently explored. Notably, state-of-the-art generative models, including diffusion models [13, 9], demonstrate a strong ability to generate samples that conform to the overall data distribution. However, they often fail to effectively capture the distribution of block maxima values. This limitation highlights the urgent need for further research aimed at improving generative models' performance in preserving the characteristics

of extreme values within time series data.

In short, the key research challenges include the rarity of extreme values, the need for specialized regularization, capturing tail distributions, effective data fusion, and enabling generative models to preserve extreme value distribution. Addressing these challenges is essential for advancing the modeling of extreme values in time series.

1.2 Thesis Statement

Integrating extreme value theory within deep learning frameworks offers a robust approach to modeling extreme values in time series data by enhancing forecasting accuracy, advancing representation learning, and improving generative capabilities.

1.3 Thesis Contributions

To validate the thesis statement, The primary objective of this thesis is to develop deep learning frameworks that tackle the key research challenges discussed in the previous section. Chapter 3 addresses the challenge of rarity in extreme values and limited training data by incorporating extreme value theory (EVT) as a form of regularization into deep learning models, improving their ability to forecast extreme events using historical data. Chapter 4 extends this approach to multivariate model-based forecasts, with techniques for handling long-term and randomly occurring missing values, addressing both data fusion and generalization issues. Chapter 5 explores the challenge of accurately modeling tail distributions by developing deep learning methods that focus on learning effective representations for extremes in time series. Chapter 6 focuses on enhancing generative models to accurately preserve the distribution of extreme values by combining frequency-domain manipulation with EVT. A high-level overview of these contributions is presented in the following subsections.

1.3.1 `DeepExtrema`: A Deep Learning Approach for Forecasting Block Maxima in Time Series Data

Forecasting block maxima in time series data is essential for assessing risk and impact in various domains but remains challenging due to the rarity of extreme values and limited training data, as highlighted in the first two research challenges. To address these, we introduce `DeepExtrema`,

a novel framework that integrates deep neural networks (DNNs) with EVT for effective block maxima forecasting. This approach leverages EVT, more specifically generalized extreme value (GEV), to capture the distribution of extreme values and regularize the training, while the DNN captures complex dependencies in the data. Integrating EVT into deep learning involves certain technical challenges, such as maintaining GEV constraints during training and preventing violations of regularity constraints due to random network initialization. `DeepExtrema` overcomes these issues with innovations like model bias offset, constraint reformulation, and reparameterization, effectively addressing the first two research challenges. Extensive experiments on synthetic and real-world datasets show that `DeepExtrema` significantly outperforms baseline methods, improving the accuracy of extreme event forecasting by 6.5%-16% in time series data.

1.3.2 `Self-Recover`: Forecasting Block Maxima in Time Series from Predictors with Disparate Temporal Coverage using Self-Supervised Learning

Relying solely on historical observations is often insufficient for accurately predicting block maxima, particularly in non-stationary time series where historical data may not capture the complexities of future extremes. To address this, integrating domain-driven process-based models can significantly improve forecast accuracy by providing additional context and insights. However, this integration introduces challenges, such as handling missing data, reconciling different temporal coverage, and managing uncertainties from diverse sources. To tackle these issues, we propose `Self-Recover`, a self-supervised learning framework designed to predict block maxima by effectively fusing disparate temporal data sources. This approach employs a combination of contrastive learning and generative modeling techniques to impute long-term missing values and a denoising autoencoder for handling data that are missing at random. By seamlessly combining representations from historical observations and process model outputs, `Self-Recover` ensures comprehensive modeling of the available data while also incorporating the GEV distribution for consistency in predictions. Extensive experiments on real-world datasets demonstrate the superiority of `Self-Recover` in forecasting block maxima, showcasing its ability to enhance predictive performance through effective data fusion strategies.

1.3.3 SimEXT: Enhancing Time Series Forecasting of Extreme Values via Self-supervised Representation Learning

This approach addresses the critical need for robust representation learning tailored specifically to extreme values. Modeling extreme values in time series data requires a specialized focus on the tail distribution rather than the overall distribution, as highlighted in the third research challenge. To effectively address this challenge, we introduce SimEXT, a self-supervised learning framework designed to learn robust representations that preserve the fidelity of the tail distribution. SimEXT utilizes a combination of contrastive learning and a reconstruction-based autoencoder architecture, facilitating the capture of temporal patterns linked to extreme events. To further enhance the learning process, the framework incorporates a wavelet-based data augmentation technique alongside a distribution-based loss function that emphasizes the extreme value distribution. We establish probabilistic guarantees for the wavelet-based augmentation, ensuring that perturbations to the wavelet coefficients do not significantly alter the extreme values within the time series. Experimental results on real-world datasets demonstrate that SimEXT effectively captures the unique properties of the tail distribution, significantly improving the performance of downstream tasks focused on forecasting block maxima.

1.3.4 FIDE: Frequency-Inflated Conditional Diffusion Model for Extreme-Aware Time Series Generation

Time series generation is a crucial aspect of data analysis, playing a pivotal role in learning temporal patterns and their underlying dynamics across diverse domains. However, conventional time series generation methods often struggle to adequately capture extreme values, diminishing their value in critical applications such as scenario planning and risk management in healthcare, finance, climate change adaptation, and beyond. To tackle this challenge, we introduce a conditional diffusion model called FIDE, specifically designed to preserve the distribution of extreme values in generative modeling for time series. FIDE employs a novel high-frequency inflation strategy in the frequency domain to prevent the premature fade-out of extreme values. Additionally, it extends traditional diffusion-based models to enable the generation of samples conditioned on the block maxima, thereby enhancing the model's capacity to capture extreme events. The FIDE framework

also incorporates the Generalized Extreme Value (GEV) distribution within its generative modeling framework, ensuring fidelity to both block maxima and overall data distribution. Experimental results on real-world and synthetic datasets demonstrate the efficacy of `FIDE` over baseline methods, highlighting its potential to advance generative AI for time series analysis, particularly in accurately modeling extreme events. This approach addresses the critical need for enhanced generative modeling techniques for extreme values, which is essential for effective decision-making across various fields.

1.4 Publications

The following works were used as a source for some of the content in this thesis:

- **Asadullah Hill Galib**, Andrew McDonald, Tyler Wilson, Lifeng Luo, & Pang-Ning Tan (2022, Jul.). DeepExtrema: A Deep Learning Approach for Forecasting Block Maxima in Time Series Data. In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, **IJCAI 2022** (pp. 2980-2986). (Chapter 3)
- **Asadullah Hill Galib**,Andrew McDonald , Pang-Ning Tan & Luo, L. (2023, Jan.). Self-Recover: Forecasting Block Maxima in Time Series from Predictors with Disparate Temporal Coverage using Self-Supervised Learning. In Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, **IJCAI 2023** (pp. 3723-3731). (Chapter 4)
- **Asadullah Hill Galib**, Pang-Ning Tan & Lifeng Luo (2023, Feb.). SimEXT: Self-supervised Representation Learning for Extreme Values in Time Series. In Proceedings of the 23rd IEEE International Conference on Data Mining, **ICDM 2023**, (pp. 1031-1036), IEEE. (Chapter 5)
- **Asadullah Hill Galib**, Pang-Ning Tan & Lifeng Luo (2023, Feb.). FIDE: Frequency-Inflated Conditional Diffusion Model for Extreme-Aware Time Series Generation. Advances in Neural Information Processing Systems **NeurIPS 2024**, 36. [To Appear] (Chapter 6)

In addition, here are the publications related to this research that I have published:

- Yue Deng, **Asadullah Hill Galib**, Pang-Ning Tan & Lifeng Luo (2024, Aug.). Unraveling Block Maxima Forecasting Models with Counterfactual Explanation. In Proceedings of the

30th ACM SIGKDD 2022 Conference on Knowledge Discovery and Data Mining, **KDD 2024** (pp. 562-573).

- Tyler Wilson, **Asadullah Hill Galib**, Pang-Ning Tan, and Lifeng Luo. Beyond Point Prediction: Capturing Zero-Inflated & Heavy-Tailed Spatiotemporal Data with Deep Extreme Mixture Models. In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, **KDD 2022**, Washington, DC (2022).
- Tyler Wilson, Pang-Ning Tan, Lifeng Luo, and **Asadullah Hill Galib**. Deep Learning With Extreme Value Theory for Modeling Precipitation Events. In AGU Fall Meeting Abstracts, vol. 2021, pp. A15Q-07. 2021.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 provides a review of relevant literature and previous research related to the topic. Chapter 3 presents the `DeepExtrema` framework, which combines a deep neural network with a generalized extreme value distribution to predict the block maximum value of a time series. In Chapter 4, a new deep learning framework named `Self-Recover` is introduced, which overcomes the temporal data availability problem by using self-supervised learning to forecast block maxima. Chapter 5 presents `SimEXT`, a self-supervised learning framework that learns a robust representation of a time series to accurately capture its tail distribution. Chapter 6 presents `FIDE`, a frequency-inflated conditional diffusion model to preserve block maxima distribution in time series generation. Finally, Chapter 7 concludes this thesis by outlining potential future research.

CHAPTER 2

RELATED WORK

In this section, we describe the current state of research in the application of deep learning to time series data. Additionally, we will explore the advancements in extreme value theory as it pertains to time series analysis. Also, we will discuss the emerging trend of self-supervised learning, which has been gaining traction as an effective approach to analyzing time series data. Finally, we will discuss anomaly detection in time series and how it differs from forecasting extreme values.

2.1 Deep Learning for Time Series Forecasting

Deep Learning (DL) has emerged as a promising technique for time series forecasting due to its ability to handle complex patterns and nonlinear relationships in the data. In this section, we summarize recent advances and developments in DL-based methods for time series forecasting.

One of the most common DL-based methods for time series forecasting is Recurrent Neural Networks (RNNs), which have been shown to be effective in capturing temporal dependencies in sequential data. In particular, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) have been widely adopted in various applications. LSTM networks [14] are able to learn long-range dependencies in data, which is essential for forecasting time series that exhibit trends and seasonality. Also, it overcomes the vanishing and exploding gradient problem in other recurrent neural networks. Since its inception, LSTM and its variants have been successfully applied to forecasting time series in various application domains. For example, [3] used an LSTM-based model to predict time series electric load prediction [3]. [2] proposed a variation of LSTM to predict petroleum production. Another popular deep learning architecture for time series forecasting is the gated recurrent unit (GRU) network. GRU networks are similar to LSTM networks, but they are simpler and faster to train. [1] employed an encoder-decoder-based GRU to predict host workloads in a cloud computing environment.

Another direction of DL-based methods for time series forecasting is using 1D Convolutional Neural Networks (CNNs), which have been shown to be effective in capturing local patterns in the data. Some of the earliest successful applications of neural networks have involved convolutional

neural networks. In applications to time series, a small filter is passed over the temporally localized region to model temporal relationships. For example, [15] proposed a deep CNN-based model for forecasting time series data, achieving competitive performance compared to traditional methods. In addition, [16] proposed a deep spatio-temporal CNN for traffic flow prediction, outperforming various baseline methods. [17] proposed a CNN architecture for activity recognition while [18] employed a CNN-based architecture for a variety of time series classification problems. [19] empirically demonstrated the effectiveness of CNNs as alternatives to LSTMs in sequence-based modeling.

Moreover, DL-based methods for time series forecasting have also been extended to include attention mechanisms [20], which can selectively focus on important features and temporal patterns in the data. Attention-based RNN seemed to be a promising direction in time series analysis for its consideration of the context apart from the typical data-driven approach. For example, [21] proposed a deep learning model combining CNN, LSTM, and attention mechanism for stock price prediction, achieving superior performance compared to traditional methods. [22] proposed a transformer-based architecture for time series forecasting. [23] used attention-based RNN for multi-step time series prediction of process performance. [24] proposed an attention-based LSTM framework for financial time series forecasting. [5] focuses on transformer challenges and introduces an effective transformer-based model, called Informer, that significantly enhances the inference speed of long time-series forecasting (LSTF) by predicting entire sequences in one forward operation instead of step-by-step.

There are a number of other deep learning architectures that have been used for time series forecasting. These include recurrent convolutional neural networks (RCNNs) [25, 26] temporal convolutional networks (TCNs) [27, 28], reservoir computing [29, 30], graph neural networks [31], etc.

While deep learning models have demonstrated substantial predictive accuracy in time series forecasting, capturing uncertainty remains a significant challenge, especially for extreme values that diverge from typical patterns. Traditional forecasting models are generally optimized for point

predictions, limiting their capacity to account for the inherent uncertainty in time series data and often overlooking extreme event probabilities. To address this, recent research has focused on integrating uncertainty quantification techniques with deep learning models to better capture the range of possible outcomes. For instance, [32] proposed a distribution-free novel approach called DeepPIPE. It simultaneously predicted point estimations as well as quantile estimations without any prior assumption about the data distribution. They proposed a hybrid loss function based on point estimations and point intervals to leverage point and quantile estimations. Although DeepPIPE provides a robust baseline for uncertainty estimation, it does not incorporate extreme value theory (EVT), which is crucial for accurately predicting rare events in the tails of the distribution. [4] proposed an LSTM-based architecture for time series extreme event forecasting where they combined Bootstrap and Bayesian approaches for uncertainty estimation.

2.2 Deep Learning for Time Series Extremes

[33] is representative of much of the traditional statistical work utilizing Extreme Value Theory (EVT). They analyzed GEV parameters assuming there was a simple relationship between those parameters and a single predictor, i.e., time. However, they did not provide a method for making point estimates and their model was only able to model very simple relationships. Most of the prior works combining deep learning with EVT suffer from significant limitations. For instance, [34] incorporated the GP distribution in their loss function to forecast excesses over a threshold. However, instead of predicting the GP parameters from covariates, they assume the GP parameter values are known *a-priori* and can be provided as hyperparameters of their algorithm. Thus, the GP parameters are assumed to be fixed (constant) for the all-time series, which is a strong assumption especially if the time series is generated for different locations. [35] proposed to combine a deep learning model with extreme value theory to model the tail behavior of a time series. They applied the GP distribution for modeling excess values and used its negative log-likelihood as the loss function. However, a major issue with their proposed framework is that it does not incorporate a mechanism to enforce constraints on parameters of the learned GP distribution, which is essential to ensure the predicted distribution is well-behaved. [36] combine copulas and normalizing flows

with the GP distribution to model multivariate extremes. [37, 38] combine deep learning with the GP distribution for spatiotemporal data using the maximum log-likelihood of approach and a reparameterization technique. Similar to other works, their approach is not designed for block maxima prediction nor does it handle the varying temporal data availability issue.

2.3 Self-supervised Representation Learning for Time Series

Self-supervised learning has become increasingly popular in recent years due to its excellent performance on various benchmark datasets along with strong theoretical foundations [39, 40, 41]. Self-supervised learning has been shown to outperform supervised learning in various instances [39, 42]. There are a number of studies focusing on the theoretical foundations of self-supervised learning. For example, [43] provided provable guarantees on the performance of the learned representations through self-supervised learning. [44] showed that self-supervised learned representations can extract task-relevant information and discard task-irrelevant information. Several studies theoretically explored these insights and demonstrated provable guarantees for contrastive [41, 45], non-contrastive [46], and reconstruction-based [47] self-supervised learning.

In the context of time series, SSL has been shown to be an effective method for representation learning and feature extraction, which can benefit various downstream tasks such as forecasting, classification, and anomaly detection. One of the most popular SSL-based methods for time series is Autoencoding, which aims to learn a compressed representation of the input data by reconstructing it from a compressed latent space. For instance, [48] proposed a self-supervised framework for time series forecasting, which uses recurrent autoencoder ensembles to learn a compressed representation of the input data and generate future predictions. The proposed method achieved competitive performance compared to traditional methods.

Another direction of SSL-based methods for time series is using contrastive learning, which aims to learn a representation that maximizes the similarity between positive examples and minimizes the similarity between negative examples. For example, [49] proposed a Contrastive Predictive Coding (CPC) framework that employs a contrastive objective to learn a representation that captures temporal dependencies in audio signals. Similarly, [50] proposed a Contrastive Multiview Coding

(CMC) framework for time series that uses a contrastive objective to learn a representation capturing different perspectives of the input data. The proposed method has shown exceptional performance in various downstream tasks such as forecasting and classification. Researchers have extended contrastive learning in several ways. For instance, [51] use contrastive learning hierarchically over augmented context views for representation learning. [52] apply intra-level contrastive learning to disentangle seasonal and trend representations from time series. [53] incorporate frequency domain information while maintaining consistency between time and frequency domains. [54] combine temporal and contextual contrasting for representation learning. Additionally, [55] propose novel data augmentation techniques that not only generate phase shifts and amplitude changes but also retain the structure and feature information of the time series along with contrastive learning. It is evident from these studies that Contrastive Learning is a powerful tool for representation learning in time series analysis.

Moreover, SSL-based methods for time series have also been extended to include Generative Adversarial Networks (GANs), which can learn a generator model that can generate realistic samples from the input data distribution. For example, [56] proposed a self-supervised framework for time series forecasting, which uses GANs to generate future predictions from the input data. Similarly, [57] proposed a self-supervised framework for time series anomaly detection, which uses GANs to generate normal samples from the input data distribution and detect anomalies based on the reconstruction error.

SSL-based methods for time series have shown remarkable progress in recent years, with a variety of architectures and techniques being proposed and achieving competitive performance. Future research directions may include exploring more complex models and architectures, developing more efficient training algorithms, and investigating the interpretability and explainability of SSL-based methods.

2.4 Generative Modeling for Time Series

Time series generation has been an active area of research, with various statistical and machine learning techniques employed to capture temporal dependencies and complexities in data. Classical

statistical models, such as AR, ARMA, MA, and ARIMA [58], have been utilized for time series generation. However, these models often struggle to capture non-linear and complex relationships, limiting their applicability to diverse problems.

To address these limitations, advanced generative models have emerged, including Generative Adversarial Networks (GANs) [59], Variational Autoencoders (VAEs) [60], normalizing flows [61], and diffusion-based approaches [13, 62]. These methods have demonstrated efficacy in time series generation due to their ability to learn underlying data distributions for data generation. While normalizing flows are constrained by computational complexity, limited expressiveness, and suboptimal sample quality, GANs and VAEs offer advantages in this regard. Numerous GAN architectures, such as RcGAN [63] and TimeGAN [7], have been proposed for generating realistic time series data. TimeGAN [7] employs an encoder-decoder architecture to transform time series samples into latent vectors. However, GAN-based models are susceptible to issues like mode collapse and unstable training behavior. VAEs, although not extensively applied to synthetic time series generation, have shown promise in related tasks like time series imputation [64], suggesting their potential utility in this domain.

Diffusion-based models are also gaining traction for their ability to generate high-quality data, bypassing challenges associated with discriminator networks in GANs and avoiding artifact-prone lower-dimensional latent spaces of VAEs. While a few diffusion-based works [11, 12] have been employed for time series, they are specifically designed for discriminative tasks. Despite the advantages of generative AI for time series, modeling extreme values using these models remains largely unexplored. Previous research [65] has recognized the difficulty of capturing extremes using generative models like normalizing flows. Studies [66, 67] have highlighted the inability of normalizing flows to accurately capture heavy-tailed marginal distributions, as mapping heavy-tailed distributions to light-tailed distributions cannot maintain Lipschitz-boundedness. However, this challenge remains largely unaddressed within the realm of diffusion models.

2.5 Anomaly Detection in Time Series

While it may seem that anomaly detection and extreme value forecasting are similar, it is crucial to recognize that they have distinct goals and necessitate different methods. Although the terms may occasionally be used interchangeably or confused with one another, it is essential to understand their differences. Anomaly detection primarily involves identifying abnormal patterns or data points in a time series, without necessarily predicting the specific magnitude or value of the anomaly. In contrast, extreme value forecasting aims to forecast the magnitude or value of a specific event, such as a maximum or minimum value, in the future.

Anomaly detection in time series is a crucial task in various applications, including finance, healthcare, and industrial process control. The ability to detect deviations from the expected behavior in real time can prevent critical failures, avoid financial losses, and improve overall efficiency. One of the most popular approaches for anomaly detection in time series is statistical methods, which rely on modeling the probability distribution of the data and identifying deviations from it. For instance, the use of Gaussian distribution-based methods, such as the Z-score and the Mahalanobis distance, has been extensively studied in the literature [68]. However, these methods are often limited by their assumption of normality and may not be effective in detecting anomalies in complex and high-dimensional data.

In recent years, machine learning techniques, especially deep learning-based methods, have shown remarkable progress in anomaly detection for time series data. One of the most popular deep learning-based methods for anomaly detection is the use of autoencoder networks, which can learn a compressed representation of the input data and reconstruct it from a latent space. Anomaly detection can then be performed by identifying samples with high reconstruction errors [69]. For example, [70] proposed a self-supervised framework for time series anomaly detection, which uses Variational Autoencoders (VAE) to learn a compressed representation of the input data and reconstruct it from the latent space. Another direction of deep learning-based methods for anomaly detection in time series is the use of recurrent neural networks (RNNs), which can capture temporal dependencies and patterns in sequential data. For instance, [71] proposed a framework for anomaly

detection in time series data based on LSTMs, which achieved state-of-the-art performance on benchmark datasets. Moreover, the use of attention mechanisms in RNN-based models has been shown to improve the accuracy of anomaly detection by focusing on important temporal features in the data [72]. Furthermore, recent studies have also explored the use of unsupervised learning methods for anomaly detection in time series data, which do not require labeled data for training. For example, [73] proposed an autonomous anomaly detection technique for multivariate time series data (TimeAutoAD) based on a novel self-supervised contrastive loss. Machine learning techniques, especially deep learning-based methods, have shown great potential in improving the accuracy and efficiency of anomaly detection in time series data.

CHAPTER 3

DEEPEXTREMA: A DEEP LEARNING APPROACH FOR FORECASTING BLOCK MAXIMA IN TIME SERIES DATA

3.1 Introduction

Extreme events such as droughts, floods, and severe storms occur when the values of the corresponding geophysical variables (such as temperature, precipitation, or wind speed) reach their highest or lowest point during a period or surpass a threshold value. Extreme events have far-reaching consequences for both humans and the environment. For example, four of the most expensive hurricane disasters in the United States since 2005—Katrina, Sandy, Harvey, and Irma—have each incurred over $50 billion in damages with enormous death tolls [74]. Accurate forecasting of the extreme events [75] is, therefore, crucial as it not only helps provide timely warnings to the public but also enables emergency managers and responders to better assess the risk of potential hazards caused by future extreme events.

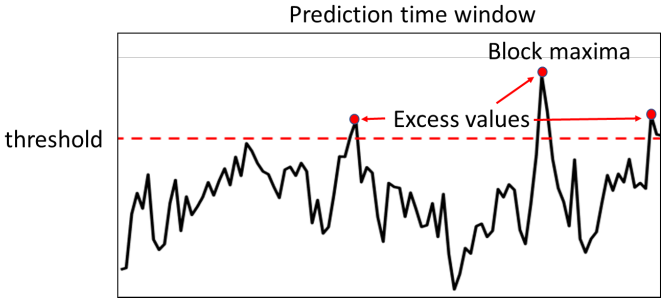


Figure 3.1 Types of extreme values in a given time window.

Despite its importance, forecasting time series with extremes can be tricky as the extreme values may have been ignored as outliers during training to improve the generalization performance of the model. Furthermore, as mentioned in chapter 1, current approaches are mostly designed to minimize the mean-square prediction error, their fitted models focus on predicting the conditional expectation of the target variable rather than its extreme values [76]. Extreme value theory (EVT) offers a statistically well-grounded approach to derive the limiting distribution governing a sequence of extreme values [10]. The two most popular distributions studied in EVT are the generalized extreme value (GEV) and generalized Pareto (GP) distributions. Given a prediction time window,

GEV governs the distribution of its block maxima, whereas GP is concerned with the distribution of excess values over a certain threshold, as shown in Figure 3.1. This work focuses on forecasting the block maxima as it allows us to assess the worst-case scenario in the given forecast time window and avoids making ad-hoc decisions regarding the choice of excess threshold to use for the GP distribution. As mentioned in chapter 1, classical EVT has limited capacity in terms of modeling highly complex, nonlinear relationships present in time series data. For example, [33] uses a simple linear model to infer the GEV parameters.

Deep learning methods have grown in popularity in recent years due to their ability to capture nonlinear dependencies in the data. Previous studies have utilized a variety of deep neural network architectures for time series modeling, including long short-term memory networks [2, 3, 4], convolutional neural networks [19, 18, 17], encoder-decoder based RNN [1], and attention-based models [24, 23]. However, these works are mostly focused on predicting the conditional mean of the target variable, as mentioned in chapter 1. While there have been some recent attempts to incorporate EVT into deep learning [38, 34, 35], they are primarily focused on modeling the tail distribution, i.e., excess values over a threshold, using the GP distribution, rather than forecasting the block maxima using the GEV distribution. Furthermore, instead of inferring the distribution parameters from data, some methods [34] assume that the parameters are fixed at all times and can be provided as user-specified hyperparameters while others [35] do not enforce the necessary constraints on parameters of the extreme value distribution.

Incorporating the GEV distribution into the deep learning formulation presents many technical challenges. First, the GEV parameters must satisfy certain positivity constraints to ensure that the predicted distribution has a finite bound [10]. Maintaining these constraints throughout the training process is a challenge since the model parameters depend on the observed predictor values in a mini-batch. Another challenge is the scarcity of data since there is only one block maximum value per time window. This makes it hard to accurately infer the GEV parameters for each window from a set of predictors. Finally, the training process is highly sensitive to model initialization. For example, the random initialization of a deep neural network (DNN) can easily violate certain

regularity conditions of the GEV parameters estimated using maximum likelihood (ML) estimation. An improper initialization may lead to ξ estimates that defy the regularity conditions. For example, if $\xi < -0.5$, then resulting ML estimators may not have the standard asymptotic properties [77], whereas if $\xi > 1$, then the conditional mean is not well-defined. Without proper initialization, the DNN will struggle to converge to a feasible solution with acceptable values of the GEV parameters. Thus, controlling the initial estimate of the parameters is difficult but necessary.

To overcome these challenges, we propose a novel framework called `DeepExtrema` that utilizes the GEV distribution to characterize the distribution of block maximum values for a given forecast time window. The parameters of the GEV distribution are estimated using a DNN, which is trained to capture the nonlinear dependencies in the time series data. This is a major departure from previous work by [34], where the distribution parameters are assumed to be a fixed, user-specified hyperparameter. `DeepExtrema` reparameterizes the GEV formulation to ensure that the DNN output is compliant with the GEV positivity constraints. In addition, `DeepExtrema` offers a novel, model bias offset mechanism in order to ensure that the regularity conditions of the GEV parameters are satisfied from the beginning.

In summary, the main contributions of the work are:

1. We present a novel framework to predict the block maxima of a given time window by incorporating GEV distribution into the training of a DNN.
2. We propose a reformulation of the GEV constraints to ensure they can be enforced using activation functions in the DNN.
3. We introduce a model bias offset mechanism to ensure that the DNN output preserves the regularity conditions of the GEV parameters despite its random initialization.
4. We perform extensive experiments on both real-world and synthetic data to demonstrate the effectiveness of `DeepExtrema` compared to other baseline methods.

3.2 Preliminaries

3.2.1 Problem Statement

Let $z_1, z_2, \cdots, z_T$ be a time series of length T. Assume the time series is partitioned into a set of time windows, where each window $[t-\alpha, t+\beta]$ contains a sequence of predictors, $x_t = (z_{t-\alpha}, z_{t-\alpha+1}, \cdots, z_t)$, and target, $\tilde{y}_t = (z_{t+1}, z_{t+2}, \cdots, z_{t+\beta})$. Note that β is known as the forecast horizon of the prediction. For each time window, let $y_t = \max_{\tau \in \{1,\cdots,\beta\}} z_{t+\tau}$ be the block maxima of the target variable at time t. Our time series forecasting task is to estimate the block maxima, $\hat{y}_t$, as well as its upper and lower quantile estimates, $\hat{y}_U$ and $\hat{y}_L$, of a future time window based on current and past data, x_t.

3.2.2 Generalized Extreme Value Distribution

The GEV distribution governs the distribution of block maxima in a given window. Let $Y = \max\{z_1, z_2, \cdots, z_t\}$. If there exist sequences of constants $a_t > 0$ and b_t such that

$$Pr(Y - b_t)/a_t \leq y \rightarrow G(y) \quad \text{as } t \rightarrow \infty$$

for a non-degenerate distribution G, then the cumulative distribution function G belongs to a family of GEV distribution of the form [10]:

$$G(y) = \exp\left\{ -\left[1 + \xi\left(\frac{y-\mu}{\sigma}\right)\right]^{-1/\xi} \right\} \tag{3.1}$$

The GEV distribution is characterized by the following parameters: μ (location), σ (scale), and ξ (shape). The expected value of the distribution is given by

$$y_{mean} = \mu + \frac{\sigma}{\xi}\left[\Gamma(1-\xi) - 1\right] \tag{3.2}$$

where $\Gamma(x)$ denotes the gamma function of a variable $x > 0$. Thus, y_{mean} is only well-defined for $\xi < 1$. Furthermore, the p^{th} quantile of the GEV distribution, y_p, can be calculated as follows:

$$y_p = \mu + \frac{\sigma}{\xi}\left[(-\log p)^{-\xi} - 1\right] \tag{3.3}$$

Given n independent block maxima values, $\{y_1, y_2, \cdots, y_n\}$, with the distribution function given by Equation (3.1) and assuming $\xi \neq 0$, its log-likelihood function is given by:

$$\ell_{GEV}(\mu, \sigma, \xi) = -n \log \sigma - (\frac{1}{\xi} + 1) \sum_{i=1}^{n} \log(1 + \xi \frac{y_i - \mu}{\sigma}) - \sum_{i=1}^{n} (1 + \xi \frac{y_i - \mu}{\sigma})^{-1/\xi} \tag{3.4}$$

The GEV parameters (μ, σ, ξ) can be estimated using the maximum likelihood (ML) approach by maximizing (3.4) subject to the following positivity constraints:

$$\sigma > 0 \quad \text{and} \quad \forall i : 1 + \frac{\xi}{\sigma}(y_i - \mu) > 0 \tag{3.5}$$

In addition to the above positivity constraints, the shape parameter ξ must be within a certain range of values in order for the ML estimators to exist and have regular asymptotic properties [10]. Specifically, the ML estimators have regular asymptotic properties as long as $\xi > -0.5$. Otherwise, if $-1 < \xi < -0.5$, then the ML estimators may exist but will not have regular asymptotic properties. Finally, the ML estimators do not exist if $\xi < -1$ [77].

3.3 Proposed Framework: DeepExtrema

This section presents the proposed `DeepExtrema` framework for predicting the block maxima of a given time window. The predicted block maxima $\hat{y}$ follows a GEV distribution, whose parameters are conditioned on observations of the predictors x. Figure 3.2 provides an overview of the `DeepExtrema` architecture. Given the input predictors x, the framework uses a stacked LSTM network to learn a representation of the time series. The LSTM will output a latent representation, which will used by a fully connected layer to generate the GEV parameters:

$$(\mu, \sigma, \xi_u, \xi_l) = LSTM(x) \tag{3.6}$$

where μ, σ, and ξ's are the location, shape, and scale parameters of the GEV distribution. Note that ξ_u and ξ_l are the estimated parameters due to the reformulation of the GEV constraints, which will be described in the next subsection.

The proposed Model Bias Offset (MBO) component performs bias correction on the estimated GEV parameters to ensure that the LSTM outputs preserve the regularity conditions of the GEV

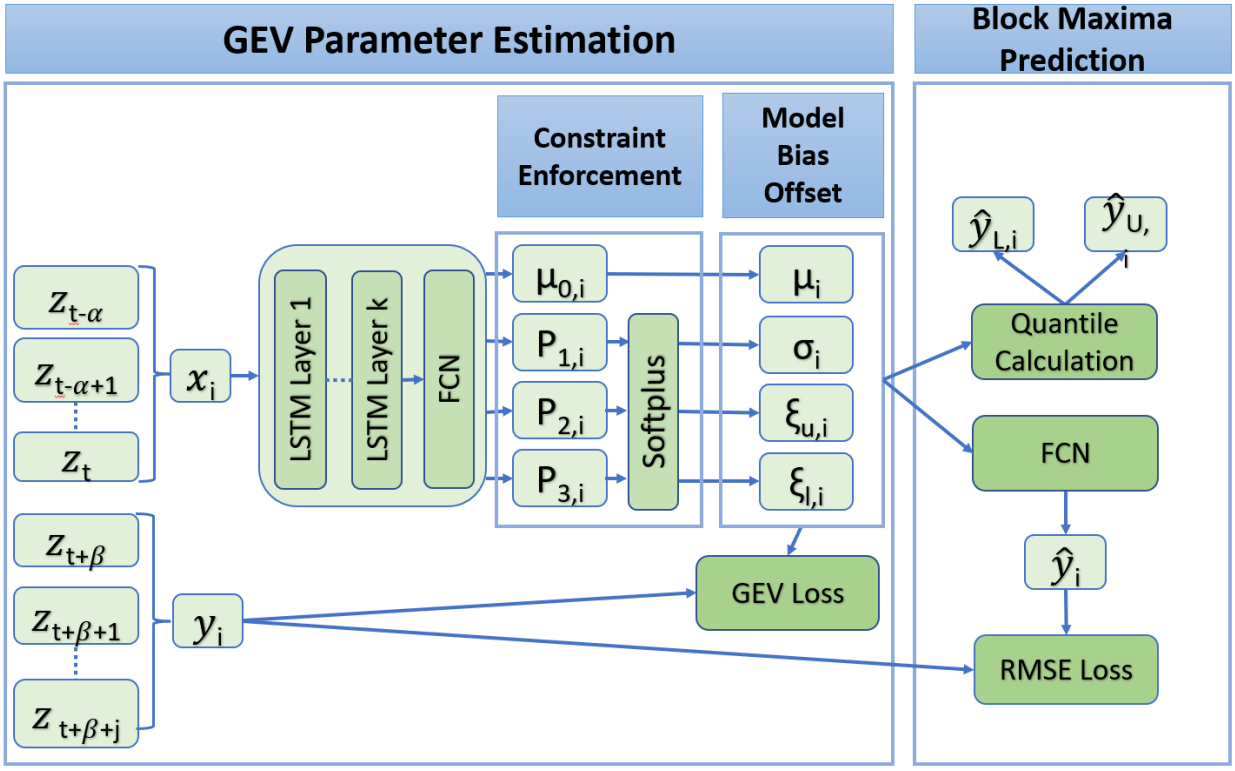


Figure 3.2 Proposed `DeepExtrema` framework for predicting block maxima using GEV distribution.

parameters irrespective of how the network was initialized. The GEV parameters are subsequently provided to a fully connected layer to obtain point estimates of the block maxima, which include its expected value $\hat{y}$ as well as upper and lower quantiles, $\hat{y}_U$ and $\hat{y}_L$, using the equations given in (3.2) and (3.3), respectively. The GEV parameters are then used to compute the negative log-likelihood of the estimated GEV distribution, which will be combined with the root-mean-square error (RMSE) of the predicted block maxima to determine the overall loss function. Details of the different components are described in the subsections below.

3.3.1 GEV Parameter Estimation

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ be a set of training examples, where each x_i denotes the predictor time series and y_i is the corresponding block maxima for time window i. A naïve approach is to assume that the GEV parameters (μ, σ, ξ) are constants for all time windows. This can be done by fitting a global GEV distribution to the set of block maxima values y_i's using the maximum likelihood approach given in (3.4). Instead of using a global GEV distribution with fixed parameters, our

goal is to learn the parameters (μ_i, σ_i, ξ_i) of each window i using the predictors x_i. The added flexibility enables the model to improve the accuracy of its block maxima prediction, especially for non-stationary time series.

The estimated GEV parameters generated by the LSTM must satisfy the two positivity constraints given by the inequalities in (3.5). While the first positivity constraint on σ_i is straightforward to enforce, maintaining the second one is harder as it involves a nonlinear relationship between y_i and the estimated GEV parameters, ξ_i, μ_i, and σ_i. The GEV parameters may vary from one input x_i to another, and thus, learning them from the limited training examples is a challenge. Worse still, some of the estimated GEV parameters could be erroneous, especially at the initial rounds of the training epochs, making it difficult to satisfy the constraints throughout the learning process.

To address these challenges, we propose a reformulation of the second constraint in (3.5). This allows the training process to proceed even though the second constraint in (3.5) has yet to be satisfied especially in the early rounds of the training epochs. Specifically, we relax the hard constraint by adding a small tolerance factor, $\tau > 0$, as follows:

$$\forall\, i : 1 + \frac{\xi}{\sigma}(y_i - \mu) + \tau \geq 0. \tag{3.7}$$

The preceding soft constraint allows for minor violations of the second constraint in (3.5) as long as $1 + \frac{\xi}{\sigma}(y_i - \mu) > -\tau$ for all time windows i. Furthermore, to ensure that the inequality holds for all y_i's, we reformulate the constraint in (3.7) in terms of $y_{\min} = \min_i y_i$ and $y_{\max} = \max_i y_i$ as follows:

Theorem 1. *Assuming $\xi \neq 0$, the soft constraint in* (3.7) *can be reformulated into the following bounds on ξ:*

$$-\frac{\sigma}{y_{\max} - \mu}\,(1+\tau) \leq \xi \leq \frac{\sigma}{\mu - y_{\min}}\,(1+\tau) \tag{3.8}$$

where τ is the tolerance on the constraint in 3.5.

Proof. For the lower bound on ξ, set y_i to be $y_{\max}$ in (3.7):

$$\begin{aligned} 1 + \frac{\xi}{\sigma}(y_{\max} - \mu) + \tau \geq 0 \implies & \frac{\xi}{\sigma}(y_{\max} - \mu) \geq -\,(1+\tau) \\ \implies & \xi \geq -\frac{\sigma}{(y_{\max} - \mu)}\,(1+\tau) \end{aligned}$$

To obtain the upper bound on ξ, set y_i to be $y_{\min}$ in (3.7):

$$1 + \frac{\xi}{\sigma}(y_{\min} - \mu) + \tau \geq 0 \implies \frac{\xi}{\sigma}(\mu - y_{\min}) \leq (1 + \tau)$$
$$\implies \xi \leq \frac{\sigma}{(\mu - y_{\min})}(1 + \tau)$$

□

Following Theorem 1, the upper and lower bound constraints on ξ in (3.8) can be restated as follows:

$$\frac{\sigma}{\mu - y_{\min}}(1 + \tau) - \xi \geq 0$$
$$\xi + \frac{\sigma}{y_{\max} - \mu}(1 + \tau) \geq 0 \tag{3.9}$$

The reformulation imposes lower and upper bounds on ξ, which can be used to re-parameterize the second constraint in (3.5). Next, we describe how the reformulated constraints in (3.9) can be enforced by `DeepExtrema` in a DNN.

Given an input x_i, `DeepExtrema` will generate the following four outputs: μ_i, P_{1i}, P_{2i}, and P_{3i}. A softplus activation function, $softplus(x) = \log(1 + \exp(x))$, which is a smooth approximation to the ReLU function, is used to enforce the non-negativity constraints associated with the GEV parameters. The scale parameter σ_i can be computed using the softplus activation function on P_{1i} as follows:

$$\sigma_i = softplus(P_{1i}) \tag{3.10}$$

This ensures the constraint $\sigma_i \geq 0$ is met. The lower and upper bound constraints on ξ_i given by the inequalities in (3.9) are enforced using the softplus function on P_{2i} and P_{3i}:

$$\frac{\sigma_i}{\mu_i - y_{\min}}(1 + \tau) - \xi_{u,i} = softplus(P_{2i})$$
$$\frac{\sigma_i}{y_{\max} - \mu_i}(1 + \tau) + \xi_{l,i} = softplus(P_{3i}) \tag{3.11}$$

By re-arranging the above equation, we obtain

$$\xi_{u,i} = \frac{\sigma_i}{\mu_i - y_{\min}}(1 + \tau) - softplus(P_{2i})$$
$$\xi_{l,i} = softplus(P_{3i}) - \frac{\sigma_i}{y_{\max} - \mu_i}(1 + \tau) \tag{3.12}$$

`DeepExtrema` computes the upper and lower bounds on ξ_i using the formulas in (3.12). During training, it will minimize the distance between $\xi_{u,i}$ and $\xi_{l,i}$ and will use the value of $\xi_{u,i}$ as the estimate for ξ_i. Note that the two ξ_i's converge rapidly to a single value after a small number of training epochs.

3.3.2 Model Bias Offset (MBO)

Although the constraint reformulation approach described in the previous subsection ensures that the DNN outputs will satisfy the GEV constraints, the random initialization of the network can produce estimates of ξ that violate the regularity conditions described in Section 3.2.2. Specifically, the ML-estimated distribution may not have the asymptotic GEV distribution when $\xi < -0.5$ while its conditional mean is not well-defined when $\xi > 1$. Additionally, the estimated location parameter μ may not fall within the desired range between $y_{\min}$ and $y_{\max}$ when the DNN is randomly initialized. Thus, without proper initialization, the DNN will struggle to converge to a good solution and produce acceptable values of the GEV parameters.

One way to address this challenge is to repeat the random initialization of the DNN until a reasonable set of initial GEV parameters, i.e., $y_{\min} \leq \mu \leq y_{\max}$ and $-0.5 < \xi < 1$, is found. However, this approach is infeasible given the size of the parameter space of the DNN. A better strategy is to control the initial output of the neural network in order to produce an acceptable set of GEV parameters, (μ, σ, ξ) during initialization. Unfortunately, controlling the initial output of a neural network is difficult given its complex architecture.

We introduce a simple but effective technique called Model Bias Offset (MBO) to address this challenge. The key insight here is to view the GEV parameters as a biased output due to the random initialization of the DNN and then perform bias correction to alleviate the effect of the initialization. To do this, let μ_{desired}, σ_{desired}, and ξ_{desired} be an acceptable set of initial GEV parameters. The values of these initial parameters must satisfy the regularity conditions $-0.5 < \xi_{\text{desired}} < 1$, $\sigma_{\text{desired}} > 0$, and $y_{\min} \leq \mu_{\text{desired}} \leq y_{\max}$. This can be done by randomly choosing a value from the preceding range of acceptable values, or more intelligently, using the GEV parameters estimated from a global GEV distribution fitted to the block maxima y_i's in the training data via the ML approach given in

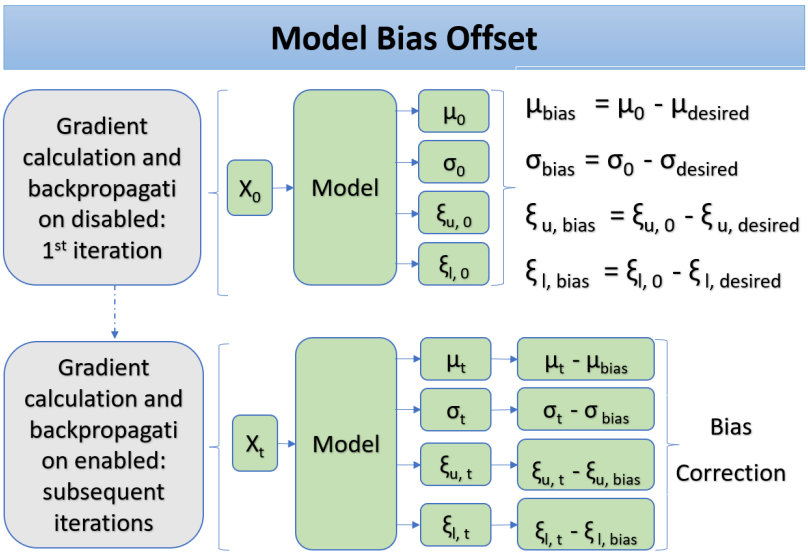


Figure 3.3 Model Bias Offset (MBO) to ensure the initial estimates of the GEV parameters are reasonable even when the DNN is randomly initialized.

(3.4), without considering the input predictors (see the discussion at the beginning of Section 3.3.1). We find that the latter strategy works well in practice as it can lead to faster convergence especially when the global GEV parameters are close to the final values after training.

When the DNN is randomly initialized, let $\mu_0, \sigma_0, \xi_{u,0}$, and $\xi_{l,0}$ be the initial DNN output for the GEV parameters. These initial outputs may not necessarily fall within their respective range of acceptable values. We consider the difference between the initial DNN output and the desired GEV parameters as a *model bias* due to the random initialization:

$$\mu_{\text{bias}} = \mu_0 - \mu_{\text{desired}} \qquad \sigma_{\text{bias}} = \sigma_0 - \sigma_{\text{desired}}$$
$$\xi_{u,\text{bias}} = \xi_{u,0} - \xi_{u,\text{desired}} \qquad \xi_{l,\text{bias}} = \xi_{l,0} - \xi_{l,\text{desired}} \tag{3.13}$$

The model bias terms in (3.14) can be computed during the initial forward pass of the algorithm. The gradient calculation and back-propagation are disabled during this step to prevent the DNN from computing the loss and updating its weight with the unacceptable GEV parameters. After the initial iteration, the gradient calculation will be enabled and the bias terms will be subtracted from

the DNN estimate of the GEV parameters in all subsequent iterations t:

$$\mu_t \rightarrow \mu_t - \mu_{\text{bias}} \qquad \sigma_t \rightarrow \sigma_t - \sigma_{\text{bias}}$$
$$\xi_{u,t} \rightarrow \xi_{u,t} - \xi_{u,\text{bias}} \qquad \xi_{l,t} \rightarrow \xi_{l,t} - \xi_{l,\text{bias}} \tag{3.14}$$

Note that, when μ_t is set to μ_0, then the debiased output $\mu_t - \mu_{\text{bias}}$ will be equal to μ_{desired}. By debiasing the output of the DNN in this way, we guarantee that the initial GEV parameters satisfy the GEV regularity conditions.

3.3.3 Block Maxima Prediction

Given an input x_i, the DNN will estimate the GEV parameters needed to compute the block maxima $\hat{y}_i$ along with its upper and lower quantiles, $\hat{y}_{U,i}$ and $\hat{y}_{L,i}$, respectively. The quantiles are estimated using the formula given in (3.3). The GEV parameters are provided as input to a fully connected network (FCN) to generate the block maxima prediction, $\hat{y}_i$.

`DeepExtrema` employs a combination of the negative log-likelihood function ($-\ell_{GEV}(\mu, \sigma, \xi)$) of the GEV distribution and a least-square loss function to train the model. This enables the framework to simultaneously learn the GEV parameters and make accurate block maxima predictions. The loss function to be minimized by `DeepExtrema` is:

$$\mathcal{L} = \lambda_1 \, \hat{\mathcal{L}} + (1 - \lambda_1) \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.15}$$

where $\hat{\mathcal{L}} = -\lambda_2 \, \ell_{GEV}(\mu, \sigma, \xi) + (1 - \lambda_2) \sum_{i=1}^{n} (\xi_{u,i} - \xi_{l,i})^2$ is the regularized GEV loss. The first term in $\hat{\mathcal{L}}$ corresponds to the negative log-likelihood function given in Equation (3.4) while the second term minimizes the difference between the upper and lower-bound estimates of ξ. The loss function $\mathcal{L}$ combines the regularized GEV loss ($\hat{\mathcal{L}}$) with the least-square loss. Here, λ_1 and λ_2 are hyperparameters to manage the trade-off between different factors of the loss function.

3.4 Experimental Evaluation

This section presents our experimental results comparing `DeepExtrema` against the various baseline methods. The code and datasets are available at https://github.com/galib19/DeepExtrema-IJCAI22-.

3.4.1 Data

3.4.1.1 Synthetic Data

As the ground truth GEV parameters are often unknown, we have created a synthetic dataset to assess our method's ability to correctly infer the GEV parameters. The data is generated assuming the GEV parameters are functions of some input predictors $x \in \mathbb{R}^6$. We first generate x by random sampling from a uniform distribution. We then assume a non-linear mapping from x to the GEV parameters μ, σ, and ξ, via the following nonlinear equations:

$$\mu(x) = w_\mu^T(\exp(x) + x) \qquad \sigma(x) = w_\sigma^T(\exp(x) + x)$$
$$\xi(x) = w_\xi^T(\exp(x) + x) \tag{3.16}$$

where w_μ, w_σ, and w_ξ are generated from a standard normal distribution. Using the generated μ, σ, and ξ parameters, we then randomly sample y from the GEV distribution governed by the GEV parameters. Here, y denotes the block maxima as it is generated from a GEV distribution. We created 8,192 block maxima values for our synthetic data.

3.4.1.2 Real World Data

We consider the following 3 datasets for our experiments.

Hurricane: This corresponds to tropical cyclone intensity data obtained from the HURDAT2 database [78]. There are altogether 3,111 hurricanes spanning the period between 1851 and 2019. For each hurricane, wind speeds (intensities) were reported at every 6-hour interval. We consider only hurricanes that have at least 24-time steps at minimum for our experiments. For each hurricane, we have created non-overlapping time windows of length 24 time steps (6 days). We use the first 16 time steps (4 days) in the window as the predictor variables and the block maxima of the last 8 time steps (2 days) as the target variable.

Solar: This corresponds to half-hourly energy use (kWh) for 55 families over the course of 284 days from the Ausgrid database [79]. We preprocess the data by creating non-overlapping time windows of length 192 time steps (4 days). We use the first 144 time steps (3 days) in the window as the predictor variables and the block maxima of the last 48 time steps (1 day) as the target variable.

Weather: We have used a weather dataset from the Kaggle competition [80]. The data is based on hourly temperature data for a city over a ten-year period. We use the first 16 time steps (16 hours) in the window as the predictor variables and the block maxima of the last 8 time steps (8 hours) as the target variable.

3.4.2 Experimental Setup

For evaluation purposes, we split the data into separate training, validation, and testing with a ratio of 7:2:1. The data is standardized to have zero mean and unit variance. We compare `DeepExtrema` against the following baseline methods: (1) Persistence, which uses the block maxima value from the previous time step as its predicted value, (2) fully-connected network (FCN), (3) LSTM, (4) Transformer, (5) DeepPIPE [32], and (6) EVL [34]. We will use the following metrics to evaluate the performance of the methods: (1) Root mean squared error (RMSE) and correlation between the predicted and ground truth block maxima and (2) Negative log-likelihood (for synthetic data). Finally, hyperparameter tuning is performed by assessing the model performance on the validation set. The hyperparameters of the baseline and the proposed methods are selected using Ray Tune, a tuning framework with an ASHA (asynchronous successive halving algorithm) scheduler for early stopping.

3.4.3 Experimental Results

3.4.3.1 Results On Synthetic Data

In this experiment, we have compared the performance of `DeepExtrema` against using a single (global) GEV parameter to fit the data. Based on the experiments, `DeepExtrema` achieves a substantially lower negative log-likelihood of 4410 than the global GEV estimate, which has a negative log-likelihood of 4745. This result supports the assumption that each block maxima comes from different GEV distributions rather than a single (global) GEV distribution. Figure 3.4 also illustrates that `DeepExtrema` can accurately predict the GEV parameters: location ((μ), scale ((σ), and shape ((ξ).

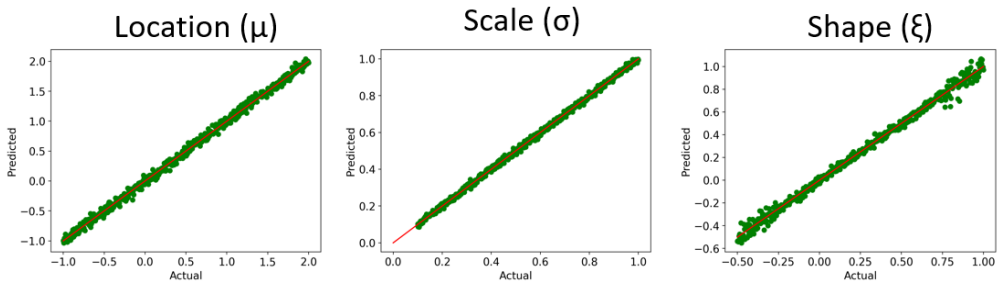


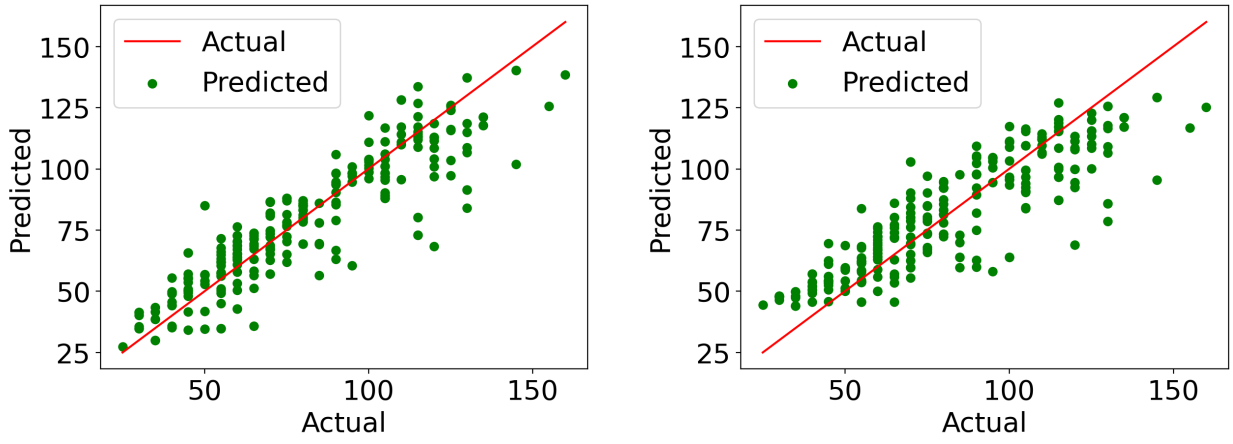
Figure 3.4 Actual and predicted GEV parameters (location, scale, and shape) using `DeepExtrema`.

Methods	Hurricanes		Ausgrid		Weather	
	RMSE	ρ	RMSE	ρ	RMSE	ρ
Persistence	28.6	0.6	0.84	0.65	4.16	0.96
FCN	14.14	0.87	0.69	0.65	2.5	0.97
LSTM	13.31	0.88	0.65	0.64	2.53	0.97
Transformer	13.89	0.88	0.68	0.62	2.43	**0.98**
DeepPIPE	13.67	0.87	0.71	0.59	2.59	0.94
EVL	15.72	0.83	0.75	0.54	2.71	0.90
DeepExtrema	**12.81**	**0.90**	**0.63**	**0.67**	**2.27**	0.97

Table 3.1 Performance comparison on real-world data using RMSE and Correlation (ρ).

3.4.3.2 Results On Real World Data

Evaluation of real-world data shows that `DeepExtrema` outperforms other baseline methods used for comparison for all data sets (see Table 3.1). For RMSE, `DeepExtrema` generates lower RMSE compared to all the baselines on all 3 datasets, whereas for correlation, `DeepExtrema` outperforms the baselines on 2 of the 3 datasets.



(a) `DeepExtrema` (b) `EVL` [34]

Figure 3.5 Comparison between Actual and predicted block maxima of hurricane intensities for `DeepExtrema` and `EVL` [34].

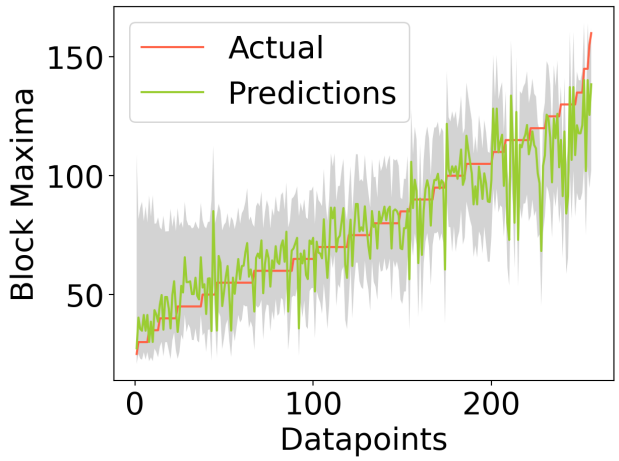


Figure 3.6 90% confidence interval of the hurricane intensity predictions for `DeepExtrema`, sorted in increasing block maxima values. Ground truth values are shown in red.

To demonstrate how well the model predicts the extreme values, Figure 3.5 shows a scatter plot of the actual versus predicted values generated by `DeepExtrema` on the test set of the hurricane intensity data. The results suggest that `DeepExtrema` can accurately predict hurricane intensities for a wide range of values, especially those below 140 knots. `DeepExtrema` also does a better job at predicting the high-intensity hurricanes compared to `EVL` [34]. Figure 3.6 shows the 90% confidence interval of the predictions. Apart from the point and quantile estimations, `DeepExtrema` can also estimate the GEV parameter values for each hurricane.

3.4.3.3 Ablation Studies

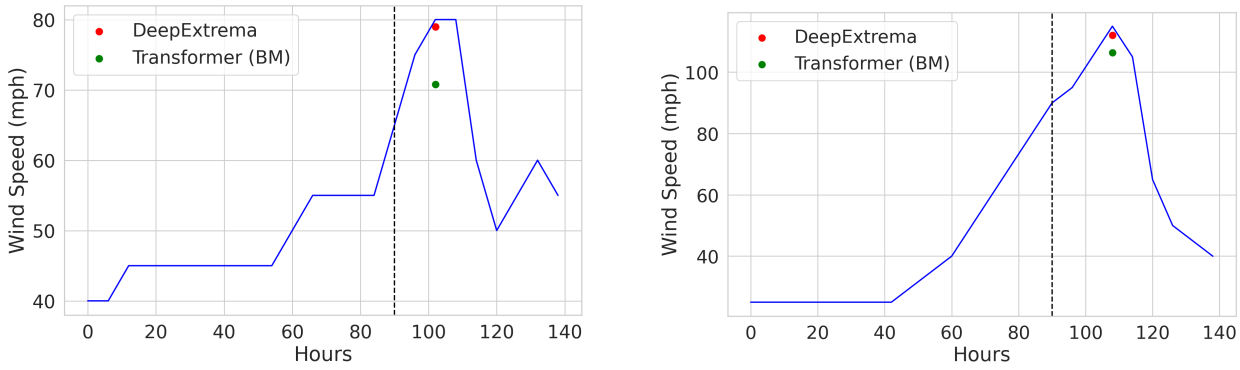
The hyperparameter λ_1 in the objective function of `DeepExtrema` denotes the trade-off between regularized GEV loss and RMSE loss. Experimental results show that the RMSE of block maxima prediction decreases when λ_1 increases (see Table 3.2). This validates the importance of incorporating GEV theory to improve the accuracy of block maxima estimation instead of using the mean squared loss alone.

3.4.3.4 Case Study

To further illustrate the efficacy of our proposed approach, `DeepExtrema`, we conducted a case study focused on forecasting maximum hurricane intensity. This case study analyzed two samples from our test dataset: Major Hurricane Harvey (2017) and Tropical Storm Ernesto (2012). Major Hurricane Harvey was a particularly devastating event, ranking as the second most expensive

Hyperparameter	RMSE of Block maxima		
	Hurricanes	Ausgrid	Weather
$\lambda_1 = 0.0$	13.28	0.68	2.52
$\lambda_1 = 0.5$	13.03	**0.63**	2.41
$\lambda_1 = 0.9$	**12.81**	0.64	**2.27**

Table 3.2 Effect of λ_1 on RMSE of block maxima prediction.



(a) Major Hurricane Harvey (2017)

(b) Tropical Storm Ernesto (2012)

Figure 3.7 Case Study: Forecasting Maximum Hurricane Intensity using `DeepExtrema` and Transformer.

hurricane in U.S. history, causing $152.2 billion in damages and claiming at least 103 lives. Tropical Storm Ernesto also had significant impacts, with heavy rainfall and flooding resulting in $500 million in losses.

In this study, we aimed to predict the maximum intensity for the next two days based on the intensity values from the preceding four days. By employing our `DeepExtrema` framework alongside the second-best performing baseline model, Transformer, we observed notable differences in their forecasting performance. As illustrated in Figure 3.7, `DeepExtrema` demonstrated a remarkably close alignment with the ground truth in forecasting maximum intensity. In contrast, the Transformer model consistently underestimated the maximum intensity for both hurricane events.

This case study highlights the effectiveness and accuracy of `DeepExtrema` in capturing and forecasting extreme values in time series data, particularly in the context of high-impact weather events like hurricanes. The superior performance of our proposed approach compared to the baseline model further reinforces the potential of `DeepExtrema` in enabling more accurate predictions of

extreme events, which can have far-reaching implications for disaster preparedness, risk mitigation, and resource allocation strategies.

3.5 Conclusion

This work presents a novel deep learning framework called `DeepExtrema` that combines extreme value theory with deep learning to address the challenges of predicting extremes in time series. We offer a reformulation and re-parameterization technique for satisfying constraints and a model bias offset technique for proper model initialization. We evaluated our framework on synthetic and real-world data and showed its effectiveness. For future work, we plan to extend the formulation to enable more complex deep learning architectures such as attention mechanisms. Furthermore, the system will be expanded to model spatiotemporal extremes.

CHAPTER 4

SELF-RECOVER: FORECASTING BLOCK MAXIMA IN TIME SERIES FROM PREDICTORS WITH DISPARATE TEMPORAL COVERAGE USING SELF-SUPERVISED LEARNING

4.1 Introduction

Forecasting of extreme values such as block maxima in a future time window is an important but challenging problem. Accurate forecasting of the block maxima enables us to anticipate the worst-case scenario expected to occur within the forecast period. As mentioned in chapter 2, deep learning [2, 3, 4, 1, 20, 24, 23] has become increasingly popular in recent years for time series forecasting due to their capacity to learn complex nonlinear relationships in data. As outlined in Chapter 1, these approaches are primarily designed to predict the conditional mean rather than the tail distribution, which often limits their effectiveness in forecasting extreme values. This has led to considerable interest in incorporating sound statistical principles from extreme value theory (EVT) into the deep learning formulation [81, 36, 38, 34, 35].

Another challenge in forecasting block maxima is the limited availability of extreme historical observations. Towards this end, process-based models have been developed in many domains to incorporate our understanding of the physical processes that regulate the dynamical system. For example, general circulation models (GCMs) are widely-used process-based models for generating robust representations of future climate scenarios. The forecasts generated from these process-based models can be used as domain-informed predictors that can be coupled with historical observations to enhance the performance of time series forecasting. However, a key challenge in integrating the

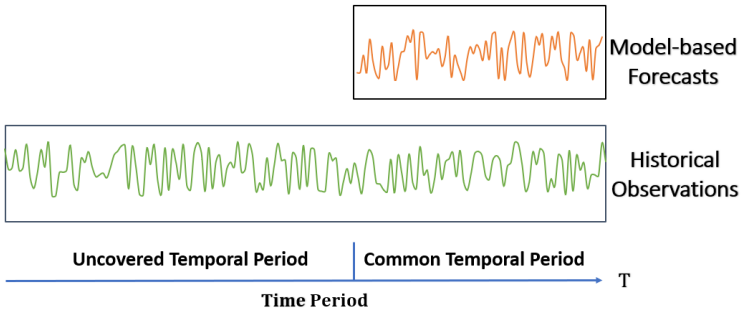


Figure 4.1 Disparate temporal coverage between historical observations and process-based model outputs.

process-based models forecasts with historical observations is their disparate temporal coverage, as shown in Figure 4.1. For example, long-term historical records of climate observations, dating back before the 20th century, are available for various locations, whereas archived scenarios of GCM runs may only be available for the 20th century and beyond since scenario development is time-consuming and costly [82]. The presence of large blocks of missing values can significantly impede the ability to make accurate predictions [83].

Self-supervised learning [84, 85] offers a possible approach to address this challenge. Self-supervised learning is well-suited for learning representations from available data and for generating new data. This approach can help address the challenges presented by missing or disparately distributed predictor variables in time series. By using self-supervised techniques, it is possible to extract useful information from the data that is available and to generate new data that can be used to fill in gaps in the original data set. This can improve the accuracy of predictions and increase the overall utility of the data. However, despite its success in various domains such as image and language processing, its application in time series imputation remains relatively under-explored. In particular, how to account for the missing values from one of the two sources and integrate their representations to train an accurate DNN model are two key issues that need to be addressed.

In this work, we present a framework called `Self-Recover` to enable accurate forecasts of block maxima in time series using both historical observations and model-based forecasts in addressing the issue of disparate temporal coverage. A combination of contrastive and generative self-supervised schemes followed by a denoising autoencoder is used to impute long-term and random missing values of model-based forecasts. Also, `Self-Recover` employs a residual learning technique to combine the representations of historical and model-based forecasts. The combined representation is used to generate parameters of the generalized extreme value (GEV) distribution characterizing the block maxima values. It also uses a reparameterization technique to ensure that the DNN output adheres to the GEV positivity constraints [10] and a model bias offset technique [81] to ensure that the GEV parameters' regularity conditions are satisfied in spite of the random initialization of DNNs.

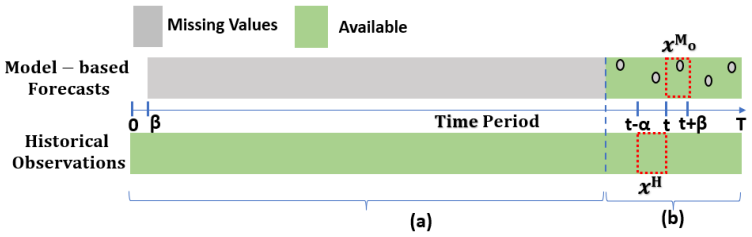


Figure 4.2 Illustration of the long-term and random missing values for model-based forecasts. Time period (a) is the case of long-term missing values where all of the model-based forecasts are missing. Time period (b) is the case of random missing values where a few of the model-based forecasts are missing.

In summary, the main contributions of the work are:

1. We present a novel framework, `Self-Recover`, to forecast the block maxima of a future time window while simultaneously recover missing values from the model-based forecasts.

2. To impute the missing values, a novel self-supervised learning approach is proposed by integrating contrastive and generative schemes followed by a denoising autoencoder. This approach can impute both structured and random missing values in the model-based forecasts.

3. We present a residual technique to combine the representations from historical and model-based forecasts and show that it is better than simple concatenation of the two features.

4. We perform extensive experiments on real-world data to demonstrate the effectiveness of `Self-Recover` compared to state-of-the-art time series forecasting methods.

4.2 Preliminaries

4.2.1 Problem Statement

Consider a time series data that is partitioned into a set of overlapping time windows, $\{w_1, w_2, \cdots, w_n\}$, where each window w_i is defined by its start time, $t_i - \alpha$, and end time, $t_i + \beta$. Let the sample of historical observation values is denoted by $z^H_{t-\alpha}, z^H_{t-\alpha+1}, \cdots, z^H_t, z^H_{t+1}, \cdots, z^H_{t+\beta}$ for time window $[t-\alpha, t+\beta]$, where each $z^H_i \in \mathbb{R}$. We assume the time window $[t-\alpha, t]$ contains historical observations while the time window $[t+1, t+\beta]$ contains the future values to be predicted, where β is the forecast horizon. We further assume there are m process model forecasts associated with some of the time windows. Let $z^M_t, z^M_{t+1}, \cdots, z^M_{t+\beta}$, be the sample of model-based forecasts, where each $z^M_i \in \mathbb{R}^m$. For each time window $[t-\alpha, t+\beta]$, let $x^H_t = (z^H_{t-\alpha}, z^H_{t-\alpha+1}, \cdots, z^H_t) \in \mathbb{R}^{\alpha+1}$

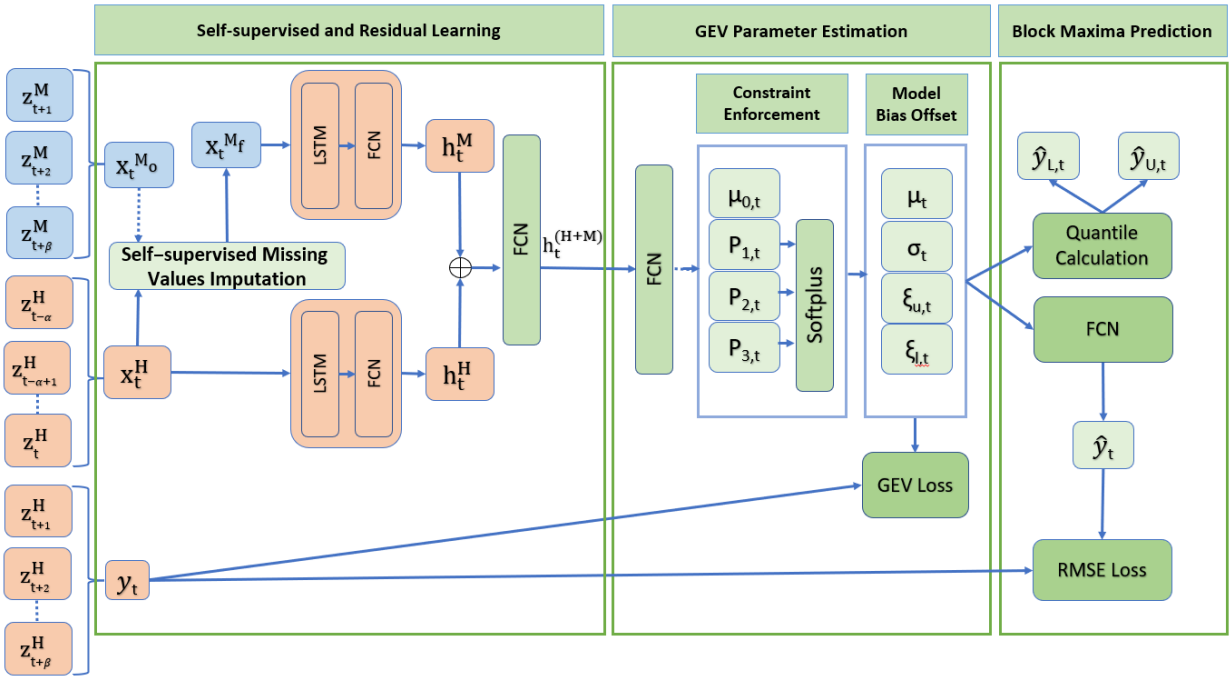


Figure 4.3 `Self-Recover:` Proposed framework for forecasting block maxima by using both historical observations and model-based forecasts as predictors along with the generalized extreme value (GEV) distribution.

be the historical predictors, $x_t^{M_0} = (z_t^M, z_{t+1}^M, \cdots, z_{t+\beta}^M) \in \mathbb{R}^{m\times\beta}$ be the initial model-based forecasts and $y_t = \max_{\tau\in\{1,\cdots,\beta\}} z_{t+\tau}^{(H)} \in \mathbb{R}$ be the target variable (block maxima) to be predicted. The initial model-based forecasts $x_t^{M_\circ}$ could be partially or completely missing as illustrated in Figure 4.2.

Our initial objective is to impute the partially and completely missing values of the initial model-based forecasts $(x_t^{M_0})$ for the future time window $[t+1, t+\beta]$ using the historical predictors (x_t^H) in $[t-\alpha, t]$. We will train a model $F_{impute} : \mathbb{R}^{\alpha+1} \longrightarrow \mathbb{R}^{m\times\beta}$ to impute the missing values in the initial model-based forecasts $x_t^{M_0}$ and get the full imputed model-based forecasts $x_t^{M_f} = F_{impute}(x_t^H)$.

Our final objective is to predict the block maxima value (y_t) of the future time window $[t+1, t+\beta]$ using both the historical predictors (x_t^H) in $[t-\alpha, t]$ and model-based forecasts $(x_t^{M_f})$ in the future time window $[t+1, t+\beta]$. We will train a model $F_{forecast} : \mathbb{R}^{\alpha+1} \times \mathbb{R}^{m\times\beta} \longrightarrow \mathbb{R}$ to predict the block maxima value, $\hat{y}_t = F_{forecast}(x_t^H, x_t^{M_f})$, of the time window along with its upper and lower quantiles, $\hat{y}_U$ and $\hat{y}_L$.

4.2.2 DeepExtrema Framework

The DeepExtrema framework [81] is designed to predict the block maxima of a time window by incorporating the Generalized Extreme Value (GEV) distribution [10] given in (3.1) into the training of deep neural networks (DNNs). DeepExtrema learns the parameters of the GEV distribution and predicts the block maxima using DNNs. Furthermore, it can produce estimates of uncertainties in its prediction using the p^{th} quantile of the GEV distribution, y_p using (3.3).

Given training data consisting of n block maxima values, $\{y_1, y_2, \cdots, y_n\}$, DeepExtrema is trained to minimize the mean-square error of the block maxima prediction as well as the negative log-likelihood function in (3.4) to ensure its predictions are consistent with the GEV distribution.

DeepExtrema also reformulates the GEV constraints given in (3.5) and reparameterizes the DNN output accordingly to ensure the constraints can be enforced during DNN training. It further uses a novel model bias offset mechanism to ensure that the DNN output preserves the GEV constraints in spite of its random initialization. Specifically, it computes the bias introduced by the randomly initialized network and performs bias correction to mitigate its effect in subsequent training epochs. This strategy of debiasing the DNN outputs guarantees that the GEV parameters estimated by the DNN would satisfy the regularity conditions at all times even if the DNN is not properly initialized.

4.3 Proposed Self-Recover Framework

Self-Recover builds upon the DeepExtrema framework to enable the incorporation of both historical and model-based forecasts into its formulation. Figure 4.3 presents a high-level overview of the proposed Self-Recover framework. Given the historical observations and model-based forecasts, the framework initially imputes the long-term and random missing values. It employs a combination of contrastive and generative self-supervised learning schemes to impute long-term missing values of model-based forecasts. Also, a denoising autoencoder (DAE) is employed to impute random missing values. Then, it employs two separate DNNs to learn individual feature representations of the historical and model-based forecasts. Afterward, it uses a residual technique to augment the learned model-based forecasts representation, h^M, into the representation of the

historical observations, h^H. It then passes the combined representation to the downstream modules for parameter estimation of the GEV distribution and prediction of block maxima. These downstream modules are derived from `DeepExtrema`. Specifically, it assumes the following data generation process for the block maxima $\hat{y}_t$ in each time window $[t-\alpha, t+\beta]$:

$$h^H = g_H(x^H) \quad \text{and} \quad h^M = g_M(x^{M_f}) \tag{4.1}$$

$$(\mu_t, \sigma_t, \xi_t) \quad = \quad f(h^H + \alpha h^M) \tag{4.2}$$

$$\hat{y}_t \quad \sim \quad GEV(\mu_t, \sigma_t, \xi_t) \tag{4.3}$$

Equation (4.1) states that the hidden representations for the historical observations and model-based forecasts are functions of their corresponding predictors. Equation (4.2) states that the GEV parameters are functions of these hidden representations. Finally, the observed block maxima are drawn from the GEV distribution according to Equation (4.3).

4.3.1 Self-supervised Learning for Handling Disparate Temporal Coverage

To deal with the disparate temporal coverage, `Self-Recover` presents a self-supervised approach based on contrastive and generative learning schemes followed by a denoising autoencoder (DAE) for simultaneously imputing long-term and random missing values. The proposed approach is illustrated in Figure 4.4.

First, it employs a contrastive learning scheme to learn representations from historical observations. Contrastive learning [85] is a method of learning representations from unlabeled data by comparing different augmented versions of the same data sample. In the context of time series, this would involve training the model on a set of time series data and then presenting it with pairs of time series data, where one is a positive example (e.g., a similar time series) and the other is a negative example (e.g., a dissimilar time series). The model is then trained to identify the differences between the positive and negative examples.

Given the historical time series x_t^H for each window, it augments the time series to get similar examples $(x_{1,t}^H, x_{2,t}^H)$. For data augmentation, jittering, scaling, and flipping are employed, but jittering performs better for imputation overall. All other samples in the mini-batch are treated

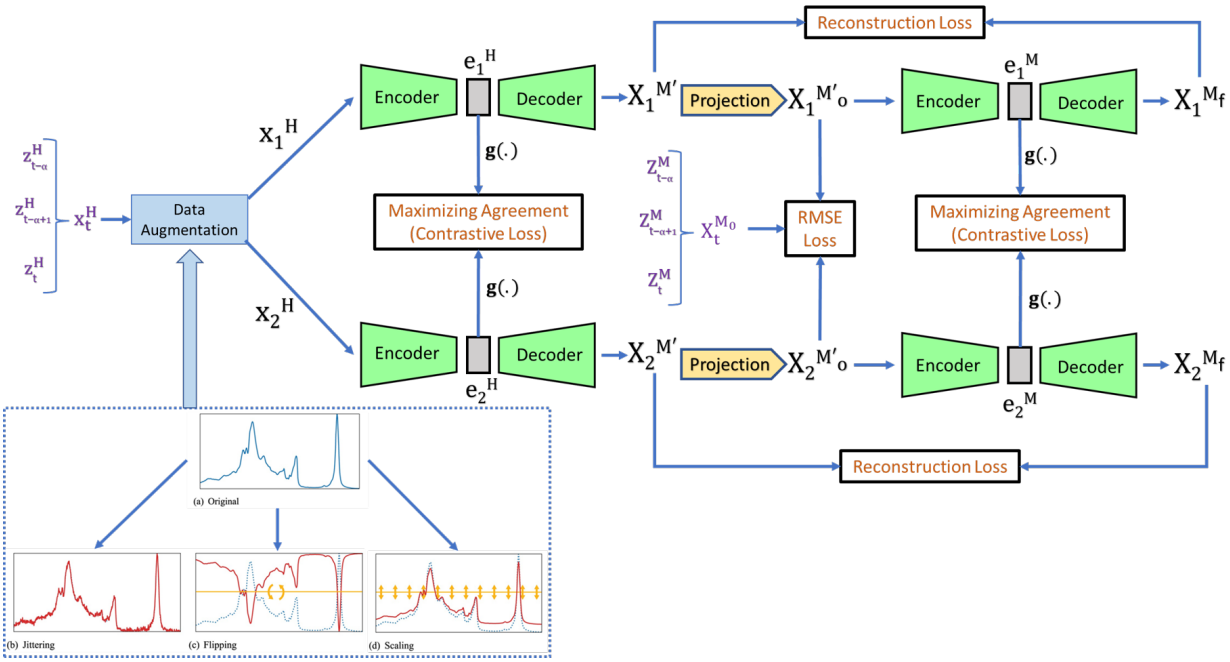


Figure 4.4 Long-term and random missing values imputation with the proposed self-supervised learning approach. For data augmentation, jittering, scaling, and flipping are used.

as negative examples. Then, it passes the samples through an encoder-decoder model to generate model-based forecasts as follows:

$$x_t^{M'} = Decoder(e_t^H) \quad \text{where} \quad e_t^H = Encoder(x_t^H) \tag{4.4}$$

The intermediate embedding (e_t^H) by the encoder is then projected using a projection head $g(\cdot)$ to get the projected values, $c_t^H = g(e_t^H)$. The projected values should be similar for similar input samples. The projection head maps the intermediate embeddings to a lower-dimensional space, where the contrastive loss is calculated. The contrastive loss is a measure of similarity between the projected values of two similar samples. Therefore, the encoder part of the encoder-decoder model is used to learn embeddings of the historical observations, which are used in contrastive loss.

The encoder-decoder model is trained to generate initial model-based forecasts ($x_t^{M'}$) from historical observations (x_t^H). These initial model-based forecasts do not have any missing values. However, they may not be consistent with the true model-based forecasts $x_t^{M'_0}$. Thus, the initial model-based forecasts need to be calibrated with the available ground truth model-based forecasts. To ensure the initial model-based forecasts are robust against noise and to make the framework

capable of imputing random missing values, a denoising autoencoder (DAE) [86] is used. DAE is a type of autoencoder that is trained to reconstruct the original input from a corrupted version of the input. The goal is to learn a representation of the input that is robust to noise. A DAE can also be used for random missing value imputation. The idea is to treat the missing values as noise and train the DAE to reconstruct the original data without the missing values. For creating corrupting data, missing values need to be introduced as noise. To introduce missing values in the initially generated model-based forecasts, we consider two cases. If there are any ground truth model-based forecasts available with missing values, we introduce the same missing values for the generated model-based forecasts as well. In other cases (if there is not any ground truth available or there are not any missing values in the ground truth), we introduce missing values randomly. So, the corrupted model-based forecasts $x_t^{M'_o}$ are generated from the initial model-based forecasts using ground truth guided projection or random projection of missing values. Then, an autoencoder is used to reconstruct the original input from the corrupted data. Also, the intermediate embedding (e_t^M) by the encoder is projected using another projection head $g'(\cdot)$ to get the projected values $c_t^M = g'(e_t^M)$. Ideally, the projected values should be similar for similar input samples. Thereby, another contrastive loss can be used here. Finally, when the ground truth model-based forecasts $x_t^{M_0}$ are available, a root-mean-square error (RMSE) loss can be used to ensure the corrupted model-based forecasts $x_t^{M'_0}$ is close to the ground truth. The output of the autoencoder is the final model-based forecasts ($x_t^{M_f}$).

4.3.2 Optimization of Self-supervised Learning

`Self-Recover` trains the self-supervised imputation in two phases based on time period (a) and (b) as illustrated in Figure 4.2. It starts its training with the time period (b) as both historical and model-based forecasts are available. After several hundred epochs with time period (b) only, it starts simultaneous training with the both time periods (a) and (b). The optimization is carried out this way to take advantage of time period (b) that has the ground truth model-based forecasts. The model-based forecasts in time period (b) would facilitate the training initially with an additional RMSE loss.

For the training batch with the time period (b), combines contrastive loss, reconstruction loss,

and RMSE loss. For contrastive learning, it adopts the NT-Xent (Normalized Temperature-scaled Cross Entropy) loss [87]. The contrastive losses historical ($\ell^{H}_{\text{contrast}}$) and model-based ($\ell^{M}_{\text{contrast}}$) representations are defined for a positive pair of examples (i, j) as follows:

$$\begin{aligned}
\ell^{H}_{\text{contrast}} &= \sum_{i,j} -\log \frac{exp(sim(c^{H}_{t,i}, c^{H}_{t,j}))}{\sum_{k=1}^{2N} \mathbb{1}_{k\neq i} exp(sim(c^{H}_{t,i}, c^{H}_{t,k})} \\
\ell^{M}_{\text{contrast}} &= \sum_{i,j} -\log \frac{exp(sim(c^{M}_{t,i}, c^{M}_{t,j}))}{\sum_{k=1}^{2N} \mathbb{1}_{k\neq i} exp(sim(c^{M}_{t,i}, c^{M}_{t,k})}
\end{aligned} \tag{4.5}$$

For the denoising autoencoder architecture, the reconstruction loss can be expressed as follows:

$$\ell^{M}_{\text{reconstruction}} = \frac{1}{n} \sum_{i=1}^{n} ||x^{M'}_{t,i} - x^{M_f}_{t,i}||^2$$

where $x^{M'}_{t,i}$ is the initial generated model-based forecasts and $x^{M_f}_{t,i}$ is final model-based forecasts for i-th sample.

The RMSE loss is defined between the ground truth $x^{M_0}_{t,i}$ and the projected model-based forecasts $x^{M'_0}_{t,i}$ as follows:

$$\ell^{M}_{\text{RMSE}} = \sqrt{\sum_{i=1}^{n} \frac{(x^{M_0}_{t} - x^{M'_0}_{t})^2}{n}}$$

For the training batch with the time period (b), the overall loss can be expressed as follows:

$$\begin{aligned}
\mathcal{L}_{\text{SSL}} = \quad & \gamma_1 \, \ell^{H}_{\text{contrast}} + \gamma_2 \, \ell^{M}_{\text{contrast}} + \gamma_3 \, \ell^{M}_{\text{reconstruction}} \\
& + (1 - \gamma_1 - \gamma_2 - \gamma_3) \, \ell^{M}_{\text{RMSE}}
\end{aligned} \tag{4.6}$$

where $\gamma_1, \gamma_2, \gamma_3,$ and γ_4 are hyperparameters for the weights.

In summary, the self-supervised loss $\mathcal{L}_{\text{SSL}}$ is a weighted sum of contrastive loss on historical observations, contrastive loss on model-based forecasts, reconstruction loss on model-based forecasts, and RMSE on model-based forecasts. For the training batch with the time period (a), γ_4 is set to 0 as there is not any ground truth for the time period (a).

4.3.3 Learning Temporal Representations using a Residual Learning Approach

After the self-supervised learning, all the long-term and random missing values are imputed. Then, `Self-Recover` learns the individual temporal representations (h_t^H and h_t^M) of the historical predictors x_t^H and the final model-based forecasts $x_t^{M_f}$ using stacked bi-directional LSTM networks followed by fully connected networks. The next step is to combine the two learned feature representations (h_t^H and h_t^M) into a joint representation that can be used to predict the GEV distribution parameters. The simplest way to do this would be to concatenate them together into a single representation, but this would increase the number of DNN parameters to be estimated from the data. Furthermore, the DNN can be susceptible to the vanishing gradient problem [88].

Alternatively, we may consider the learned representation from the historical predictors (h_t^H) as the basis for a residual learning approach. The learned representation from model-based forecasts (h_t^M) can then be augmented to refine and correct the representation error in h_t^H:

$$h_t^{(H+M)} = h_t^H + \alpha \times h_t^M, \tag{4.7}$$

where α is a scalar hyperparameter for the residual weight.

In this work, we investigate the effectiveness of both approaches—simple concatenation versus residual learning—and compare their relative performance in terms of their effectiveness (forecast accuracy) and efficiency (convergence rate). As will be shown in our experiments, although the accuracy of both approaches is quite comparable, `Self-Recover` achieves faster convergence when using the residual approach compared to simple concatenation.

4.3.4 Incorporating GEV Distribution for Block Maxima Prediction

The combined representation $h_t^{(H+M)}$ in (4.7) is used as input to a fully connected network to estimate the GEV parameters, μ, σ, and ξ. Similar to `DeepExtrema`, `Self-Recover` must learn the GEV parameters in a way that preserves their inter-dependent constraints in (3.5). To do this, the hard GEV constraints are reformulated as soft constraints as follows:

$$\forall\, i : 1 + \frac{\xi}{\sigma}(y_i - \mu) + \tau \geq 0. \tag{4.8}$$

The slack variable τ accommodates for minor violations of the second constraint in (3.5) as long as $1 + \frac{\xi}{\sigma}(y_i - \mu) > -\tau$ for all time windows i. The reformulation allows us to reparameterize the constraint as learning an upper (ξ_u) and lower bound (ξ_l) on ξ:

$$\frac{\sigma}{\mu - y_{\min}} (1 + \tau) \geq \xi \geq -\frac{\sigma}{y_{\max} - \mu} (1 + \tau) \tag{4.9}$$

The fully connected network will generate ξ_u and ξ_l as its output instead of just a single ξ (along with μ and σ). A regularization penalty involving $(\xi_u - \xi_l)^2$ is also introduced into the loss function to ensure the estimated parameters $\xi_u \approx \xi_l \approx \xi$. The estimated GEV parameters are subsequently provided to the Model Bias Offset module to debias them in order to deal with issues due to the random initialization of the deep neural networks. More details on the Model Bias Offset can be found in [81].

Finally, the debiased GEV parameters are passed to a fully connected layer to predict the block maxima ($\hat{y}_t$) for the time window. The debiased GEV parameters are also used to compute its upper and lower quantiles, $\hat{y}_U$ and $\hat{y}_L$, respectively, using the GEV quantile formula given in (3.3).

4.3.5 Optimization of Block Maxima Prediction

To learn the network parameters, `Self-Recover` integrates the regularized negative log-likelihood loss with the mean-square error function. This combination of loss functions allows the framework to find a GEV distribution that best fits the data while making an accurate point estimation of the block maxima. The combined loss function is given by:

$$\begin{aligned} \mathcal{L}_{\text{Block-maxima}} = \lambda_1 \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \\ (1 - \lambda_1)\left\{\lambda_2 \, \ell_{GEV}(\mu, \sigma, \xi) + (1 - \lambda_2) \sum_{i=1}^{n} (\xi_{u,i} - \xi_{l,i})^2)\right\} \end{aligned} \tag{4.10}$$

$\mathcal{L}_{\text{Block-maxima}}$ combines the negative log-likelihood function of the GEV distribution and the difference between the upper and lower-bound estimates of ξ. Note that λ_1 and λ_2 are hyperparameters that manage the trade-off between the different components of the loss function. Adam [89] optimizer is used while training. The trained network can be used to generate the block

maxima prediction, $\hat{y}$, for any future input, along with its upper and lower bounds, $(\hat{y}_L, \hat{y}_U)$, as well as the GEV parameters $(\hat{\mu}, \hat{\sigma}, \hat{\xi})$.

4.4 Experimental Results

We have performed extensive experiments to evaluate the performance of our `Self-Recover` framework. We consider the following two real-world datasets for our experiments.

Hurricane: We utilize the same HURDAT2 database [78] employed for `DeepExtrema` in Chapter 3. We created non-overlapping time windows of length 16-time steps (i.e., 4 days) from the 3,111 hurricanes. We use the first 8 time steps for the predictor variables and the remaining 8-time steps as the forecast window. We also extracted the corresponding model output forecasts for 396 hurricanes (between 2011 and 2020) from 21 statistical and dynamical models, such as CMC (Canadian Global Model Forecast), SHIP (SHIPS Model Intensity Forecast), etc. from the *Hurricane Forecast Model Output* website at the University of Wisconsin-Milwaukee.[1] After preprocessing, we have 5912 time windows, out of which 768 contain both historical observations and model-based forecasts.

Climate: We consider the daily maximum temperature data from the *North American Regional Climate Change Assessment Program (NARCCAP)* data archive[2]. The historical observations contain daily maximum temperature data of 3 climate stations (Maple City, Hart, and Eau Claire) along the eastern shore of Lake Michigan for 7665 consecutive days (from 1978 to 1998). We then generate non-overlapping time windows of 14 days in length, with the first 7 days serving as the historical time window and the last 7 days serving as the target window for predicting its block maxima. The corresponding model-based forecasts are simulation data from four regional climate models (RCMs), namely, CRCM, WRFG, HRM3, and RCM3. These four RCM simulations are driven by the four GCMs - CCSM, CGCM3, GFDL, and HadCM3. The model-based simulation data consists of 8 predictor variables. For evaluating self-supervised missing values imputation, we completely remove model-based simulation data from 1978 to 1985 and randomly remove 5% data for each time window from 1986 to 1998. After the data preprocessing, there are 3285 time

[1] http://derecho.math.uwm.edu/models

[2] https://www.narccap.ucar.edu/data/index.html

windows in total.

To demonstrate its effectiveness, we compare `Self-Recover` against the following baseline methods: (1) **Persistence:** This approach uses the block maxima from the historical time window as the block maxima for the forecast time window. (2) **MF** (Model-based Forecasts): This approach simply computes the block maxima of the model-based forecasts as its prediction. (3) **FCN**: This approach uses a fully connected network to predict the block maxima given a set of predictors. (4) **LSTM**: This approach employs a bi-directional stacked LSTM network followed by a fully connected network to predict the block maxima. (5) **Transformer**: This approach uses an attention-based transformer network to predict the block maxima. (6) **InceptionTime**[90]: This is a state-of-the-art CNN-based time series classification method. We replace the final softmax layer with a fully connected network for block maxima prediction. (7) **DeepPIPE**[32]: This is an uncertainty quantification-based approach to predict block maxima by using a hybrid loss function. (8) **EVL**[34]: This approach uses an ad-hoc EVT-based loss function to predict the block maxima.

4.4.1 Evaluation Setup

For the evaluation of block maxima prediction, we split the data into separate training, validation, and testing sets with a ratio of 8:1:1. For evaluating self-supervised missing values imputation, we further split the training data of block maxima prediction into separate training, validation, and testing sets with a ratio of 8:1:1. We repeated our experiment 10 times, each time using a different random split. The data is standardized to have zero mean and unit variance. All the methods are trained by varying their DNN hyperparameters as follows: number of layers (2-6), number of nodes (8-128), learning rate (10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}), and batch size (32, 64, 128, 256), while assessing their performance on the validation set. The best hyperparameters are chosen using Ray Tune, a tuning framework with an ASHA scheduler. The training and evaluation are carried out in an NVIDIA Tesla K80 GPU with 12GB RAM.

We evaluate the performance of the proposed self-supervised missing values imputation on the corresponding test set using the following metrics: (1) Root mean squared error (RMSE) and (2) correlation between the imputed and ground truth values. For comparison, we also use a randomized

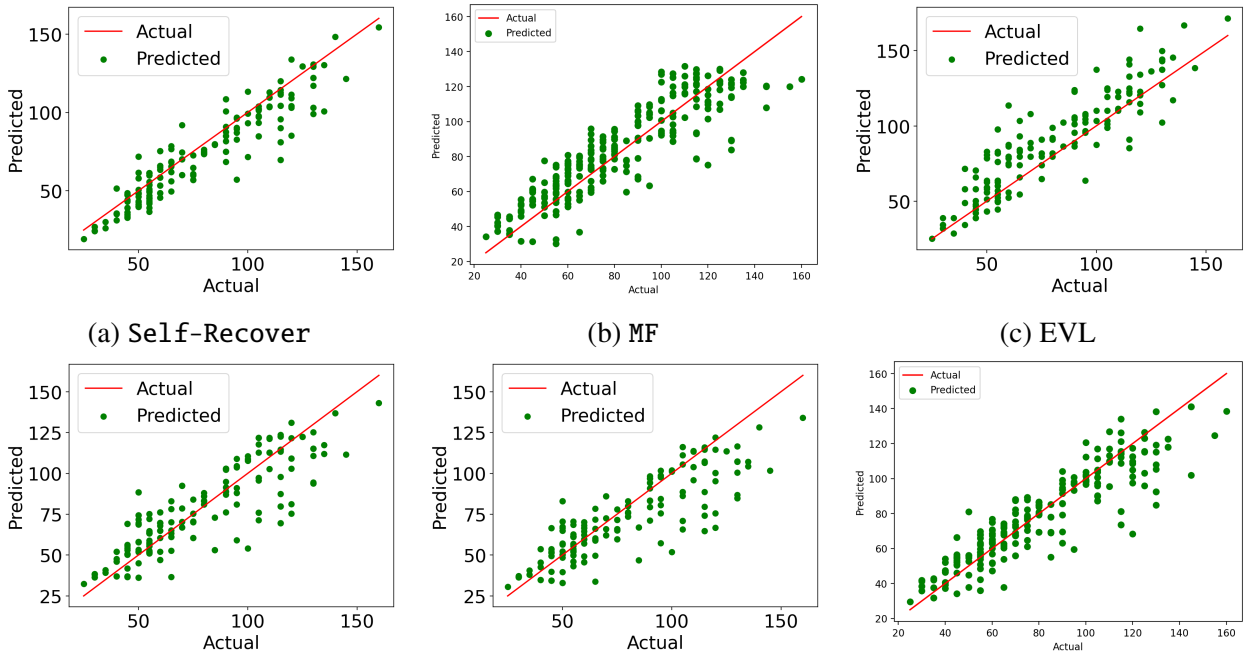
imputation technique based on the mean (μ) and standard deviation (σ) of the available data. The missing values are imputed by randomly drawing values from $\mathbb{N}(\mu, \sigma)$. For block maxima prediction, we evaluate the performance of the methods on the test set using: (1) RMSE and (2) correlation between the predicted and ground truth block maxima as well as (3) Accuracy and (4) F_1 score for classifying extreme events. For the Hurricane dataset, we define extreme hurricane intensity as values that exceed either (1) 111 mph (i.e., category 3 and above hurricanes) or (2) 130 mph (i.e., category 4 and above hurricanes). For the climate dataset, we define extreme temperature events as values that are either above the 80th or 90th percentiles of the temperature values at a location.

4.4.2 Results

Methods	Hurricane (Intensity)		Climate (Temperature)	
	RMSE	Correlation	RMSE	Correlation
Persistence	28.05	0.57	7.48	0.46
MF	17.12	0.80	3.94	0.58
FCN	16.62 ± 0.27	0.84 ± 0.04	3.81 ± 0.38	0.59 ± 0.06
LSTM	16.11 ± 0.24	0.85 ± 0.04	3.57 ± 0.34	0.62 ± 0.04
Transformer	15.31 ± 0.23	0.87 ± 0.04	3.10 ± 0.24	0.67 ± 0.03
Inception	15.51 ± 0.28	0.86 ± 0.03	3.07 ± 0.28	0.66 ± 0.04
DeepPIPE	17.02 ± 0.33	0.81 ± 0.04	3.76 ± 0.21	0.61 ± 0.05
EVL	15.61 ± 0.26	0.85 ± 0.04	3.28 ± 0.31	0.63 ± 0.05
`DeepExtrema`	15.86 ± 0.29	0.85 ± 0.03	3.49 ± 0.24	0.64 ± 0.05
`Self-Recover`	**14.88 ± 0.22**	**0.90 ± 0.03**	**2.79 ± 0.26**	**0.71 ± 0.04**

Table 4.1 Overall performance comparison in terms of RMSE and correlation of block maxima prediction.

Table 4.1 compares the forecasting performance of the various methods in terms of their RMSE and correlation. The results show that the proposed `Self-Recover` framework outperforms all the baselines on both Hurricane and Climate datasets. To verify its effectiveness in terms of modeling extreme values, Figure 4.5 shows scatter plots of the actual versus predicted block maxima values for `Self-Recover` and other baseline methods (MF, EVL, Transformer, DeepPIPE, and Inception) on the Hurricane intensity data. The plots indicate that `Self-Recover` can accurately estimate the block maxima values of hurricane intensity including the upper and lower extreme values. In contrast, the block maxima predictions generated by other competing baselines have larger biases.



(a) `Self-Recover` (b) `MF` (c) EVL

(d) Transformer (e) DeepPIPE (f) Inception

Figure 4.5 Comparison of actual versus predicted block maxima for various methods on the Hurricane dataset.

Methods	Accuracy (F-1 Score)			
	Hurricane (Intensity)		Climate (Temperature)	
	Category 4 and above	Category 5 and above	About 90 Percentile	About 80 Percentile
MF	0.81 (0.80)	0.83 (0.84)	0.77 (0.75)	0.79 (0.80)
FCN	0.75 (0.78)	0.80 (0.82)	0.78 (0.79)	0.79 (0.77)
LSTM	0.81 (0.82)	0.82 (0.83)	0.79 (0.78)	0.80 (0.81)
Transformer	0.84 (0.85)	0.87 (0.88)	0.82 (0.83)	0.84 (0.82)
Inception	0.83 (0.82)	0.85 (0.86)	0.80 (0.81)	0.82 (0.84)
DeepPIPE	0.77 (0.80)	0.81 (0.82)	0.78 (0.80)	0.79 (0.78)
EVL	0.85 (0.79)	0.86 (0.81)	0.80 (0.77)	0.83 (0.81)
`DeepExtrema`	0.84 (0.85)	0.86 (0.87)	0.82 (0.82)	0.83 (0.85)
`Self-Recover`	**0.89 (0.90)**	**0.93 (0.92)**	**0.86 (0.88)**	**0.89 (0.91)**

Table 4.2 Performance comparison in terms of accuracy and F_1 score for predicting extreme events only.

For example, EVL tends to overestimate the block maxima values whereas Transformer, Inception, and DeepPIPE tend to underestimate the values, particularly at the upper extremes, which is not surprising since the methods do not incorporate EVT into their modeling approach.

Finally, Table 4.2 compares the accuracy of each method in terms of classifying extremely high

Methods	Hurricane		Climate	
	RMSE	Correlation	RMSE	Correlation
Randomized Imputation	5.21 ± 0.31	0.70 ± 0.05	2.81 ± 0.28	0.73 ± 0.05
Self-supervised Imputation	4.51 ± 0.36	0.78 ± 0.04	2.26 ± 0.24	0.81 ± 0.03

Table 4.3 Comparing randomized missing values imputation via Gaussian Distribution against the proposed self-supervised missing values imputation in `Self-Recover`.

Methods	Hurricane		Climate	
	RMSE	Runtime (s)	RMSE	Runtime (s)
Concatenation	14.95 ± 0.23	1503.4	2.86 ± 0.24	1012.7
Residual	14.88 ± 0.22	1215.9	2.79 ± 0.26	891.6

Table 4.4 Comparing concatenation against residual learning approaches for merging representations in `Self-Recover`.

hurricane intensity and temperature events. Observe that `Self-Recover` outperforms all other baselines in terms of accuracy and F_1 score.

Effect of Self-supervised Missing Values Imputation: Table 4.3 compares the missing values imputation performance of `Self-Recover` when using randomized imputation via Gaussian distribution against the proposed self-supervised imputation approach. The results show that our self-supervised imputation approach outperforms random imputation in terms of RMSE and correlation.

Effect of Residual Learning: Table 4.4 compares the performance of `Self-Recover` when using simple concatenation to combine its learned representations against the residual learning approach. While their accuracies are quite similar, the training runtime for the simple concatenation approach is significantly higher due to its slower convergence.

4.4.3 Ablation Studies

We examine the effects of varying two hyperparameters of our algorithm, α, and λ_1. The hyperparameter α determines the residual weight of the model-based representation when combined with the historical representation. As shown in Figure 4.6 (left), the RMSE generally increases with increasing value of α. For both data sets, a smaller value of α results in higher performance, which validates our strategy for incorporating the model-based representation as a “residual" term to enhance the representation of historical observations.

The trade-off between minimizing point prediction (RMSE loss) and preserving the GEV

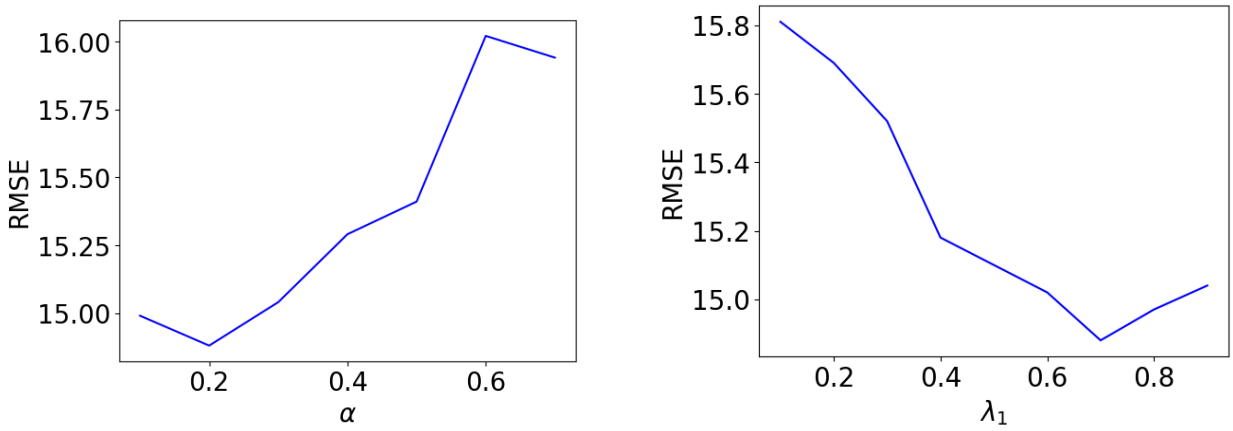


Figure 4.6 Examining the effect of α (left) and λ_1 (right) on RMSE of the block maxima prediction for Hurricane dataset with `Self-Recover`.

distribution (GEV loss) of `Self-Recover` is determined by the hyperparameter λ_1 in Equation 4.10. A smaller λ_1 places greater emphasis on GEV loss, which focuses more on extreme values. This can be seen from the results shown in Figure 4.6 (right), where the RMSE of the block maxima prediction increases when λ_1 decreases. This suggests that the combined loss is useful rather than using only the RMSE loss, thus validating the need to incorporate GEV distribution into the `Self-Recover` framework.

4.5 Conclusion

This work introduces `Self-Recover`, a novel deep learning framework for predicting the block maxima of time series by handling disparate temporal coverage of predictors. A combination of contrastive and generative self-supervised learning schemes followed by a DAE is proposed to impute long-term and random missing values of model-based forecasts. To learn and merge the representations from historical and model-based forecasts, we provide a residual technique and show that it is more efficient than the straightforward concatenation method. Our experimental results on real-world data demonstrate the superiority of `Self-Recover` compared to other state-of-the-art methods.

CHAPTER 5

SIMEXT: SELF-SUPERVISED REPRESENTATION LEARNING FOR EXTREME VALUES IN TIME SERIES

5.1 Introduction

Deep time series forecasting models are widely used to predict the future outcomes of complex processes that evolve over time as discussed in chapter 2. The accuracy and effectiveness of these models often depend on their ability to discern the underlying patterns of the data and utilize them to make inferences about the future evolution of the time series. One crucial aspect to consider in time series forecasting is the ability to predict extreme values, i.e., values that fall far outside the typical range of observations. Forecasting of extreme values is important in many application domains as extreme events may signify dire scenarios such as natural disasters, financial collapse, or public health crises.

Block maxima or minima [10] are commonly used to define extreme values in a time series. Prediction of block maxima or minima is of paramount importance as it enables us to anticipate the worst-case scenario within the forecast period. For instance, predicting the maximum intensity of an upcoming hurricane or amplitude of seismic activity for a future time window can assist emergency planners in assessing its potential damages.

Accurate forecasting of extreme values in time series is challenging for several reasons. First, the modeling of extreme values requires a specific focus on capturing the tail distribution [10], which is a major departure from conventional techniques that are often biased toward modeling the conditional mean. The problem is further exacerbated by the fact that the extreme values are rare, making them hard to predict even with a large amount of historical data available. Finally, the extreme values could be associated with certain peculiarities in the time series, such as abrupt changes, volatility clustering, persistent dependencies, etc [91, 92]. Advanced representation learning approach is therefore needed to learn the underlying patterns in the time series that can be utilized for extreme value forecasting.

Self-supervised learning (SSL) [85, 84] is an emerging technique in machine learning to learn

useful feature representation from available data. One of the main advantages of self-supervised learning is that it can facilitate robust representation learning despite limited data. For example, self-supervised contrastive learning [85] is a specific approach that uses data augmentation to overcome the labeled data scarcity problem. It is based on the idea of comparing different augmented versions of the same input and learning representations that are robust to changes introduced by the augmentations. For time series, it can be used to learn representations that are invariant to time shifts, scaling, or warping [93], thereby improving the generalizability of the model to new data. Another advantage of using SSL is that it can help capture the complex patterns associated with the non-linear dependencies of the time series. For example, reconstruction-based self-supervised models such as auto-encoders can effectively capture non-linear relationships and subtle variations in the data.

Though existing SSL approaches have shown promise in extracting meaningful representations from time series [85, 84], they are mostly focused on capturing the common patterns in the time series instead of its extreme values. Worse still, the perturbation introduced by existing data augmentation techniques may alter the extreme values of the time series, leading to a distorted representation of its tail distribution. Preserving the fidelity of the tail behavior during augmentation is crucial to ensure that the learned representations are robust to different scenarios yet able to capture the characteristics of extreme events. Therefore, it is important to develop data augmentation techniques that are aware of the extreme values when transforming the time series data for self-supervised learning.

To address this problem, we propose an SSL framework called `SimEXT` to learn a useful representation that captures extreme values in time series. Our framework leverages the contrastive learning approach along with a reconstruction-based autoencoder architecture for time series modeling. A novel wavelet-based data augmentation technique is introduced to maintain the distribution of extreme values after data augmentation. Theoretical proofs are also given on the probabilistic guarantees provided by our wavelet-based augmentation to ensure that the block maxima (or minima) of the original time series are preserved by the augmented data. `SimEXT` uses the reconstruction loss between the original and reconstructed sample to promote robust representation

learning. In addition, we investigate two distribution loss functions to ensure the fidelity of the tail distribution is preserved—one based on the weighted distance of their empirical cumulative distribution functions (CDFs) while the other corresponds to the Cramér–von Mises distance of fitted Generalized Extreme Value (GEV) distributions. The learned representations are then applied to downstream tasks that focus on predicting future block maxima (or minima) of a time series.

5.2 Related Works

Deep learning techniques such as LSTM networks [1, 3], convolutional neural networks (CNNs) [18, 17], attention models [22], and their variants have found success in various time series applications. Prior works on modeling extreme values with deep learning have explicitly incorporated either the Generalized Pareto (GP) distribution for modeling excess values over a threshold or the Generalized Extreme Value (GEV) distribution for modeling the block maxima (or minima) of a time series. For instance, [34] [37, 38] combined deep learning with the GP distribution for predictive modeling of spatiotemporal data using the maximum log-likelihood approach. [36] combined copula theory and normalizing flows with the GP distribution for generating multivariate extremes. [81] integrated the GEV distribution with deep learning formulation to predict block maxima in univariate time series.

Self-supervised learning (SSL) has become increasingly popular in recent years due to its excellent performance on various benchmark datasets along with strong theoretical foundations [39, 40, 41]. In particular, it has been shown to outperform supervised learning methods in various instances [39, 42]. SSL for time series has been shown to be an effective method for representation learning and feature extraction, benefitting various downstream tasks such as forecasting, classification, and anomaly detection. One of the most popular SSL-based methods for time series is autoencoding, which aims to learn a compressed representation of the input data by reconstructing it from its latent features. For instance, [48] proposed an SSL framework that uses recurrent autoencoder ensembles to learn a compressed representation of the input data and generate future predictions. The proposed method achieved competitive performance compared to traditional methods.

Contrastive learning is another well-known SSL method, which aims to learn a representation

that maximizes the similarity between positive examples and minimizes the similarity between negative examples. For example, [49] proposed a Contrastive Predictive Coding (CPC) framework that employs a contrastive objective to learn a representation that captures temporal dependencies in audio signals. Similarly, [50] proposed a Contrastive Multiview Coding (CMC) framework for time series that uses a contrastive objective to learn a representation capturing different perspectives of the input data. [52] applied intra-level contrastive learning to disentangle seasonal and trend representations from time series. [53] incorporated frequency information into contrastive learning while maintaining consistency between the time and frequency domains. [54] combined temporal and contextual contrasting for representation learning while [55] proposed augmentation techniques that not only generate phase shifts and amplitude changes but also retain the structure and feature information of the time series.

Another class of SSL methods for time series is based on Generative Adversarial Networks (GANs), which can learn a generator model that produces realistic samples from the input data distribution. For example, [56] proposed an SSL framework for time series forecasting, which uses GANs to generate future predictions from the input data. Similarly, [57] proposed an SSL framework for time series anomaly detection, which uses GANs to generate normal samples from the input data distribution and detect anomalies based on the reconstruction error. Despite the extensive literature, to the best of our knowledge, none of the existing studies have focused on developing SSL for forecasting extreme values.

5.3 Preliminaries

5.3.1 Problem Statement

Consider a time series of T time steps, $y_1, y_2, \cdots, y_T$, where $y_i \in \mathbb{R}$. Assume the time series is divided into a set of non-overlapping time windows, where each window w_t corresponds to the interval $[t-\alpha, t+\beta]$ and contains a segment $(y_{t-\alpha}, \cdots, y_t, \cdots, y_{t+\beta})$ of the input time series. For each window w_t, we define its *predictor window* as the interval $[t-\alpha, t]$ and its *forecast window* as the interval $[t+1, t+\beta]$.

Definition 1. *Given a time window w_t, with its corresponding forecast window, $[t+1, t+\beta]$, the*

block maxima (or minima) of the time series for the forecast window is defined as follows:

$$m_t = \max_{\tau \in [t+1, t+\beta]} y_t \quad or \quad \mu_t = \min_{\tau \in [t+1, t+\beta]} y_t$$

Note that the block minima of a forecast window is equivalent to the block maxima of its flipped segment, $(-y_{t+1}, -y_{t+2}, \cdots, -y_{t+\beta})$. For brevity, our discussion in the remainder of this chapter focuses on block maxima only, though the approach is equally applicable to block minima.

Our first goal is to learn a robust feature representation of the time series in each predictor window, i.e., $\boldsymbol{z}_t = h_\theta(\boldsymbol{x}_t) \in \mathbb{R}^d$, where $\boldsymbol{x}_t = (y_{t-\alpha}, \cdots, y_t) \in \mathbb{R}^{\alpha+1}$ is the time series segment associated with the predictor window, $h_\theta(\cdot)$ is an encoder model that maps the input time series segment into its latent representation, and θ is the learnable parameters. Our second goal is to predict the block maxima, $\hat{m}_t$, of the forecast window by learning a mapping function $f_\phi(\cdot)$ such that $\hat{m}_t = f_\phi(\boldsymbol{x}_t, \boldsymbol{z}_t)$, where ϕ is the learnable parameters.

5.3.2 Contrastive Learning

Contrastive learning is an SSL approach that aims to learn the feature representation of data in such a way that similar instances are close to each other in the latent representation space while dissimilar instances are far apart. To apply contrastive learning, we first create two augmented views, denoted as $\boldsymbol{x}_t^1$ and $\boldsymbol{x}_t^2$, respectively, of each input sample $\boldsymbol{x}_t$. These augmented views are often generated by applying random transformation or augmentation to $\boldsymbol{x}_t$. The choice of augmentation depends on the specific application and is typically designed to encourage robustness and invariance to various input perturbations. In contrastive learning, the terms "positive" or "negative" examples refer to the relationships between pairs of augmented views from the same input (positive) or augmented views of different inputs (negative). Given an augmented view $\boldsymbol{x}_t^1$ from the input $\boldsymbol{x}_t$, the corresponding positive example is denoted as $\boldsymbol{x}_t^2$, representing another augmented view of the same input $\boldsymbol{x}_t$. Conversely, negative examples are augmented views $\boldsymbol{x}_t^1$ and $\boldsymbol{x}_{t'}^2$, where $\boldsymbol{x}_t^1$ is an augmented view of input $\boldsymbol{x}_t$ while $\boldsymbol{x}_{t'}^2$ represents an augmented view of another input sample $\boldsymbol{x}_{t'}$.

Given a random minibatch of N samples, $\{\boldsymbol{x}_{t_1}, \boldsymbol{x}_{t_2}, \cdots, \boldsymbol{x}_{t_N}\}$, let $\{\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^{2N}\}$ be the set of derived pairs of augmented samples, where each $(\boldsymbol{x}^{2k-1}, \boldsymbol{x}^{2k})$ denotes a pair of augmented views generated from the sample $\boldsymbol{x}_{t_k}$. Let $\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{|\boldsymbol{u}| \cdot |\boldsymbol{v}|}$ represent the cosine similarity between the

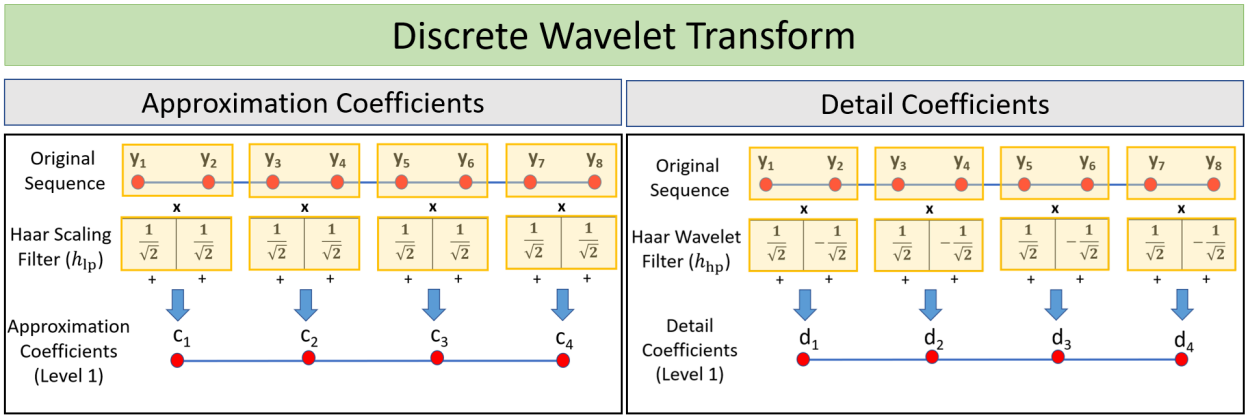


Figure 5.1 The Discrete Wavelet Transform (DWT) employing the Cascading Filter Banks Algorithm with Haar filters, as denoted in Eq. 5.5. It is noteworthy that with the original sequence containing 8 data points, the number of approximation and detail coefficients are halved to 4 at Level 1.

vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. The objective of contrastive learning is to minimize the following normalized temperature-scaled cross-entropy loss function, also known as the NT-Xent loss [87]:

$$\ell(i,j) = -\log \frac{\exp\left[sim\left(h(\boldsymbol{x}^i), h(\boldsymbol{x}^j)\right)/\tau\right]}{\sum_{k=1}^{2N} \mathbb{1}_{k\neq i} \exp\left[sim\left(h(\boldsymbol{x}^i), h(\boldsymbol{x}^k)\right)/\tau\right]} \tag{5.1}$$

where $\mathbb{1}_{k\neq i}$ is an indicator function whose value is equal to 1 if $k \neq i$, $h(\cdot)$ denotes the representation learning function, and τ is the temperature hyperparameter. The final loss is computed across all positive pairs within the minibatch as follows [87]:

$$\mathcal{L}_{\text{contrastive}} = \frac{1}{2N} \sum_{k-1}^{N} \left[\ell(2k-1, 2k) + \ell(2k, 2k-1)\right] \tag{5.2}$$

Minimizing the contrastive loss encourages the learned representation to maximize the similarity between positive pairs of instances and minimize the similarity between negative pairs.

5.3.3 Discrete Wavelet Transform

The wavelet transform is a powerful tool for analyzing non-stationary time-series data with both high-frequency and low-frequency components. The discrete wavelet transform (DWT) decomposes a signal into a set of approximation (scaling) and detail (wavelet) coefficients. Through decomposition into detail and approximation coefficients, DWT enables us to perform a multi-resolution analysis of the input signal. The approximation coefficients represent low-frequency components, conveying information about the overall trend or shape of the signal, while the detail coefficients capture high-frequency components, revealing the noisy temporal variations of the signal. An input signal

$\boldsymbol{x}_t$ can be expressed as linear combination of the scaling functions $\phi(t)$ and wavelet functions $\psi(t)$ using the detail and approximation coefficients as follows:

$$\boldsymbol{x}_t = \sum_k c_j(k)\phi_{j,k}(t) + \sum_k \sum_j d_j(k)\psi_{j,k}(t) \tag{5.3}$$

where j and k are the scale and dilation parameters respectively, $c_j(k)$ denotes the approximation (scaling) coefficient, and $d_j(k)$ denotes the detail (wavelet) coefficient. There are several popular wavelet families commonly used for DWT through their distinct scaling and wavelet functions. These wavelet families include Haar, Biorthogonal, Daubechies, etc. Using orthogonal scaling and wavelet functions, the coefficients $c_j(k)$ and $d_j(k)$ can be calculated by taking the inner product with the original signal $\boldsymbol{x}_t$ as follows:

$$c_j(k) = \langle \boldsymbol{x}_t, \phi_{j,k}(t) \rangle, \quad d_j(k) = \langle \boldsymbol{x}_t, \psi_{j,k}(t) \rangle \tag{5.4}$$

In most practical applications, it is important to note that the scaling and wavelet functions are not explicitly used to calculate the approximation and detail coefficients. Instead, the coefficients are obtained through the cascading filter banks algorithm, which allows for an efficient implementation of the DWT by employing a series of low-pass and high-pass filters. Specifically, the high-pass and low-pass filters can be applied in a cascading manner after signal extension to recursively compute the coefficients $c_j(k)$ and $d_j(k)$ according to the following equations:

$$\begin{aligned} c_j(k) &= \sum_m h_{\text{lp}}(m - 2k)\ c_{j+1}(m), \\ d_j(k) &= \sum_m h_{\text{hp}}(m - 2k)\ d_{j+1}(m) \end{aligned} \tag{5.5}$$

where $m = 2k + n$ while $h_{\text{lp}}(\cdot)$ and $h_{\text{hp}}(\cdot)$ denote the low-pass and high-pass filters, respectively. Figure 5.1 depicts an example of how the cascading filter banks algorithm has been used to decompose a signal using Haar filters. The Haar scaling and wavelet filters are applied separately, followed by downsampling by a factor of 2 to obtain the detail and approximation coefficients.

The selection of an appropriate wavelet function depends on the specific characteristics of the signal being analyzed and the application objectives. For instance, the Haar wavelet is often

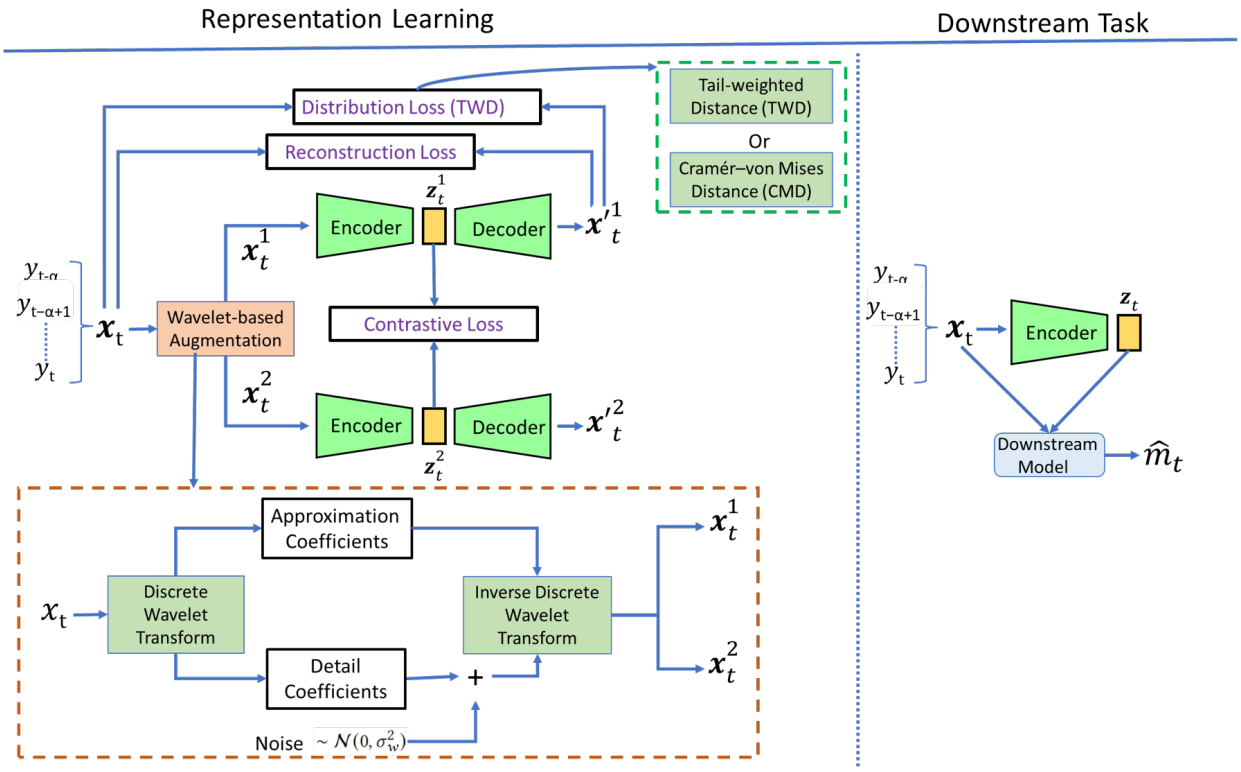


Figure 5.2 An overview of the proposed SimEXT framework for time series forecasting of extreme values.

employed in scenarios where rapid computation is crucial. When using the Haar wavelet, the number of approximation and detail coefficients are halved at each level of decomposition, leading to a more compact representation of the signal. This reduction in coefficients can be advantageous when dealing with large-scale datasets or real-time applications where computational efficiency is paramount.

5.4 Methodology

SimEXT is an SSL framework for representation learning of time series with an emphasis on preserving the fidelity of its tail distribution. An overview of the proposed framework is shown in Fig. 5.2. The framework combines a contrastive learning approach with a reconstruction-based autoencoder to generate a robust latent representation of its input time series. For contrastive learning, SimEXT uses a novel wavelet-based data augmentation technique to ensure that the augmented time series would maintain the block maxima (or minima) value of its original time series. Details of the data augmentation approach are given in Section 5.4.1. It also employs a distribution-based loss

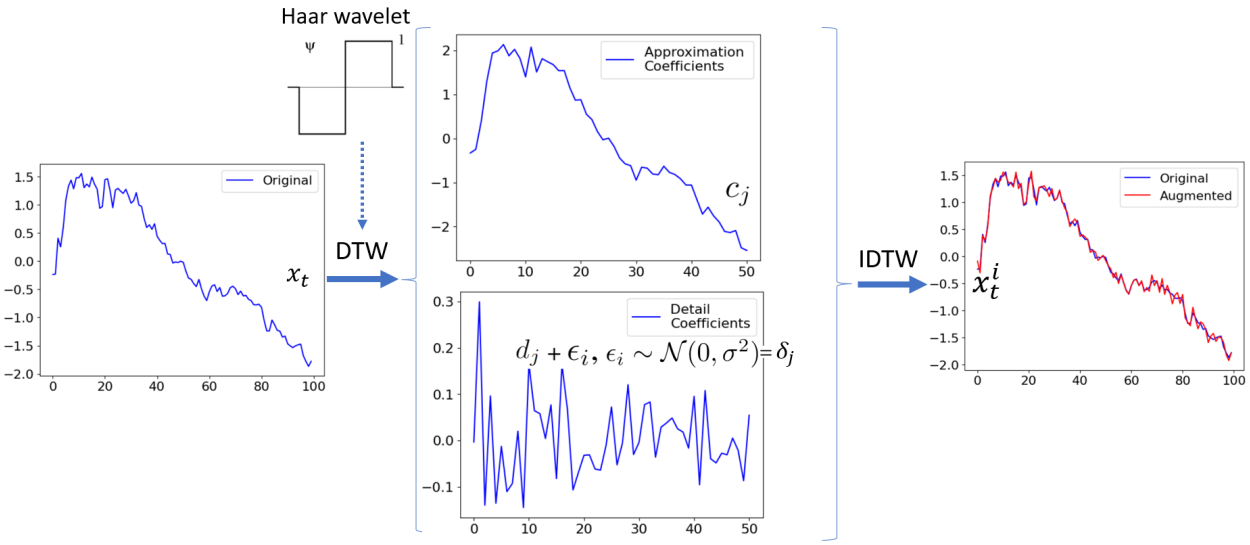


Figure 5.3 An overview of the wavelet-based data augmentation.

function to ensure that the learned representation better captures patterns characterizing the extreme values of the time series. The learned representation can be utilized by any downstream forecasting models for block maxima prediction, as shown in the right part of Fig. 5.2.

5.4.1 Wavelet-based Data Augmentation

As mentioned in Section 5.3.2, contrastive learning requires performing data augmentation to generate conceptually similar samples by perturbing the original data. In principle, there are many ways to perform data augmentation for time series, which include jittering, flipping, shuffling, time warping, etc [94]. Despite the wide variety of approaches available, choosing the right data augmentation approach is crucial for block maxima forecasting to ensure that the learned representation preserves the overall pattern of the time series without significantly altering its block maxima. To achieve this, the framework introduces a wavelet-based data augmentation technique.

Before describing the approach, we first examine the effect of jittering on the block maxima of a time series.

Theorem 1. *Let* $y_1, y_2, \cdots, y_n$ *be a sequence, where* $y_i \in \mathbb{R}$ *and* $M_n = \max_i y_i$. *Assuming* $\hat{M}_n = \max_i \hat{y}_i$, *where* $\hat{y}_i = y_i + \epsilon_i$ *and* $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, *it can be shown that:*

$$\mathbb{E}_\epsilon \left[\hat{M}_n - M_n\right] \leq \log n + \frac{\sigma^2}{2} \tag{5.6}$$

Proof. First, observe that $\mathbb{E}_\epsilon\left[\hat{M}_n - M_n\right] = \mathbb{E}_\epsilon\left[\hat{M}_n\right] - M_n$. Since the exponent function is non-negative, the expected value of $\hat{M}_n$ can be expressed as follows:

$$\begin{aligned}\mathbb{E}_\epsilon\left[\hat{M}_n\right] = \mathbb{E}_\epsilon\left[\max_i\left(\epsilon_i + y_i\right)\right] &= \mathbb{E}_\epsilon\left[\max_i\left(\log e^{\epsilon_i + y_i}\right)\right] \\ &\leq \mathbb{E}_\epsilon\left[\log\sum_{i=1}^{n} e^{\epsilon_i + y_i}\right]\end{aligned}$$

The inequality above can be further simplified using Jensen inequality as follows:

$$\mathbb{E}_\epsilon\left[\hat{M}_n\right] \leq \log\mathbb{E}_\epsilon\left[\sum_{i=1}^{n} e^{\epsilon_i} e^{y_i}\right] = \log\left(\sum_{i=1}^{n}\mathbb{E}_\epsilon\left[e^{\epsilon_i}\right] e^{y_i}\right)$$

Assuming $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, it can be shown that

$$\mathbb{E}\left[e^{\epsilon}\right] = \int_{-\infty}^{\infty} e^{\epsilon}\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\epsilon^2}{2\sigma^2}} d\epsilon = e^{\frac{\sigma^2}{2}}$$

Replacing the expected value into the inequality given in 5.7, we obtain the following:

$$\begin{aligned}\mathbf{E}_\epsilon\left[\hat{M}_n\right] \leq \log\left(\sum_{i=1}^{n} e^{\frac{\sigma^2}{2} + y_i}\right) &\leq \log\left(\sum_{i=1}^{n} e^{\frac{\sigma^2}{2} + M_n}\right) \\ &= \log n + \frac{\sigma^2}{2} + M_n\end{aligned}$$

The proof follows by subtracting M_n from the expected value given above. □

Theorem 1 provides an upper bound on the expected value of the difference between the perturbed block maxima and the original block maxima of a time series of length n. Note that the bound is proportional to the variance of the noise as well as the number of perturbed data points. Thus, by using a smaller variance during the jittering process, fewer perturbed data points, or both, this can help avoid altering the block maxima value significantly.

Algorithm 5.1 summarizes the pseudocode of our proposed wavelet-based data augmentation approach. Given an input segment, $\boldsymbol{x}_t$, we first employ DWT to decompose the time series into its approximation coefficients ($c_{j_0,k}$) and wavelet (detail) coefficients ($d_{j,k}$). We apply the Daubechies 1 wavelet, also known as the Haar wavelet, in this study but the methodology is applicable to other wavelet functions. As mentioned in Section 5.3.3 and shown in Figure 5.1, by employing the Haar

Algorithm 5.1 Wavelet-based Data Augmentation Algorithm

Input: Time series predictor $\boldsymbol{x_t}$ and noise variance, σ^2.
Output: Augmented time series pair, $\boldsymbol{x}_t^1$ and $\boldsymbol{x}_t^2$.

$(c_{j_0,k}, d_{j,k}) \leftarrow \mathbf{DWT}(\boldsymbol{x_t})$
for i = 1 to 2 **do**
  $\delta^i_{j,k} \leftarrow d_{j,k} + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$
  $\boldsymbol{x}_t^i \leftarrow \mathbf{IDWT}(c_{j_0,k}, \delta^i_{j,k})$
end for
return $\boldsymbol{x}_t^1$ and $\boldsymbol{x}_t^2$

wavelet, the number of detail and approximation coefficients will be halved of the length of the original time series. Since the approximation coefficient captures the general trend of the input signal while the detail coefficient captures the noisy, high-frequency component, we perform augmentation by applying jittering to the detail coefficients only. Specifically, each detail coefficient is perturbed by adding a Gaussian noise with variance σ^2. As the number of detail coefficients is half of the length of $\boldsymbol{x}_t$ and using a low variance σ^2, following Theorem 1, this ensures that the block maxima of the augmented time series are close to that of its input time series. Finally, we employ the inverse discrete wavelet transform (IDWT) to reconstruct the corresponding augmented time series using the approximation coefficients as well as the perturbed wavelet (detail) coefficients.

In summary, the proposed wavelet-based data augmentation perturbs the detail coefficients only, without altering the approximation coefficients. This ensures that the overall temporal pattern and block maxima distribution are preserved, as only the high-frequency components and essentially half of the data points in the signal are perturbed.

5.4.2 Self-supervised Representation Learning

The wavelet-based data augmentation technique described in the previous section is used to create augmented pairs of similar samples for each predictor window $\boldsymbol{x}_t$. Each augmented pair $(\boldsymbol{x}_t^1, \boldsymbol{x}_t^2)$ is passed to an autoencoder to generate its corresponding feature embedding $(\boldsymbol{z}_t^1, \boldsymbol{z}_t^2)$, capable of reconstructing the original sample:

$$\boldsymbol{x}_t'^i = Decoder(\boldsymbol{z}_t^i) \text{ where } \boldsymbol{z}_t^i = Encoder(\boldsymbol{x}_t^i) \tag{5.7}$$

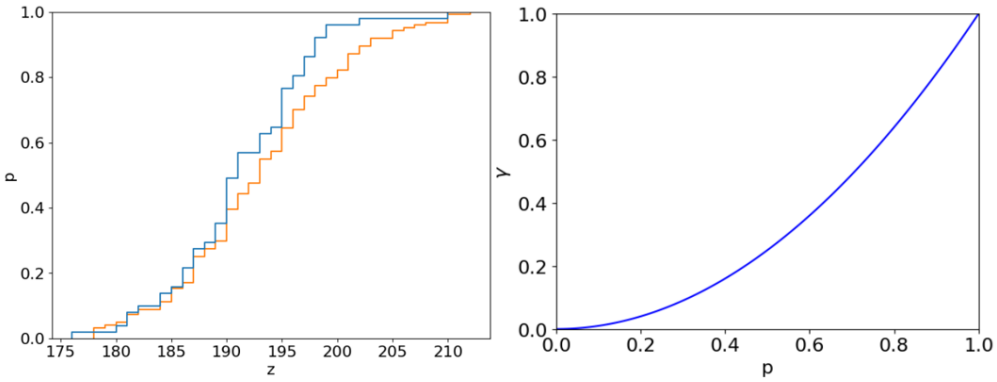


Figure 5.4 Tail-Weighted Distance (TWD) between Empirical Cumulative Distribution Functions (ECDFs).

A reconstruction loss based on the squared Euclidean norm between the original and reconstructed samples after data augmentation is used to ensure that the learned representation preserve the important features of the original sample. The contrastive loss given by Eq. 5.2 is also calculated for all positive samples (z_t^1, z_t^2) to ensure that the learned representations of similar augmented pairs are close to each other.

In the context of the proposed framework, the combination of contrastive learning and reconstruction-based autoencoder is used to facilitate robust representation learning. By jointly minimizing the reconstruction loss and contrastive loss, the framework is able to learn representations that are robust to minor perturbations of the original data.

5.4.3 Enforcing Fidelity of Tail Distribution

The reconstruction and contrastive losses described in the previous section are insufficient to ensure that the learned representation would preserve the fidelity of the block maxima (or minima) distribution. To overcome this limitation, we introduce a distribution loss into the objective function to emphasize learning features that consider extreme values of the time series. Let $\boldsymbol{x}_t$ be the original input time series and $\boldsymbol{x}'_t$ be the reconstructed time series. Their corresponding empirical cumulation distribution functions (ECDFs) are defined as

$$F_x(z) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[x_{(i)} \leq z], \; F_{x'}(z) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[x'_{(i)} \leq z] \tag{5.8}$$

where n is the sample size, $x_{(i)}$ is the i-th largest observation in $\boldsymbol{x}$, and $\mathbb{1}[x_{(i)} \leq z]$ is an indicator function, whose value is 1 if $x_{(i)} \leq z$ and 0 otherwise. We consider two approaches for measuring

the distribution loss.

5.4.3.1 Tail-Weighted Distance (TWD) between Empirical Cumulative Distribution Functions (ECDFs)

Figure 5.4 shows an overview of the proposed TWD approach. The loss is motivated by the Kolmogorov–Smirnov (KS) distance [95] for determining whether a sample is drawn from a particular distribution: $D_{KS}(F_{x'}, F_x) = \sup_z |F_{x'}(z) - F_x(z)|$. Specifically, the KS distance between two ECDFs is given by the largest absolute deviation at any given position. However, the distance does not consider whether the extreme values of the distribution are well-preserved and have high variance unless n is sufficiently large. Instead, we introduce the following tail-weighted distance to put more emphasis on the extreme values:

$$\text{TWD} = \frac{1}{n} \sum_{j=1}^{n} \gamma(p_j) \cdot \ |F_{x'}(x_j) - F_x(x_j)| \tag{5.9}$$

where F_x and $F_{x'}$ are the ECDFs of the original and reconstructed time series respectively, $p_j = F_x[x_j]$ is the percentile of the j-th observation and $\gamma(p_j) = p_j^2$ is a tail-weighted function of the percentile p_j, whose value grows quadratically with increasing p_j. This would bias the TWD to put more emphasis on the right tail of the distribution. Alternatively, to put a large emphasis on both upper and lower tails, the weight function could be defined as $\gamma(p_j) = 2(p_j - 0.5)^2$.

5.4.3.2 Cramér–von Mises Distance (CMD) between the ECDFs of GEV Distribution for Block Maxima

In this approach, we employ the Cramér-von Mises (CM) distance [96, 97] to measure the deviation between the GEV distribution of the block maxima values in the original and reconstructed time series. Let $m = \max_{x_i \in \boldsymbol{x}} x_i$ be the block maxima of the original series and $m' = \max_{x_i \in \boldsymbol{x'}} x_i$ be the block maxima of the reconstructed series. Given a set of block maxima values generated from the predictor windows of the training data, we use the maximum likelihood approach, given in Eq. 3.4 to estimate GEV parameters of the block maxima values. Let G_x and $G_{x'}$ be the ECDF of the fitted GEV distributions of block maxima values associated with the original and reconstructed time series, respectively. We then compute the distribution loss using Cramér–von Mises Distance

[96, 97] between the ECDFs of the two GEV distributions as follows:

$$\text{CMD} = \sqrt{\sum_{i=1}^{n} |G_{x'}(z_i) - G_x(z_i)|} \tag{5.10}$$

where $\{z_1, z_2, \cdots, z_n\}$ are the samples drawn from the fitted GEV distribution of the original time series and n is the number of samples drawn.

5.4.4 Downstream Task

The utilization of learned representations derived from self-supervised learning holds promise in enhancing the performance of downstream extreme value prediction tasks. This application is illustrated in the right part of Figure 5.2. Specifically, an input time series $\boldsymbol{x}_t$ is presented to the trained `SimEXT` model to generate its feature embedding, which can be then combined with feature representation from a downstream model to predict the block maxima value associated with its forecast window. The integration of the `SimEXT` model's feature embedding with the feature representation from a downstream model creates a synergistic relationship, where the combined knowledge and insights contribute to more accurate and robust predictions of block maxima values.

5.4.5 Optimization

The `SimEXT` framework is trained to minimize the following loss function:

$$\mathcal{L} = \lambda_1 \, \mathcal{L}_{contrastive} + \lambda_2 \, \mathcal{L}_{recon} + \lambda_3 \, \mathcal{L}_{dist} \tag{5.11}$$

where $\mathcal{L}_{contrastive}$ is the contrastive loss given in Eq. 5.2, $\mathcal{L}_{recon} = \sum_t \|\boldsymbol{x}_t - \boldsymbol{x}'_t\|_2^2$ is the reconstruction loss of the autoencoder, and $\mathcal{L}_{dist}$ is either the TWD or CMD distribution losses described in Section 5.4.3. Note that λ_1, λ_2, and λ_3 are hyperparameters that manage the trade-off between the different components of the loss function. Adam [89] optimizer is used while training. The trained network can be used to generate latent representation given input time series.

5.5 Experimental Evaluation

We have performed extensive experiments to evaluate the performance of our `SimEXT` framework.

5.5.1 Datasets

We consider the following three datasets for our experiments.

Hurricane We utilize the same HURDAT2 database [78] employed for `DeepExtrema` in Chapter 3. For each hurricane, we created non-overlapping time windows of length 24-time steps (6 days). We use the first 16 time steps (4 days) as the predictor window and the last 8 time steps (2 days) as the forecast window.

Climate We use the same daily maximum temperature data from the *North American Regional Climate Change Assessment Program (NARCCAP)* data archive[1] employed for `DeepExtrema` in Chapter 3. We generate non-overlapping time windows of 14 days in length, with the first 7 days serving as the predictor time window and the last 7 days serving as the target window for predicting its block maxima. After the data preprocessing, there are 3285 time windows in total.

ECL This dataset comprises of hourly electricity consumption of 321 clients obtained from the UCI repository [2]. We partition the time series into non-overlapping time windows of 14-day duration. The initial 7-day period is used as a predictor time window, while the remaining 7-day interval defines the forecast window. After preprocessing, the cumulative number of time windows is equal to 5638.

5.5.2 Baseline Methods

In order to demonstrate the efficacy of our proposed self-supervised representation learning framework, `SimEXT`, we conduct a comparative analysis against the following state-of-the-art methods in time series representation learning:

1. **CoST** [52]: CoST focuses on acquiring disentangled seasonal and trend representations from time series data. It explores both the time domain and frequency domain concurrently by utilizing the discrete Fourier transform. Additionally, CoST incorporates intra-level contrastive learning to enhance the quality of learned representations.

2. **TS2Vec** [51]: TS2Vec adopts a hierarchical contrastive learning strategy, leveraging augmented context views to capture multi-scale contextual information. It introduces the concept of contextual consistency for the selection of positive pairs, thereby improving the discriminative

[1] https://www.narccap.ucar.edu/data/index.html

[2] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

power of learned representations.

3. **TNC** [98]: TNC employs a debiased contrastive objective to learn time series representations. It ensures that, in the encoding space, the distribution of signals within a neighborhood is distinguishable from the distribution of signals that are not neighboring.

4. **TimeCLR** [55]: This approach uses dynamic time warping for augmentation. This augmentation method not only considers phase shifts and amplitude changes but also preserves the underlying structure and feature information of the time series. TimeCLR further combines the augmentation technique with InceptionTime, a powerful deep architecture for time series prediction.

To evaluate the robustness of the learned representations generated by our proposed `SimEXT` framework, we conducted experiments using several downstream models for block maxima prediction. Specifically, we employ the following downstream models: (1) **LSTM**, which is a bi-directional stacked Long Short-Term Memory network, coupled with a fully connected network. (2) **Transformer**, which is an attention-based transformer network. (3) **Informer**[5], which is a modified transformer-based framework. (4) **EVL**[34], which is a forecasting model that incorporates extreme value theory into its loss function. (5) **DeepExtrema**[81], which applies the Generalized Extreme Value (GEV) distribution for block maxima prediction.

To efficacy of the features generated by our proposed `SimEXT` framework is demonstrated by comparing them under the following experimental settings: (1) Original Features: In this setting, we directly feed the original features of the time series into different downstream forecasting models, without additional transformations. (2) Autoencoder only (**AE**): This approach utilizes only the autoencoder to reconstruct the original sample and learn the representation of the time series. (3) Autoencoder + Contrastive Learning (**AE + CL**): This setting incorporates both the autoencoder and contrastive learning modules to jointly learn the representation of the time series. (4) Autoencoder + Contrastive Learning + TWD (**AE + CL + TWD**): In this configuration, we employ the Tail-Weighted Distance (TWD) between empirical cumulative distribution functions (ECDFs) as distribution loss to further enhance the representation learning of the time series. (5)

Autoencoder + Contrastive Learning + CMD (**AE + CL + CMD**): This approach incorporates the Cramér-von Mises Distance (CMD) between the ECDFs of the GEV distributions for measuring distribution loss, enabling improved representation learning of the time series.

Evaluation on Representation Learning Models									
Methods	Climate			Hurricane			ECL		
	RMSE	Corr	F1	RMSE	Corr	F1	RMSE	Corr	F1
CoST [52]	3.08 ± 0.27	0.80 ± 0.03	0.85 ± 0.02	15.12 ± 0.25	0.87 ± 0.04	0.84 ± 0.04	0.98 ± 0.05	0.79 ± 0.03	0.82 ± 0.03
TS2Vec [51]	2.99 ± 0.28	0.81 ± 0.03	0.86 ± 0.01	15.03 ± 0.26	0.88 ± 0.03	0.86 ± 0.03	0.95 ± 0.03	0.78 ± 0.03	0.80 ± 0.04
TNC [98]	3.05 ± 0.23	0.80 ± 0.02	0.84 ± 0.03	15.06 ± 0.28	0.89 ± 0.02	0.85 ± 0.03	0.94 ± 0.04	0.78 ± 0.04	0.82 ± 0.04
TimeCLR [55]	3.18 ± 0.28	0.79 ± 0.03	0.82 ± 0.03	15.22 ± 0.30	0.85± 0.03	0.83 ± 0.04	1.04 ± 0.07	0.76 ±0.03	0.81 ± 0.03
SimEXT (TWD)	**2.82 ± 0.25**	**0.82 ± 0.02**	0.87 ± 0.02	14.86 ± 0.22	**0.90 ± 0.03**	0.88 ± 0.04	**0.88 ± 0.05**	**0.81 ± 0.03**	**0.85 ± 0.03**
SimEXT (CMD)	2.84 ± 0.28	0.80 ± 0.03	**0.88 ± 0.03**	**14.82 ± 0.24**	0.89 ± 0.03	**0.90 ± 0.02**	0.90 ± 0.04	0.80 ± 0.04	0.84± 0.04

Table 5.1 Performance comparison against state-of-the-art time series representation learning models for block maxima forecasting and classification. RMSE and correlation are calculated for block maxima prediction. F1 score is measured for classification extreme events (Category 4 and above for Hurricane and Above 90th Percentile for Climate and ECL data).

5.5.3 Experiment Settings

We partitioned each dataset into training, validation, and testing, according to an 8:1:1 ratio. We repeated the experiments 5 times using different random partitioning of the data. Prior to applying the various algorithms, the time series data is standardized to have zero mean and unit variance.

The encoder and decoder components of our framework employ a 4-layer bidirectional Long Short-Term Memory (LSTM) architecture, accompanied by fully connected layers. The training was facilitated using the Adam optimizer. For all the methods, we perform extensive hyperparameter tuning on the length of the embedding vector, the number of hidden layers (ranging from 2 to 6), the number of nodes (ranging from 8 to 128), the learning rate (ranging from 10^{-5} to 10^{-2}), and the batch size (ranging from 32 to 256). The optimal hyperparameters were determined using the Ray Tune framework, integrating an Asynchronous Successive Halving Algorithm (ASHA) scheduler to enable early stopping. All experiments were conducted on a single NVIDIA T4 GPU.

To evaluate the performance of the proposed framework, we employ the following evaluation metrics: (1) Root Mean Squared Error (RMSE): This metric quantifies the root mean squared error between the predicted block maxima and the ground truth block maxima within the forecast

Evaluation on Downstream Models										
Methods	Configuration	Climate			Hurricane			ECL		
		RMSE	Corr	F1	RMSE	Corr	F1	RMSE	Corr	F1
LSTM	Original Features	3.22 ± 0.25	0.73 ± 0.04	0.74 ± 0.04	15.51 ± 0.35	0.76 ± 0.35	0.78 ± 0.03	1.22 ± 0.09	0.74 ± 0.05	0.76 ± 0.03
	AE	3.18 ± 0.26	0.72 ± 0.05	0.75 ± 0.05	15.55 ± 0.37	0.78 ± 0.05	0.79 ± 0.04	1.12 ± 0.08	0.76 ± 0.03	0.77 ± 0.03
	AE + CL	3.11 ± 0.24	0.75 ± 0.04	0.78 ± 0.03	15.34 ± 0.32	0.81 ± 0.03	0.82 ± 0.03	1.06 ± 0.05	0.76 ± 0.02	0.79 ± 0.02
	AE + CL + TWD	3.02 ± 0.25	0.77 ± 0.02	0.78 ± 0.03	15.12 ± 0.26	0.82 ± 0.04	0.82 ± 0.03	1.00 ± 0.06	0.78 ± 0.03	0.80 ± 0.02
	AE + CL + CMD	2.97 ± 0.24	0.76 ± 0.03	0.79 ± 0.04	15.09 ± 0.24	0.82 ± 0.03	0.84 ± 0.02	0.99 ± 0.07	0.79± 0.03	0.79 ± 0.03
Informer	Original Features	3.19 ± 0.23	0.73 ± 0.04	0.78 ± 0.05	15.33 ± 0.32	0.79 ± 0.04	0.82± 0.03	1.11 ± 0.08	0.76 ± 0.03	0.77 ± 0.03
	AE	3.15 ± 0.21	0.74 ± 0.05	0.79 ± 0.05	15.30 ± 0.29	0.81 ± 0.03	0.81 ± 0.03	1.05 ± 0.05	0.75 ± 0.03	0.77 ± 0.04
	AE + CL	3.08 ± 0.23	0.77 ± 0.03	0.82 ± 0.04	15.08 ± 0.24	0.86 ± 0.04	0.84 ± 0.04	0.99 ± 0.07	0.77 ± 0.04	0.80 ± 0.03
	AE + CL + TWD	2.97 ± 0.25	0.81 ± 0.02	0.83 ± 0.03	14.94 ± 0.23	0.86 ± 0.04	0.86 ± 0.02	0.96 ± 0.05	0.79 ± 0.03	0.81 ± 0.03
	AE + CL + CMD	2.99 ± 0.26	0.78 ± 0.03	0.85 ± 0.04	14.97 ± 0.25	0.87 ± 0.04	0.89 ± 0.02	0.98 ± 0.06	0.80 ± 0.03	0.82 ± 0.03
Transformer	Original Features	3.21 ± 0.26	0.72 ± 0.05	0.78 ± 0.05	15.41 ± 0.36	0.78 ± 0.03	0.81 ± 0.04	1.14 ± 0.07	0.75 ± 0.04	0.75 ± 0.04
	AE	3.12 ± 0.23	0.73 ± 0.06	0.77 ± 0.06	15.36 ± 0.35	0.80 ± 0.04	0.82 ± 0.03	1.04 ± 0.06	0.77 ± 0.02	0.78 ± 0.03
	AE + CL	3.03 ± 0.25	0.78 ± 0.04	0.83 ± 0.05	15.13 ± 0.29	0.87 ± 0.05	0.85 ± 0.04	1.01 ± 0.07	0.77 ± 0.03	0.79 ± 0.02
	AE + CL + TWD	2.94 ± 0.29	0.80 ± 0.03	0.86 ± 0.03	14.98 ± 0.25	0.88 ± 0.03	0.87 ± 0.03	0.99 ± 0.06	0.80 ± 0.02	0.81 ± 0.04
	AE + CL + CMD	2.91 ± 0.26	0.79 ± 0.04	0.85 ± 0.04	14.95 ± 0.28	0.86 ± 0.04	0.89 ± 0.02	0.95 ± 0.04	0.80 ± 0.01	0.83 ± 0.02
EVL	Original Features	3.28 ± 0.27	0.72 ± 0.03	0.76 ± 0.05	15.44 ± 0.30	0.78 ± 0.04	0.78 ± 0.03	1.09 ± 0.07	0.77 ± 0.03	0.74 ± 0.05
	AE	3.21 ± 0.25	0.74 ± 0.04	0.78 ± 0.04	15.40 ± 0.32	0.80 ± 0.05	0.78 ± 0.03	1.04 ± 0.05	0.77 ± 0.02	0.77 ± 0.02
	AE + CL	3.18 ± 0.27	0.76 ± 0.03	0.82 ± 0.03	15.21 ± 0.28	0.80 ± 0.03	0.81 ± 0.03	1.02 ± 0.05	0.78 ± 0.03	0.78 ± 0.03
	AE + CL + TWD	3.07 ± 0.24	0.79 ± 0.02	0.84 ± 0.03	15.10 ± 0.21	0.84 ± 0.04	0.84 ± 0.04	0.98 ± 0.07	0.80 ± 0.02	0.82 ± 0.03
	AE + CL + CMD	3.05 ± 0.26	0.79 ± 0.03	0.83 ± 0.04	15.07 ± 0.24	0.83 ± 0.03	0.84 ± 0.02	0.99 ± 0.05	0.79± 0.02	0.81 ± 0.04
DeepExtrema	Original Features	3.10 ± 0.17	0.73 ± 0.05	0.78 ± 0.04	15.18 ± 0.26	0.83 ± 0.04	0.84 ± 0.04	1.02 ± 0.07	0.77 ± 0.04	0.80 ± 0.03
	AE	3.02 ± 0.19	0.75 ± 0.04	0.79 ± 0.05	15.21 ± 0.28	0.82 ± 0.03	0.84 ± 0.03	0.98 ± 0.06	0.79 ± 0.02	0.82 ± 0.02
	AE + CL	2.92 ± 0.21	0.78 ± 0.05	0.84 ± 0.04	15.04 ± 0.25	0.88 ± 0.04	0.88 ± 0.03	0.95 ± 0.06	0.78 ± 0.03	0.83 ± 0.03
	AE + CL + TWD	**2.79 ± 0.24**	**0.83 ± 0.02**	0.88 ± 0.02	14.81 ± 0.23	**0.90 ± 0.02**	0.88 ± 0.03	**0.86 ± 0.06**	**0.83 ± 0.02**	0.85 ± 0.03
	AE + CL + CMD	2.81 ± 0.26	0.81 ± 0.04	**0.89 ± 0.02**	**14.78 ± 0.25**	0.89 ± 0.02	**0.89 ± 0.02**	0.89 ± 0.05	0.82 ± 0.03	**0.86 ± 0.03**

Table 5.2 Performance comparison of downstream models under different experiment settings for block maxima forecasting and classification. RMSE and correlation are calculated for block maxima prediction. F1 score is measured for classification extreme events (Category 4 and above for Hurricane and Above 90th Percentile for Climate and ECL data).

window. (2) Correlation: The correlation between the predicted and ground truth block maxima within the forecast window serves as a measure of their alignment. (3) F1 score: This metric gauges the accuracy of extreme event detection within the forecast window. For the Hurricane dataset, an extreme event is defined as a hurricane intensity value surpassing 130 mph (i.e., category 4 and above hurricanes). For the climate and ECL datasets, extreme events are identified as temperature or electricity consumption values exceeding the 90th percentile of the values at a specific location.

5.5.4 Experimental Results

Table 5.1 provides a comprehensive comparison of the performance of `SimEXT` compared to other state-of-the-art representation learning methods. The results suggest that the proposed `SimEXT` framework, which incorporates distribution losses (TWD and CMD), outperforms all the baseline methods, namely CoST [52], TS2Vec [51], TNC [98], and TimeCLR [55], in terms of RMSE,

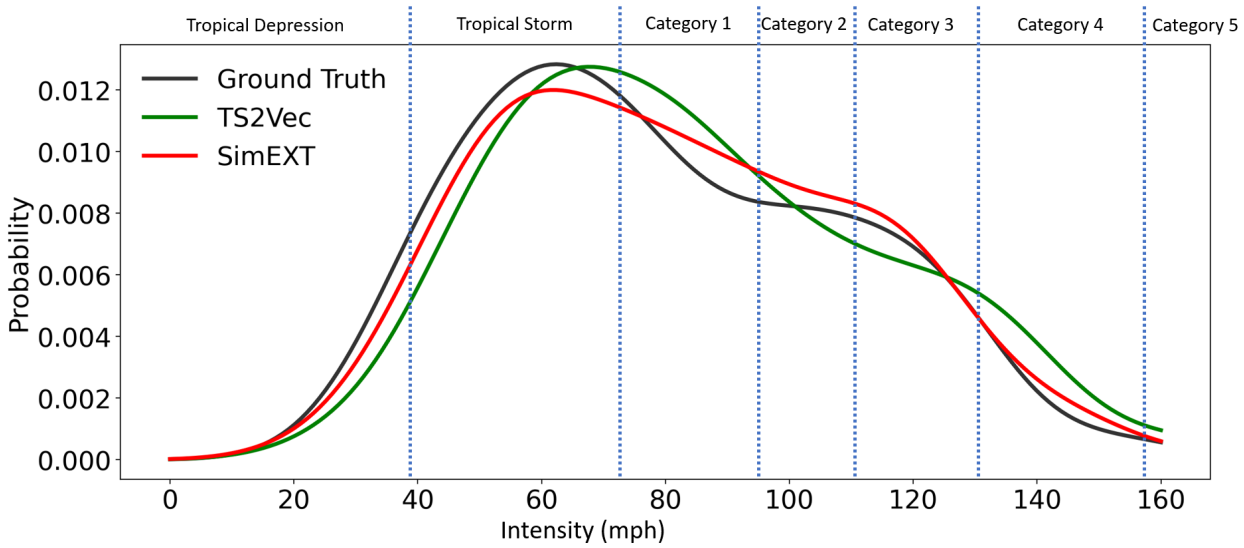


Figure 5.5 Probability distribution comparison of ground truth, `SimEXT`, and TS2Vec for block maxima forecasting with hurricane data.

Correlation, and F1 Score. To evaluate the significance of the observed performance improvement, a paired t-test was conducted to measure the significance of the difference in RMSE between `SimEXT` against the best-performing baseline method across 10 iterations for each dataset. The conventional threshold for statistical significance was set at $p < 0.05$. The results show that, for the Climate dataset, the p-value for RMSE is 0.0211; for the Hurricane dataset, the p-value is 0.0024 for RMSE; and for the ECL dataset, the p-value is 0.03379 for RMSE. Furthermore, Figure 5.5 provides a visual representation of the probability distribution comparison between the ground truth, `SimEXT`, and TS2Vec for block maxima forecasting using the hurricane dataset. The figure demonstrates that while the proposed `SimEXT` method does not accurately capture the ground truth distribution for tropical depression and storm categories, it significantly outperforms TS2Vec in matching the ground truth distribution for Category 1-5 hurricanes. From a practical perspective, the accurate forecasting of extreme events, particularly Category 1-5 hurricanes, holds far greater importance. These findings serve as compelling evidence supporting the superior performance of `SimEXT` in capturing tail distributions in time series while simultaneously learning representations. Notably, among the baseline methods, TS2Vec stands out as the top performer as this approach proves to be more suitable for time series data with diverse distributions and scales. Moreover, it is important to consider that extreme values in this context exhibit atypical distributions due to the influence of a

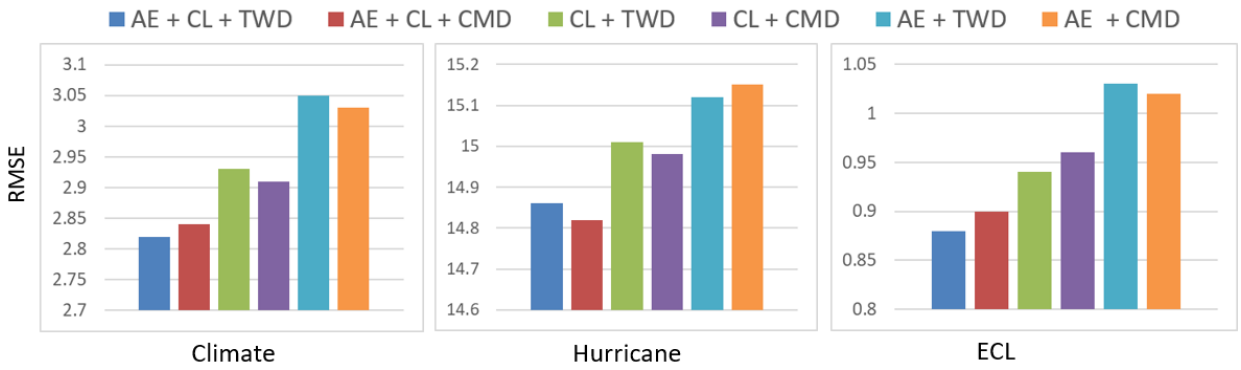


Figure 5.6 RMSE comparison of the different modules of `SimEXT` under different experimental setting for block maxima forecasting.

unique shape parameter governing their diverse shapes.

Table 2 presents the results of applying `SimEXT` to five downstream models (LSTM, Informer, Transformer, EVL, and DeepExtrema) for block maxima forecasting. The table showcases the performance of `SimEXT` under five different settings: Original Features, AE, AE + CL, AE + CL + TWD, and AE + CL + CMD. Consistently, the results demonstrate that the incorporation of distribution losses (AE + CL + TWD and AE + CL + CMD) significantly enhances the performance of block maxima prediction for all downstream models. This finding underscores the effectiveness of the proposed distribution losses in capturing extreme values and improving forecasting accuracy. Moreover, the results reveal that the DeepExtrema model for downstream tasks, when using it with the `SimEXT` representation learning approach, achieves the best performance, exhibiting the lowest RMSE and the highest correlation and F1 score.

The combined evidence from Tables 5.1 and 5.2 supports the superiority of `SimEXT` in capturing extreme values and enhancing representation learning for time series data. By incorporating distribution losses, `SimEXT` outperforms state-of-the-art methods. Additionally, when applied to downstream models for block maxima forecasting, `SimEXT` consistently improves performance across various settings.

5.5.5 Ablation study

In this study, investigate the effect of gradually incorporating the AE, CL, and TWD/CMD modules into the proposed `SimEXT` framework. Figure 5.6 shows the RMSE values obtained

when using different modules of the SimEXT framework. The results suggest that incorporating CL (contrastive learning) alone yields more substantial benefits compared to incorporating only AE (Autoencoder) for all datasets used. This finding aligns with the current understanding that contrastive learning is generally more beneficial in learning meaningful representations compared to autoencoder-based methods. Nevertheless, our results also highlight the positive impact of incorporating the distribution loss, ultimately leading to the best performance across all three datasets. This suggests that while the autoencoder module alone may not deliver optimal results on its own, it does exert a positive impact when integrated alongside the CL and distribution loss (TWD or CMD).

5.6 Conclusion

This chapter introduces SimEXT, a novel self-supervised learning framework for modeling extreme values in time series. The framework employs a wavelet-based data augmentation approach along with a combination of contrastive learning with auto-encoders to learn the representation of the time series. To ensure that the representation preserves the extreme values, SimEXT incorporates a distribution loss function that is biased towards capturing the block maxima of the time series. Our experimental results on real-world data demonstrate SimEXT can boost the performance of existing representation learning and downstream approaches for forecasting block maxima.

CHAPTER 6

FIDE: FREQUENCY-INFLATED CONDITIONAL DIFFUSION MODEL FOR EXTREME-AWARE TIME SERIES GENERATION

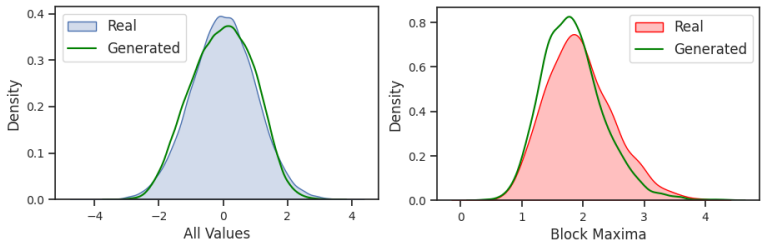
6.1 Introduction

Generative models [61, 59, 60] have revolutionized the AI landscape, demonstrating their broad applicability across diverse domains, including computer vision and natural language processing. Such models are designed to learn the underlying data distribution and exhibit resilience to overfitting while promoting automatic feature extraction. Diffusion-based models [13, 62], in particular, have emerged as a popular generative AI method due to their capability to generate realistic, high-quality data. This chapter examines the application of diffusion-based models for time series generation. In particular, we investigate the following issue: *How well do existing diffusion models preserve the fidelity of extreme values (i.e., tail distribution) of the original time series?*

The modeling of extreme values in time series is essential for informed decision-making across diverse applications, including weather forecasting, earthquake prediction, and disease outbreak detection. Effective generative modeling of these extremes is important as it aids in learning the underlying data distribution, facilitating data augmentation, and improving uncertainty estimation, all of which are crucial for developing robust risk management strategies and enhancing disaster preparedness measures. While there has been growing research on applying diffusion models for time series [11, 12], their ability to preserve the distribution of extreme values remains largely underexplored. In this study, we examine how effectively diffusion models preserve extreme values in the form of block maxima [10], defined as the peak value within a specified time window.

To illustrate the difficulty of modeling the distribution of block maxima, Figure 6.1 shows the result of applying the Denoising Diffusion Probabilistic Model (DDPM) [13] to a synthetic AR(1) dataset. While DDPM shows proficiency in generating samples that closely align with the overall data distribution (left diagram), it struggles to preserve the distribution of block maxima values (right diagram) when the generated time series is partitioned into disjoint time windows.

In this chapter, we identify the key shortcomings of existing diffusion models that hamper their



(a) All Values Distribution (b) Block Maxima Distribution

Figure 6.1 Comparing the distributions of all values and block maxima values for real and generated samples using DDPM [13] when applied to the synthetic AR(1) dataset.

ability to accurately model block maxima values. We then present a novel framework to overcome this limitation. Our key observation is that unusually large block maxima values, often linked to abrupt temporal changes, are strongly associated with high-frequency components of the time series. As the diffusion-based generative model gradually introduces noise with a linearly increasing variance schedule, it slowly diminishes the long-term trends (low-frequency components) of the time series while quickly attenuating the high-frequency components. These high-frequency components are crucial for reproducing extreme block maxima values. This limitation hampers the accurate representation of the block maxima, necessitating the development of new techniques.

To address this challenge, we propose an end-to-end diffusion model framework termed `FIDE`. First, to mitigate the rapid dissipation of high-frequency components in the diffusion model, we introduce a novel high-frequency inflation strategy within the frequency domain. This strategic augmentation ensures the sustained emphasis on block maxima, preventing their premature fade-out. We further employ a conditional diffusion-based generative modeling approach to guide the time series generation by conditioning on their block maxima. To enhance the preservation of the block maxima distribution while learning the overall data distribution, we extend the conventional framework with a regularization term in the loss function based on the negative log-likelihood of the Generalized Extreme Value (GEV) distribution. Using these strategies, we empirically show that our approach effectively addresses the challenges of learning the overall data distribution while simultaneously preserving the block maxima distribution.

6.2 Related Works

Time series generation has been a subject of extensive research, leveraging a variety of statistical [58] and machine-learning [99, 7] techniques to capture temporal dependencies and complexities within data. Generative methods, including Generative Adversarial Networks (GANs) [59], Variational Autoencoders (VAEs) [60], normalizing flows [61], and diffusion-based approaches [13, 62], have demonstrated efficacy in time series generation and garnered interest due to their ability to learn underlying data distributions for data generation. Normalizing flows are constrained by their computational complexity, limited expressiveness, and suboptimal sample quality, thereby restricting their capacity for effective modeling. Numerous works have delved into enhancing GANs, introducing variations like RcGAN [63] and TimeGAN [7], which have demonstrated improvements in generating realistic time series data. TimeGAN [7] specifically adopts a GAN architecture to generate time-series data, employing an encoder and decoder to transform a time-series sample into latent vectors. However, GAN-based generative models are susceptible to issues like mode collapse and unstable behavior during training. While VAEs have not been extensively applied to synthetic time series generation, their effectiveness in addressing related challenges, such as time series imputation [64], suggests their potential utility in this domain. Diffusion-based models are also gaining traction for their ability to generate high-quality data such as images and videos, bypassing the challenges associated with discriminator networks in GANs and avoiding the artifact-prone lower-dimensional latent spaces of VAEs. There are a couple of diffusion-based works [11, 12] that have been employed for time series, but they are specifically designed for discriminative tasks.

While generative AI for time series offers numerous advantages, it has not been extensively explored, especially in terms of modeling extreme values. The difficulty of modeling extremes using generative models such as normalizing flows [65] has been recognized in previous research. [66] and [67] highlight the inability of normalizing flows to accurately capture heavy-tailed marginal distributions. Specifically, these studies show that any attempt to map heavy-tailed distributions to light-tailed distributions (e.g., Gaussian) cannot maintain Lipschitz-boundedness. However, this challenge remains largely unexplored within the realm of diffusion models.

6.3 Preliminaries

Consider a time series dataset $\mathcal{D} = \{\mathbf{x}_{m,0}\}_{m=1}^{M}$ comprising of M samples, where each sample $\mathbf{x}_{m,0} = (x_{m,0}^1, x_{m,0}^2, \cdots, x_{m,0}^T)$ is a univariate time series of finite length T. Let $\mathbf{f}_{m,0} \in \mathbb{R}^T$ be the Fourier coefficients, whose k-th frequency component is obtained by applying the following discrete Fourier transform on $\mathbf{x}_{m,0}$:

$$f_{m,0}^k = \sum_{t=1}^{T} x_{m,0}^t \; e^{-i2\pi tk/T} = \sum_{t=1}^{T} \left[x_{m,0}^t \cos\left(\frac{2\pi tk}{T}\right) - i \cdot x_{m,0}^t \sin\left(\frac{2\pi tk}{T}\right) \right] \tag{6.1}$$

The time series can be recovered from its Fourier coefficients using the following inverse discrete Fourier transform:

$$x_{m,0}^t = \frac{1}{T} \sum_{k=1}^{T} f_{m,0}^k \; e^{i2\pi tk/T} = \frac{1}{T} \sum_{k=1}^{T} \left(f_{m,0}^k \cos\left(\frac{2\pi tk}{T}\right) + i \cdot f_{m,0}^k \sin\left(\frac{2\pi tk}{T}\right) \right) \tag{6.2}$$

For brevity, we will drop the sample subscript m when it is clear from the context. Let, $\omega_k = \frac{2\pi k}{T}$ be the k-th frequency in Fourier transform.

Given a sample $\mathbf{x}_0$, let y_0 be its corresponding block maxima value, where $y_0 = \max_{\tau \in \{1,\cdots,T\}} x_0^\tau$. The distribution of the block maxima values is governed by the Generalized Extreme Value (GEV) distribution given in (3.1). Given M independent block maxima values, denoted as $\{y_{1,0}, y_{2,0}, \cdots, y_{M,0}\}$, with the cumulative distribution function given by Equation (3.1), the distribution parameters can be estimated using the maximum likelihood approach by minimizing the negative log-likelihood function in (3.4)

6.4 On the Rapid Dissipation of Block Maxima in Diffusion Models

While diffusion models have demonstrated remarkable capabilities in learning complex data distributions, a significant challenge arises in accurately capturing the distribution of block maxima values, as evidenced by Figure 6.1. Addressing this shortcoming is crucial for enhancing the performance and applicability of these models across various domains. In this section, we delve into the root cause of this phenomenon and present insightful observations that shed light on the underlying issue.

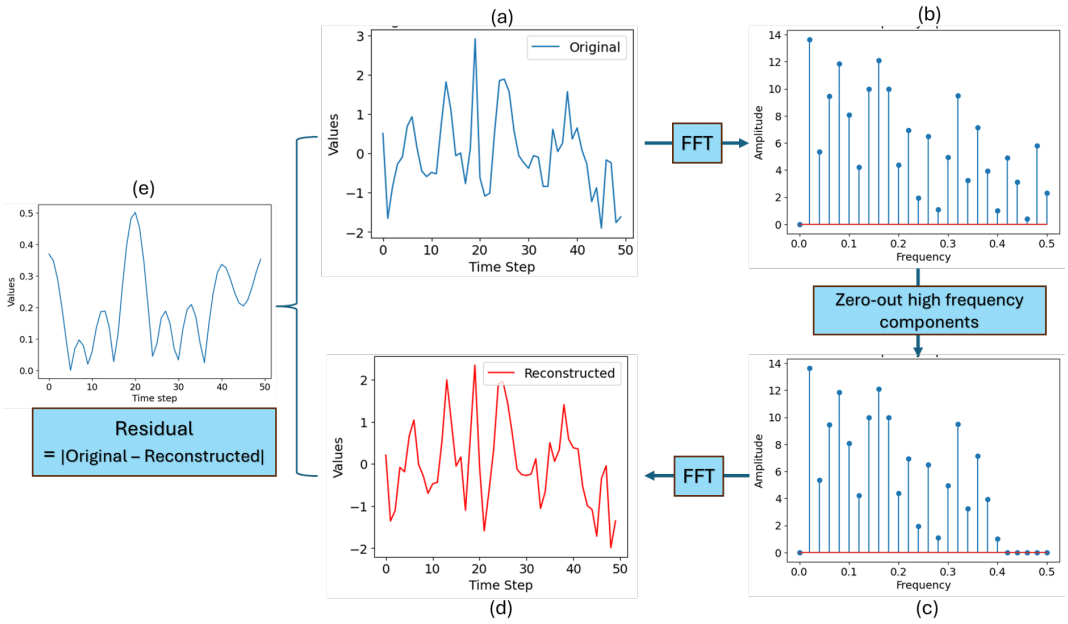


Figure 6.2 Removal of high-frequency components from daily temperature time series significantly alters the magnitude of its block maxima value (at time step 20), as evidenced by its high residual.

6.4.1 Relationship between Abrupt Block Maxima and High-Frequency Components

Our first key observation reveals a ***connection between block maxima with abrupt changes and the high-frequency components*** of many real-world time series. Block maxima, often characterized by their rarity and abrupt temporal changes, are intrinsically linked to the high-frequency components of the data. This relationship is observed in many real-world datasets, where the block maxima values do not typically evolve smoothly but rather emerge through large deviations from their adjacent values.

To illustrate this, consider the real-world temperature time series depicted in Figure 6.2. In this plot, we first transform the time series into its Fourier domain, obtaining its frequency components, and selectively zeroing out its top-5 highest frequency components. We then reconstruct the time series via its inverse Fourier transform and compute the difference between the original and reconstructed time series. The recovered signal exhibits a notable distortion around the block maxima value, as evidenced by the larger residual at time step 20, where the block maxima value occurs. This suggests that the removal of high-frequency components of a time series has a significant impact on the accurate representation of block maxima values.

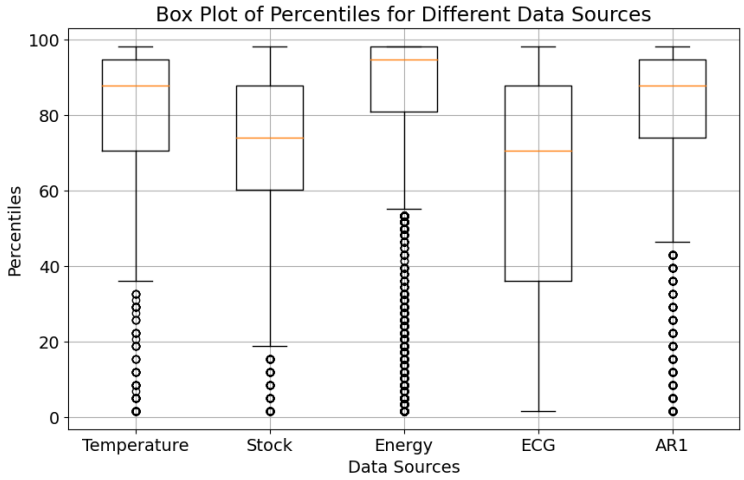


Figure 6.3 Percentile distribution of first-order derivatives for the block maxima values in different time series datasets. Observe that the derivatives tend to exhibit elevated percentile values.

Definition 2 (Abrupt Block Maxima)**.** *Let,* $\mathbf{x}_0 \in \mathbb{R}^T$ *be a time series of length* T *and* $y_0 \equiv x_0^\tau = \max_{t \in 1,\ldots,T} x_0^t$ *be its block maxima, where* τ *is the time step of the block maxima. Then,* x_0^τ *is considered an abrupt block maxima if* $\frac{dx}{dt}|_{t=\tau} > \rho$*, where* ρ *is a threshold.*

We argue that the block maxima values in real-world time series often exhibit an abrupt change behavior compared to the non-block maxima values. To substantiate this, we conduct an empirical analysis across five distinct datasets, wherein we assess the percentile distribution of the first-order derivatives associated with the block maxima values, as depicted in Figure 6.3. Specifically, given a time series, we first partition it into a set of disjoint time windows and compute the block maxima value within each window along with the first-order difference, $\Delta x^t = x_0^t - x_0^{t-1}$, for each time step t. We then compute the percentile of Δx^τ associated with the block maxima x_0^τ of the window and plot its distribution using a boxplot as shown in Figure 6.3. Our findings affirm the conjecture that the block maxima tend to exhibit an elevated value for its time derivative (with a median larger than 70% of all first-order differences), thereby indicating a notable association between block maxima occurrences and the abrupt changes in a time series.

More importantly, the abrupt changes are strongly influenced by the high-frequency components of the time series. This can be observed by differentiating the inverse Fourier transform shown in

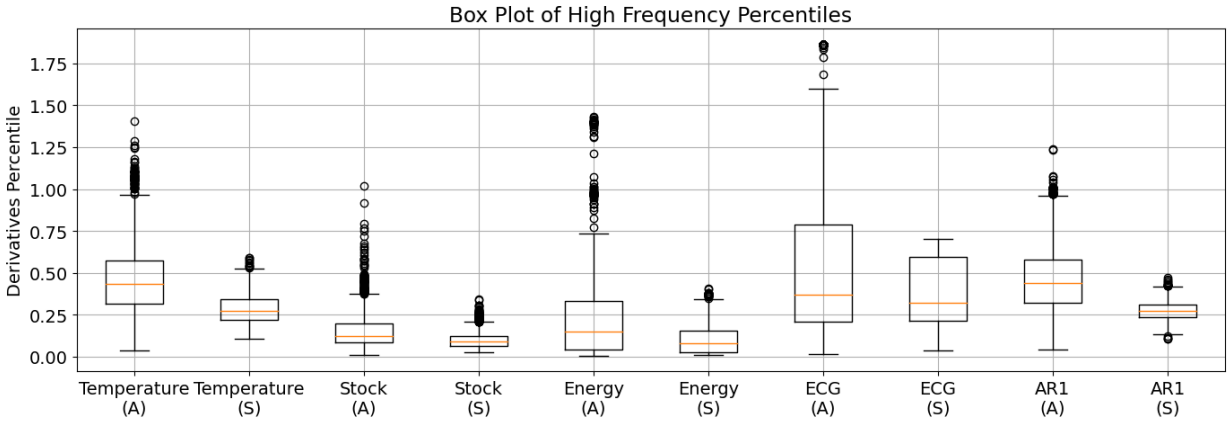


Figure 6.4 The summation of high-frequency terms for abrupt (A) changes is consistently higher than for smooth (S) changes. (A) denotes abrupt changes, (S) denotes smooth changes.

Equation 6.2 and decomposing the derivative into low and high-frequency components:

$$\frac{dx^t_{m,0}}{dt} = \frac{1}{T}\sum_{k=1}^{T} f^k_{m,0}\frac{d}{dt}\left[e^{i2\pi tk/T}\right] = \frac{1}{T}\sum_{k=1}^{T}\frac{i2\pi k}{T} f^k_{m,0} e^{i2\pi tk/T} = \frac{1}{T}\sum_{k=1}^{T} i\omega_k f^k_{m,0} e^{i\omega_k t},$$

where $\omega_k = \frac{2\pi k}{T}$. Let κ be the threshold index for dividing the frequencies into low ($k \leq \kappa$) and high ($k > \kappa$) frequency components. Then, we have:

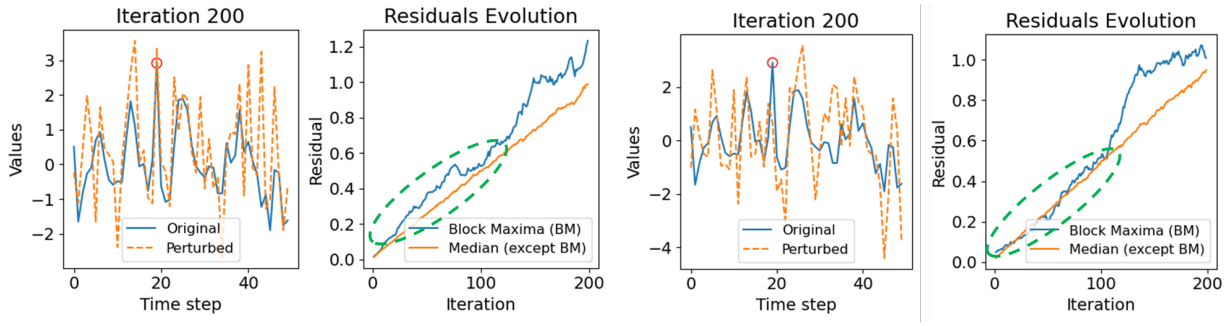
$$\frac{dx^t_{m,0}}{dt} = \frac{1}{T}\sum_{l=1}^{\kappa} i\omega_l f^l_{m,0} e^{i\omega_l t} + \frac{1}{T}\sum_{h=\kappa+1}^{T} i\omega_h f^h_{m,0} e^{i\omega_h t} \tag{6.3}$$

To illustrate the impact of high-frequency components on the abrupt changes, we compute the value of the second term in Equation (6.4) for time steps with abrupt[1] (A) and non-abrupt (S) changes for various datasets. The results shown in Fig. 6.4 suggest that the sum of high-frequency terms for abrupt changes is consistently higher than the sum of high-frequency terms for non-abrupt changes. In essence, abrupt block maxima values often manifest as high-frequency components in the Fourier domain as they introduce sharp transitions in the time domain signal. This explains the high residual shown in Fig. 6.2 for the block maxima when the high-frequency components are zeroed out.

6.4.2 Attenuation of Block Maxima in Diffusion Model

Our second key observation unveils a concerning behavior of diffusion models: ***the addition of noise diminishes high-frequency components, i.e., block maxima, at a faster rate*** compared to other

[1]A time step has abrupt change if it's $\frac{dx}{dt}$ is in the top 90%.



(a) Attenuation of Block Maxima by DDPM (b) Effect of High Frequency Inflation

Figure 6.5 A comparison of the effects of noise addition by existing DDPM versus high-frequency inflation on the block maxima of generated samples.

values in the signal. As diffusion-based generative models gradually introduce noise characterized by a linearly increasing variance scheduler, they inadvertently attenuate the signals associated with high-frequency components. These components, as established in our first observation, are crucial for accurately reproducing block maxima. Concurrently, the models effectively capture the long-term trends and low-frequency components, which are conducive to learning the overall data distribution. However, the extents of extreme events dissipate more rapidly, hindering the model's ability to adeptly learn the distribution of these rare occurrences.

Figure 6.5a illustrates this phenomenon. By tracking the evolution of residuals, or the differences between the original and perturbed time series generated by DDPM, we observe a discernible pattern: block maxima dissipate at a faster rate compared to other values, as evidenced by the higher residuals associated with these extreme points. Notably, in the early iterations highlighted by the green circle, the substantially higher residual suggests that the block maxima signal is rapidly transformed into noise, outpacing the dissipation rate of other values. This behavior poses a formidable challenge for diffusion models in effectively capturing the distributions of the block maxima values.

To substantiate our observations, we provide a theoretical justification for the rapid dissipation of block maxima during the forward process of the diffusion model

Let $\mathbf{x}_0$ be an input sample and $\mathbf{x}_n$ be the perturbed sample after n iterations of the forward process, where $x_n^t = x_{n-1}^t + \epsilon_n^t$ and $\epsilon_n^t \sim \mathbb{N}(0, \sigma_{\epsilon_n^t}^2)$ is Gaussian noise. Due to the linearity of the

Fourier transform operator $\mathcal{F}$, we have:

$$\mathcal{F}(\mathbf{x}_n) = \mathcal{F}(\mathbf{x}_{n-1}) + \mathcal{F}(\epsilon_n) \quad \Longrightarrow \quad f_n^k = f_{n-1}^k + \mathcal{E}_n^k \tag{6.4}$$

Let σ_n^2 and $\sigma_{\epsilon_n}^2$ be the variances of perturbed time series and noise respectively at diffusion step n while σ_{n-1}^2 is the variance of the time series at diffusion step $n-1$. Note that $\sigma_{\epsilon_n}^2$ increases linearly according to a linear noise scheduler as the diffusion step increases. Let $\mathcal{S}_n^f(\omega_k)$ and $\mathcal{S}_n^{\mathcal{E}}(\omega_k)$ denote the power spectral density (PSD) for the k-th frequency component of the perturbed time series and noise respectively for the diffusion step n, where $\omega_k = \frac{2\pi k}{T}$.

Our theoretical analysis is based on the following assumptions:

Assumption 1. *An abrupt block maxima is linked to the high-frequency components of the time series.*

Assumption 2. *The noise ϵ_n^t is a stationary random process with a constant power spectral density (PSD) for diffusion step n, i.e., $\forall k : \mathcal{S}_n^{\mathcal{E}}(\omega_k) \approx \sigma_{\epsilon_n}^2$.*

Assumption 3. *The power spectral density (PSD) of the perturbed time series for diffusion step n can be modeled using the following generalized Gaussian function:*

$$\mathcal{S}_n^f(\omega_k) = \sigma_n^2 \cdot \exp(-\alpha_n |\omega_k|^{\beta_n}) \tag{6.5}$$

where α_n is a scaling factor, and β_n is a shape parameter.

Remark 1. *The rationale and supporting evidence for Assumption 1 is presented in Section 6.4.*

Remark 2. *Assumption 2 is intuitive as the Gaussian noise used in the diffusion model has approximately constant PSD over the frequency range of interest.*

Remark 3. *Assumption 3 is reasonable as for most real-world time series, the energy spectrum is localized at the lower frequency ($\omega_k \approx 0$), also known as the fundamental frequency, which quickly decays with increasing frequency ($\omega_k \rightarrow \omega_{max}$) [100]. Therefore, we use the generalized Gaussian function to model this decaying behavior. Note that, the exponential decay behavior*

will eventually transition into a uniform distribution according to the diffusion model's forward process. Consequently, the shape β_n of the distribution function (Eqn 6.5) is not constant; instead, it evolves with the diffusion step n to remain consistent with the diffusion model's forward process. Initially, when n is small, $1 \leq \beta_n \leq 2$, representing an exponential decay. However, as $n \to N$, where N is the final diffusion step, $\beta_n \to \infty$, and the PSD transitions to a uniform distribution. This transformation occurs because, according to the forward process of the diffusion model, as n increases, Gaussian noise with linearly increasing variance is added to the perturbed time series, making the signal increasingly noise-like until it becomes white noise at step N.

We then present the following lemma for the diffusion forward process.

Lemma 1. *The difference between the variance of the perturbed time series x_n^t and the variance of the noise ϵ_n^t is equal to a constant ζ such that:*

$$\sigma_n^2 - \sigma_{\epsilon_n}^2 = \zeta \tag{6.6}$$

where $\zeta = \sigma_0^2 + \sum_{i=1}^{n-1} \sigma_{\epsilon_i}^2$

Proof. First, we have the following:

$$x_n^t = x_{n-1}^t + \epsilon_n^t \tag{6.7}$$

Applying the variance operator to both sides of Equation 6.7 and utilizing the property that the variance of a sum of independent random variables is the sum of their individual variances, we obtain:

$$\sigma_n^2 = \sigma_{n-1}^2 + \sigma_{\epsilon_n}^2 \tag{6.8}$$

where $\sigma_{\epsilon_n}^2$ denotes the variance of the noise at step n. Recursively applying Equation 6.8, we can express the variance of the perturbed time series at step n as:

$$\sigma_n^2 = \sigma_0^2 + \sum_{i=1}^{n} \sigma_{\epsilon_i}^2 \tag{6.9}$$

Subtracting $\sigma_{\epsilon_n}^2$ from both sides of Equation 6.9, we get:

$$\sigma_n^2 - \sigma_{\epsilon_n}^2 = \sigma_0^2 + \sum_{i=1}^{n-1} \sigma_{\epsilon_i}^2 \tag{6.10}$$

Therefore, the difference between the variance of the perturbed time series and the variance of the noise at step n is equal to the constant $\zeta = \sum_{i=1}^{n-1} \sigma_{\epsilon_i}^2$, which completes the proof. □

The lemma establishes a crucial bound on the difference between the variances of the perturbed time series and the noise, which is leveraged in the subsequent theorem to analyze the behavior of the Fourier transform of the perturbed time series at low and high frequencies.

Using Lemma 1 and Assumptions 1, 2, and 3, we can present the following theorem:

Theorem 1. *Under certain mild assumptions (1, 2, 3), the ratio of high-frequency and low-frequency components after perturbation during the forward process of the diffusion model is:*

$$\frac{\lim_{k \to k_{\max}} |f_n^k|^2}{\lim_{k \to 0} |f_n^k|^2} = \delta \ll 1 \tag{6.11}$$

where $k_{\max}$ is the index of the maximum frequency and $\delta = f_n^{k_{\max}}$, which is generally close to 0.

Proof. For low frequencies, i.e., $\omega_k \to 0$, taking the limit as $\omega_k \to 0$ on the expression for the signal power spectral density $\mathcal{S}_n^f(\omega)$, we have:

$$\lim_{k \to 0} \mathcal{S}_n^f(\omega_k) = \lim_{k \to 0} \sigma_n^2 \cdot \exp(-\alpha_n |\omega_k|^{\beta_n}) = \sigma_n^2$$

since $\exp(-\alpha_n |0|^{\beta_n}) = \exp(0) = 1$. Therefore, as $k \to 0$, $\mathcal{S}_n^f(\omega_k) = |f_n^k|^2$ approaches the variance σ_n^2. So, we can write:

$$\lim_{k \to 0} \mathcal{S}_n^f(\omega_k) = \lim_{k \to 0} |f_n^k|^2 = \sigma_n^2 = \zeta + \sigma_{\epsilon_n}^2 = \sigma_0^2 + \sum_{i=1}^{n-1} \sigma_{\epsilon_i}^2 + \sigma_{\epsilon_n}^2 = \sigma_0^2 + \sum_{i=1}^{n} \sigma_{\epsilon_i}^2$$

Similarly, for high frequencies, i.e., $k \to k_{\max}$, taking the limit as $\omega \to \omega_{\max}$ on the expression for the signal power spectral density $\mathcal{S}_n^f(\omega_k)$ (Eq. (6.5)), we have:

$$\lim_{k \to k_{\max}} \mathcal{S}_n^f(\omega_k) = \lim_{k \to k_{\max}} \sigma_n^2 \cdot \exp(-\alpha_n |\omega|^{\beta_n}) \to \sigma_n^2 \cdot \delta$$

where $\delta = f_n^{k_{\max}} \approx 0$.

As $k \to k_{\max}$, $\mathcal{S}_n^f(\omega_k)$ approaches $\sigma_n^2 \cdot \delta$. We can also write:

$$\lim_{k \to k_{\max}} |f_n^k|^2 = \delta \cdot \left(\zeta + \sigma_{\epsilon_n}^2\right) = \delta \cdot \left(\sigma_0^2 + \sum_{i=1}^{n} \sigma_{\epsilon_i}^2\right)$$

Taking the ratio of high-frequency and low-frequency components after perturbations yields:

$$\frac{\lim_{k \to k_{\max}} |f_n^k|^2}{\lim_{k \to 0} |f_n^k|^2} = \delta$$

□

Thereby, high-frequency components or abrupt block maxima dissipate rapidly compared to low-frequency components or smooth changes. In short, our findings shed light on a fundamental limitation of diffusion models while modeling block maxima and underscore the need for tailored approaches to preserve its distribution.

6.5 Proposed Framework: `FIDE`

In this section, we present the detailed methodology of our proposed framework, addressing the challenges associated with capturing extreme event distributions within generative modeling frameworks. Figure 6.6 provides an overview of the `FIDE` framework.

6.5.1 High Frequency Components Inflation

In order to counteract the rapid decay of high-frequency components in the frequency domain while adding noise in the forward process of DDPM, we present a strategy for high-frequency inflation. Let $\mathbf{f}_0 = \mathbf{FFT}(\mathbf{x}_0)$ denote the vector of Fourier coefficients resulting from applying the discrete Fourier transform to the time series$\mathbf{x}_0$. These coefficients are arranged in ascending order from lowest to highest frequency. Consequently, the last κ elements of $\mathbf{f}_0$ correspond to the coefficients associated with the κ highest frequencies. Our goal is to inflate the top-κ frequency components of $\mathbf{f}_0$ as follows:

$$\Gamma^i = \begin{cases} 1, & \text{if } i \leq \kappa \\ \gamma, & \text{if } i > T - \kappa \end{cases} \quad \text{and} \quad \bar{\mathbf{f}}_0 = \mathbf{\Gamma} \odot \mathbf{f}_0$$

where $\gamma > 1$ is the inflation weight and $\odot$ denotes the element-wise multiplication.

With the modified coefficients $\bar{\mathbf{f}}_0$, the inverse Fourier transform (IFFT) is applied to get the modified time series, $\bar{\mathbf{x}}_0 = \mathbf{IFFT}\left(\bar{\mathbf{f}}_0\right)$, containing the inflated high-frequency components. Here, the high-frequency components are inflated by $\gamma > 1$. The following theorem shows how this inflation

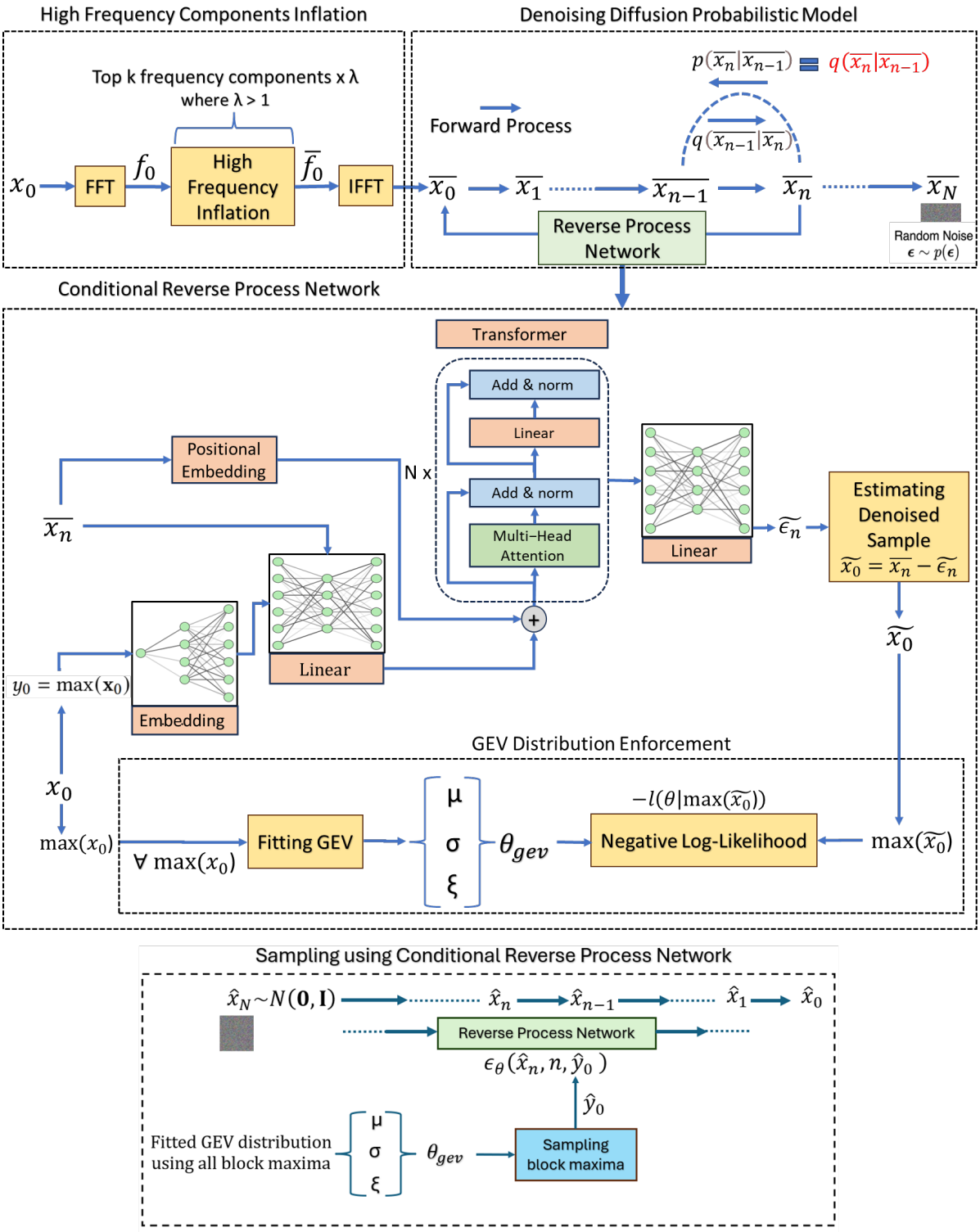


Figure 6.6 Proposed FIDE framework for generating time series with extreme events.

strategy helps the high-frequency components (block maxima) diminish less rapidly in the diffusion forward process compared to before.

Theorem 2. *Let $\overline{f_n^k}$ be the Fourier coefficient after inflating high-frequency components with a factor of γ such that $\gamma > 1$. Let, $\delta = f_n^{k_{\max}}$ be the Fourier coefficient of the maximum frequency before inflation and $\delta' = \delta \cdot \gamma = \overline{f_n^{k_{\max}}}$ be the Fourier coefficient of the maximum frequency after inflation and. Then, using Lemma 1 and under certain mild assumptions (1, 2, 3), the ratio of high-frequency and low-frequency components after inflation and perturbations is:*

$$\frac{\lim_{k \to k_{\max}} |\overline{f_n^k}|^2}{\lim_{k \to 0} |\overline{f_n^k}|^2} = \delta \cdot \gamma \tag{6.12}$$

Proof. For low frequencies, i.e., $\omega_k \to 0$, taking the limit as $\omega_k \to 0$ on the expression for the signal power spectral density $\overline{\mathcal{S}_n^f}(\omega)$, we have:

$$\lim_{k \to 0} \overline{\mathcal{S}_n^f}(\omega_k) = \lim_{k \to 0} \sigma_n^2 \cdot \exp(-\alpha_n |\omega_k|^{\beta_n}) = \sigma_n^2$$

since $\exp(-\alpha_n |0|^{\beta_n}) = \exp(0) = 1$. Therefore, as $k \to 0$, $\overline{\mathcal{S}_n^f}(\omega_k) = |f_n^k|^2$ approaches the variance σ_n^2. So, we can write:

$$\lim_{k \to 0} \overline{\mathcal{S}_n^f}(\omega_k) = \lim_{k \to 0} |\overline{f_n^k}|^2 = \sigma_n^2 = \zeta + \sigma_{\epsilon_n}^2 = \sigma_0^2 + \sum_{i=1}^{n-1} \sigma_{\epsilon_i}^2 + \sigma_{\epsilon_n}^2 = \sigma_0^2 + \sum_{i=1}^{n} \sigma_{\epsilon_i}^2$$

Similarly, for high frequencies, i.e., $k \to k_{\max}$, taking the limit as $\omega \to \omega_{\max}$ on the expression for the signal power spectral density $\overline{\mathcal{S}_n^f}(\omega_k)$ (Eq. (6.5)), we have:

$$\lim_{k \to k_{\max}} \overline{\mathcal{S}_n^f}(\omega_k) = \lim_{k \to k_{\max}} \sigma_n^2 \cdot \exp(-\alpha_n |\omega|^{\beta_n}) \to \sigma_n^2 \cdot \delta'$$

where $\delta' = \delta \cdot \gamma$

As $k \to k_{\max}$, $\overline{\mathcal{S}_n^f}(\omega_k)$ approaches $\sigma_n^2 \cdot \delta \cdot \gamma$. We can also write:

$$\lim_{k \to k_{\max}} |\overline{f_n^k}|^2 = \delta \cdot \gamma \cdot \left(\zeta + \sigma_{\epsilon_n}^2\right) = \delta \cdot \gamma \cdot \left(\sigma_0^2 + \sum_{i=1}^{n} \sigma_{\epsilon_i}^2\right)$$

Taking the ratio of high-frequency and low-frequency components after perturbations yields:

$$\frac{\lim_{k \to k_{\max}} |\overline{f_n^k}|^2}{\lim_{k \to 0} |\overline{f_n^k}|^2} = \delta \cdot \gamma$$

□

Thus, by applying high-frequency inflation, the high-frequency components including abrupt block maxima will be preserved by a factor of γ compared to the previous case. We can see the effects of this inflation empirically as well. Figure 6.5b shows how inflating the high-frequency components helps in preserving the block maxima values for longer iterations of the diffusion model. This enables the block maxima after high-frequency inflation to dissipate at a similar rate compared to other values in the earlier iterations. The diffusion model will have more iterations to capture the block maxima signal.

6.5.2 Forward Process

We use the inflated time series $\overline{\mathbf{x}_0}$ as input time series to be perturbed during the forward process instead of $\mathbf{x}_0$. By adopting $\overline{\mathbf{x}_0}$ as the reference for the unperturbed sample, we ensure that the denoising diffusion process takes advantage of the enhanced representation provided by the inflated high-frequency components. This nuanced adjustment contributes to the efficacy of our proposed framework in capturing and preserving essential information during the diffusion process.

6.5.3 Conditional Reverse Diffusion Process

To enable the generation of samples conditioned on block maxima, we extend the conventional diffusion model to a conditional model. Here, the reverse process is conditioned on block maxima y_0. Grounded in extreme value theory [10], the block maxima y_0 are mandated to adhere to an extreme value distribution, distinctly diverging from the distribution of all values $\mathbf{x_0} \sim p_\theta(\mathbf{x}0)$. This mandates a strategic shift in our learning objective. Rather than marginally targeting $p_\theta(\mathbf{x}0)$, our objective now extends to mastering the joint distribution $p_\theta(\mathbf{x}_0, y_0)$, driven by a nuanced understanding of the unique characteristics inherent in extreme events and their crucial impact on the overall distribution. We formally extend the diffusion model's marginal distribution to a joint distribution in the following theorem.

Theorem 3. *Let us consider the extension of the conventional diffusion model from learning a marginal distribution $p_\theta(\mathbf{x}_0)$ to a joint distribution $p_\theta(\mathbf{x}_0, y_0)$ conditioned on block maxima y_0. In this context, the variational lower bound can be formulated as follows:*

$$-\log p_\theta(\mathbf{x}_0, y_0) \leq \mathbb{E}q\Big[\log \frac{q(\mathbf{x}_{1:N}|\mathbf{x}_0, y_0)}{p_\theta(\mathbf{x}_{0:N})}\Big] - \log p_\theta(y_0) \tag{6.13}$$

Proof. The proof begins by expressing the negative log-likelihood of the joint distribution $-\log p_\theta(\mathbf{x}_0, y_0)$ in terms of conditional probabilities:

$$\begin{aligned}
-\log p_\theta(\mathbf{x}_0, y_0) &= -\log p_\theta(\mathbf{x}_0|y_0) \cdot p_\theta(y_0) = -\log p_\theta(\mathbf{x}_0|y_0) - \log p_\theta(y_0) \\
&\leq D_{\mathrm{KL}}(q(\mathbf{x}_{1:N}|\mathbf{x}_0, y_0)\|p_\theta(\mathbf{x}_{1:N}|\mathbf{x}_0, y_0)) \\
&\quad - \log p_\theta(\mathbf{x}_0|y_0) - \log p_\theta(y_0) \\
&= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_{1:N}|\mathbf{x}_0, y_0)}{p_\theta(\mathbf{x}_{0:N})} + \log p_\theta(\mathbf{x}_0, y_0)\Big] \\
&\quad - \log p_\theta(y_0) \\
&= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_{1:N}|\mathbf{x}_0, y_0)}{p_\theta(\mathbf{x}_{0:N})}\Big] - \log p_\theta(y_0)
\end{aligned} \tag{6.14}$$

□

First, we adopt $\overline{\mathbf{x}_0}$ as the reference for the unperturbed sample $\mathbf{x}_0$ as discussed in the previous subsection. After reparameterization and ignoring the weighting term, as suggested by [13], the first term of the variation lower bound can be expressed as:

$$\mathcal{L}_{\mathrm{DDPM}} = \mathbb{E}_{\overline{x_n}, \overline{\epsilon_n}, n, y_0} \|\widetilde{\epsilon_n}(\overline{\mathbf{x}_n}, n, y_0) - \overline{\epsilon_n}\|_2^2 \tag{6.15}$$

Additionally, considering a Generalized Extreme Value (GEV) distribution for block maxima, the second term is simplified as $\log \mathcal{L}_{\mathrm{GEV}}(\mu, \sigma, \xi)$, as defined in Eq. (3.4).

Conclusively, the remark establishes a clear link between the variational lower bound and an interpretable objective loss function:

$$-\log p_\theta(\mathbf{x}_0, y_0) \leq \mathcal{L}_{\mathrm{DDPM}} - \lambda \log \mathcal{L}_{\mathrm{GEV}}(\mu, \sigma, \xi) := L \tag{6.16}$$

Here, $\mathcal{L}_{\text{DDPM}}$ represents the expected reconstruction error between actual and estimated noise, and $\log \mathcal{L}_{\text{GEV}}(\mu, \sigma, \xi)$ captures the negative log-likelihood of the block maxima governed by a GEV distribution.

6.5.4 GEV Distribution Enforcement Module

To enforce fidelity to both the block maxima and overall data distribution, we incorporate the Generalized Extreme Value (GEV) distribution within the DDPM framework following Theorem 3. We first fit a GEV distribution using maximum log-likelihood estimation with all the block maxima (y_0) values in the training data. The fitted GEV distribution is parameterized by μ, σ, and ξ, denoted as $\theta_{\text{gev}} = \{\mu, \sigma, \xi\}$. Using the conditional diffusion process, the estimated noise is given by $\widetilde{\epsilon}_n(\overline{\mathbf{x}_n}, n, y_0)$. Consequently, the estimated denoised sample can be obtained as: $\widetilde{\mathbf{x}}_0 = \overline{\mathbf{x}_n} - \widetilde{\epsilon}_n$. Then, utilizing the fitted GEV distribution, the log-likelihood of the estimated denoised block maxima, $\widetilde{y}_0 = \max_{\tau \in \{1, \cdots, T\}} \widetilde{\mathbf{x}}_0^\tau$, is calculated. This negative log-likelihood, $-\log \mathcal{L}_{\text{GEV}}(\mu, \sigma, \xi, \widetilde{y}_0)$, is finally incorporated into the loss function of training.

6.5.5 Optimization

Algorithm 6.1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ where $\mathbf{x}_0 = (x_0^1, x_0^2, \cdots, x_0^T)$
  $\overline{\mathbf{f}_0} = \text{FFT}(\mathbf{x_0})$
  $\overline{\mathbf{f}_0} = \gamma \odot \mathbf{f_0}$
  $\overline{\mathbf{x}_0} = \text{IFFT}(\overline{\mathbf{f}_0})$
  $n \sim \text{Uniform}(\{1, \cdots, N\})$
  $\overline{\epsilon_n} \sim \mathcal{N}(0, \mathbf{I})$
  $y_0 = \max(\mathbf{x}_0)$
  $\overline{\mathbf{x}_n} = \sqrt{\alpha_n}\, \overline{\mathbf{x}_0} + \sqrt{1 - \alpha_n}\, \overline{\epsilon_n}$
  $\widetilde{\mathbf{y}}_0 = \max(\overline{\mathbf{x}_n} - \widetilde{\epsilon}_n(\overline{\mathbf{x}_n}, n, y_0))$
  Take the gradient step on
    $\nabla_\theta ||\widetilde{\epsilon}_n(\overline{\mathbf{x}_n}, n, y_0)) - \overline{\epsilon_n}||_2^2$
      $-\lambda \cdot \log \mathcal{L}_{\text{GEV}}(\mu, \sigma, \xi, \widetilde{y}_0)$
until converged

Algorithm 6.1 summarizes the pseudocode for training and Algorithm 6.2 summarizes the pseudocode of sampling. The overall loss function $\mathcal{L}_{\texttt{FIDE}}$ is constructed by combining two key

Algorithm 6.2 Sampling

Input: Block maxima $\hat{y}_0 \sim \text{GEV}(y_0)$ and Trained Model $\widetilde{\epsilon}$
Output: Generate time series, $\hat{\mathbf{x}}_0$.

$\hat{\mathbf{x}}_N \sim \mathcal{N}(0, \sigma^2)$
for $n = N, \cdots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
  $\hat{\mathbf{x}}_{n-1} = \frac{1}{\sqrt{\alpha_n}}(\hat{\mathbf{x}}_n - \frac{1-\alpha_n}{\sqrt{1-\alpha_n}}\widetilde{\epsilon}_n(\hat{\mathbf{x}}_n, n, \hat{y}_0)) + \sigma_n \mathbf{z}$
end for
return $\hat{\mathbf{x}}_0$

components: the DDPM loss $\mathcal{L}_{\text{DDPM}}$ and the negative log-likelihood of the Generalized Extreme Value (GEV) distribution $-\mathcal{L}_{\text{GEV}}$. The formulation is expressed as follows:

$$\mathcal{L}_{\texttt{FIDE}} = \mathbb{E}_{\overline{\mathbf{x}_n}, \overline{\epsilon_n}, n, y_0} \|\widetilde{\epsilon}_n(\overline{\mathbf{x}_n}, n, y_0) - \overline{\epsilon_n}\|_2^2 - \lambda \cdot \log \mathcal{L}_{\text{GEV}}(\mu, \sigma, \xi, \widetilde{y}_0) \tag{6.17}$$

where λ is a hyperparameter controlling the influence of the GEV distribution on the loss.

In this context, $\mathcal{L}$DDPM evaluates the mean squared difference between the estimated noise term $\widetilde{\epsilon}n$ and the true noise term ϵ_n within the conditional diffusion process. Its purpose is to guide the generative model towards effectively capturing the underlying data distribution. The second element, $-\log \mathcal{L}_{\text{GEV}}(\mu, \sigma, \xi, \widetilde{y}_0)$, encapsulates the negative log-likelihood of the GEV distribution. This component assesses how well the fitted GEV distribution aligns with the estimated block maxima values $\widetilde{y}_0$ derived from the denoised samples. Here, the log-likelihood is inverted to represent a minimization objective, aligning with the overall goal of minimizing the loss function.

6.6 Experimental Evaluation

We have performed extensive experiments to evaluate the performance of our `FIDE` framework. All the code and datasets used in this chapter are available at https://github.com/galib19/FIDE.

6.6.1 Data

We performed our experiments using the following datasets. **(1) Synthetic Data (AR2)**: AR(2) dataset comprises synthetic time series data generated using an autoregressive model of order 2. **(2) Financial Data (Stocks)**: It features continuous-valued and aperiodic sequences, such as daily historical Google stocks data spanning from 2004 to 2019. We consider the adjusted closing price

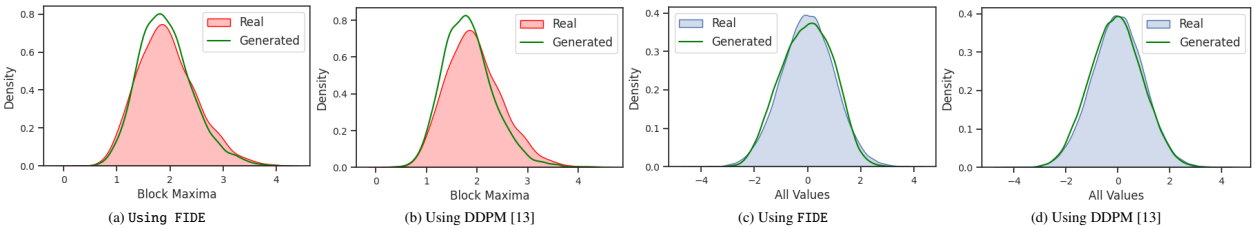
data for this work. **(3) Energy Data (Appliance Energy):** The UCI Appliances energy prediction dataset [101] encompasses multivariate, continuous-valued measurements. We consider appliance energy data for analysis. **(4) Weather/Climate Data (Daily Minimum Temperature)**: This dataset [102] comprises daily minimum temperatures in Melbourne, Australia, from 1981 to 1990. **(5) Medical Data (ECG5000: Congestive Heart Failure)**: The original dataset [103] for "ECG5000" originates from a 20-hour long electrocardiogram (ECG) obtained from the Physionet database. Specifically, it is derived from the BIDMC Congestive Heart Failure Database (chfdb), with the record labeled as "chf07." The processed data encompasses 5,000 heartbeats randomly selected from the original dataset.

6.6.2 Baseline Methods

We compared our proposed framework against various generative models: **(1) GAN-based:** We utilize two GAN-based approaches as our baselines. The first approach is Conditional GAN (cGAN [63]), which introduces conditional information to the training process, enabling targeted generation based on specified conditions. The second baseline is TimeGAN [7], which is a generative model designed specifically for time-series generation. **(2) VAE-based:** We employ beta-VAE [104], conditional beta-VAE [99], and TimeVAE [8] as baseline methods for comparison. Both beta-VAE and conditional beta-VAE incorporate a specific disentanglement objective to encourage the model to learn more interpretable and factorized representations while TimeVAE [8] promotes interpretability. **(3) Flow-based:** We use normalizing flows-based approaches such as RealNVP [105] and Fourier-Flows [106] as our baseline methods. **(4) Diffusion-based:** We consider two baselines for comparison, namely, the denoising diffusion probabilistic model (DDPM) [13] and a time series diffusion model called Diffusion-TS [9].

6.6.3 Experimental Settings

We partitioned each dataset into training, validation, and testing, according to a 8:1:1 ratio. We repeated the experiments 5 times. Prior to applying the various algorithms, the time series data is standardized to have zero mean and unit variance. The encoder component of our framework employs a 3-layer transformer architecture, accompanied by fully connected layers. The training was



(a) Using FIDE (b) Using DDPM [13] (c) Using FIDE (d) Using DDPM [13]

Figure 6.7 Comparison of block maxima distribution and all values distribution for real and generated samples using the proposed FIDE model and DDPM [13] when applied to the synthetic AR(1) dataset.

facilitated using the Adam optimizer. For all the methods, we perform extensive hyperparameter tuning on the length of the embedding vector, the number of hidden layers, the number of nodes, the learning rate, and the batch size. The optimal hyperparameters were determined using the Ray Tune framework, integrating an Asynchronous Successive Halving Algorithm (ASHA) scheduler to enable early stopping. All experiments were conducted on NVIDIA T4 GPU.

To assess the effectiveness of the proposed framework, we utilize four metrics: Jensen-Shannon Divergence, KL Divergence, CRPS (Continuous Rank Probability Score), and Predictive Score. The first three metrics are intricately linked to the analysis of distributions. Jensen-Shannon Divergence and KL Divergence are metrics measuring the difference between probability distributions. CRPS (Continuous Rank Probability Score) assesses the accuracy of predicted cumulative probabilities against observed outcomes. The fourth metric, the Predictive Score [7], was introduced with the purpose of evaluating the ability of a generative model to accurately replicate the temporal characteristics of the original data. This involves the deployment of a naive LSTM-based sequence model for time-series forecasting using synthesized samples. The performance of this predictive model is gauged by the mean absolute error (MAE) on the original test data, providing insight into the generative model's capacity to faithfully reproduce temporal properties. For our case, we evaluate the forecasting performance of block maxima on the test dataset by the model trained on generated data.

6.6.4 Experimental Results

Table 6.1 compares the performance of FIDE against several state-of-the-art baselines in terms of their ability to capture the block maxima distribution for 5 diverse datasets (AR1, Stock, Energy,

Metrics	Methods	AR1	Stock	Energy	Temperature	ECG
JS Divergence	beta-VAE	0.0211±0.0187	0.1105±0.0188	0.0722±0.0095	0.0140±0.0125	0.1210±0.0214
	c-beta-VAE	0.0190±0.0125	0.1011±0.0152	0.0710±0.0088	0.0109±0.0098	0.1120±0.0352
	TimeVAE	0.0015±0.0003	0.1054±0.0071	0.0795±0.0085	0.0096±0.0002	0.0985±0.0078
	TimeGAN	0.0840±0.0109	0.1411±0.1585	0.0950±0.0089	0.0112±0.0012	0.1620±0.0221
	cGAN	0.0690±0.0091	0.1211±0.0205	0.0890±0.0093	0.0091±0.0008	0.1440±0.0211
	RealNVP	0.0754±0.0121	0.1185±0.0108	0.0905±0.0084	0.0089±0.0007	0.1411±0.0116
	Fourier-Flows	0.0612±0.0045	0.1108±0.0195	0.0820±0.0044	0.0078±0.0010	0.1398±0.0202
	DDPM	0.0010±0.0007	0.0912±0.0062	0.0752±0.0082	0.0082±0.0009	0.1041±0.0122
	Diffusion-TS	0.0011±0.0008	0.0854±0.0045	0.0712±0.0071	0.0077±0.0008	0.1005±0.0108
	FIDE (Ours)	**0.0004±0.0001**	**0.0700±0.0061**	**0.0680±0.0092**	**0.0007±0.0001**	**0.0930±0.0082**
KL Divergence	beta-VAE	0.0110±0.0024	0.1947±0.0184	0.1210±0.0146	0.0410±0.0128	0.2020±0.0048
	c-beta-VAE	0.0091±0.0012	0.1744±0.0105	0.1160±0.0174	0.0360±0.0114	0.1880±0.0079
	TimeVAE	0.0105±0.0007	0.2514±0.0152	0.1625±0.0095	0.0490±0.0006	0.2254±0.0068
	TimeGAN	0.1920±0.0156	0.2425±0.0251	0.1590±0.0198	0.0550±0.0145	0.2540±0.0254
	cGAN	0.1240±0.0122	0.2101±0.0115	0.1510±0.0211	0.0490±0.0125	0.2210±0.0184
	RealNVP	0.1298±0.0215	0.2295±0.0154	0.1605±0.0310	0.0512±0.0108	0.2305±0.0145
	Fourier-Flows	0.1235±0.0104	0.2045±0.0255	0.1458±0.0345	0.0505±0.0136	0.2254±0.0141
	DDPM	0.0062±0.0008	0.1915±0.0125	0.1120±0.0108	0.0326±0.0090	0.1905±0.0094
	Diffusion-TS	0.0054±0.0007	0.1889±0.0108	0.1089±0.0115	0.0311±0.0078	0.1894±0.0081
	FIDE (Ours)	**0.0030±0.0009**	**0.1504±0.0128**	**0.0950±0.0098**	**0.0029±0.0008**	**0.1810±0.0084**
CRPS	beta-VAE	0.1247±0.0189	0.3149±0.0348	0.2410±0.0298	0.1554±0.0214	0.3059±0.0454
	c-beta-VAE	0.1154±0.0151	0.2698±0.0214	0.2574±0.0241	0.1420±0.0311	0.3150±0.0414
	TimeVAE	0.1511±0.0081	0.2547±0.0155	0.2853±0.1082	0.1847±0.0071	0.3252±0.0204
	TimeGAN	0.1858±0.0214	0.2825±0.0418	0.2685±0.0284	0.2110±0.0287	0.3240±0.0401
	cGAN	0.1224±0.0157	0.2689±0.0301	0.2385±0.0187	0.1990±0.0214	0.2985±0.0311
	RealNVP	0.1325±0.0144	0.2545±0.0258	0.2541±0.0214	0.2014±0.0354	0.2824±0.0425
	Fourier-Flows	0.1305±0.0254	0.2589±0.0214	0.2415±0.0211	0.1975±0.0251	0.2884±0.0215
	DDPM	0.0422±0.0084	0.2422±0.0187	0.2199±0.0874	0.1516±0.0211	0.2488±0.0388
	Diffusion-TS	0.0398±0.0092	0.2358±0.0211	0.2125±0.0454	0.1525±0.0315	0.2415±0.0451
	FIDE (Ours)	**0.0310±0.0098**	**0.2115±0.0152**	**0.2085±0.0985**	**0.0517±0.0082**	**0.2345±0.0204**
Predictive Score	beta-VAE	0.6350±0.0201	0.9528±0.0314	0.7410±0.0187	0.6814±0.0108	0.9420±0.0142
	c-beta-VAE	0.6240±0.0145	0.9226±0.0165	0.7317±0.0163	0.6718±0.0025	0.9310±0.0214
	TimeVAE	0.6150±0.0104	0.9140±0.0218	0.7325±0.0195	0.6723±0.0036	0.9150±0.0112
	TimeGAN	**0.6050±0.0104**	0.8950±0.0198	0.7280±0.0187	0.6718±0.0047	**0.8960±0.0084**
	cGAN	0.6120±0.0014	0.9354±0.0210	0.7310±0.0147	0.6847±0.0041	0.9220±0.0191
	RealNVP	0.6884±0.0011	0.9988±0.0354	0.7898±0.0254	0.7852±0.0017	0.9730±0.0215
	Fourier-Flows	0.6925±0.0031	0.9844±0.0241	0.7955±0.0088	0.7871±0.0021	0.9655±0.0221
	DDPM	0.6148±0.0081	0.8997±0.0111	0.7350±0.0102	0.6708±0.0098	0.9121±0.0121
	Diffusion-TS	0.6105±0.0045	0.8912±0.0105	0.7355±0.0084	0.6708±0.0108	0.9089±0.0095
	FIDE (Ours)	0.6081±0.0098	**0.8871±0.0104**	**0.7240±0.0087**	**0.6694±0.0082**	0.9040±0.0112

Table 6.1 Comparison of generated samples' block maxima distribution metrics and predictive score. **Bold** and Underlined entries denote the best and second-best result.

Temperature, and ECG). In terms of the 3 distribution metrics (JS divergence, KL divergence, and CRPS), FIDE consistently achieves the best results, providing evidence of FIDE's superior performance in preserving the block maxima distribution. For the Predictive Score metric, FIDE achieves the best results in 3 out of 5 datasets and ranks second in the remaining 2 datasets. The

Metrics	Methods	AR1	Stock	Energy	Temperature	ECG
JS Divergence	beta-VAE	0.0013±0.0004	0.0091±0.0011	0.0091±0.0008	0.0012±0.0002	0.0015±0.0003
	c-beta-VAE	0.0011±0.0004	0.0088±0.0012	0.0089±0.0007	0.0010±0.0001	0.0014±0.0002
	TimeVAE	0.0010±0.0003	0.0086±0.0008	0.0081±0.0011	0.0009±0.0001	0.0014±0.0001
	TimeGAN	0.0015±0.0004	0.0092±0.0015	0.0082±0.0009	0.0011±0.0003	0.0018±0.0003
	cGAN	0.0012±0.0002	0.0091±0.0010	0.0085±0.0008	0.0008±0.0001	0.0015±0.0002
	RealNVP	0.0014±0.0003	0.0093±0.0011	0.0092±0.0008	0.0010±0.0002	0.0017±0.0001
	Fourier-Flows	0.0013±0.0004	0.0087±0.0012	0.0083±0.0007	0.0009±0.0001	0.0015±0.0002
	DDPM	**0.0007±0.0001**	0.0061±0.0007	0.0058±0.0005	**0.0005±0.0001**	0.0010±0.0001
	Diffusion-TS	0.0008±0.0001	**0.0057±0.0008**	0.0061±0.0005	0.0008±0.0001	**0.0009±0.0001**
	FIDE (Ours)	0.0008±0.0001	0.0068±0.0010	**0.0056±0.0006**	0.0006±0.0002	0.0011±0.0001
KL Divergence	beta-VAE	0.0020±0.0003	0.0188±0.0016	0.0181±0.0015	0.0025±0.0003	0.0031±0.0004
	c-beta-VAE	0.0017±0.0004	0.0178±0.0019	0.0177±0.0017	0.0022±0.0004	0.0028±0.0004
	TimeVAE	0.0016±0.0003	0.0169±0.0015	0.0159±0.0021	0.0018±0.0002	0.0026±0.0003
	TimeGAN	0.0025±0.0003	0.0182±0.0025	0.0161±0.0016	0.0021±0.0005	0.0034±0.0005
	cGAN	0.0018±0.0003	0.0178±0.0018	0.0169±0.0016	0.0015±0.0003	0.0029±0.0003
	RealNVP	0.0023±0.0004	0.0185±0.0019	0.0185±0.0017	0.0019±0.0004	0.0036±0.0002
	Fourier-Flows	0.0019±0.0003	0.0173±0.0021	0.0165±0.0015	0.0017±0.0003	0.0028±0.0003
	DDPM	**0.0010±0.0001**	0.0117±0.0011	0.0114±0.0010	**0.0009±0.0001**	**0.0019±0.0003**
	Diffusion-TS	0.0011±0.0001	**0.0114±0.0012**	0.0116±0.0009	0.0010±0.0002	**0.0019±0.0003**
	FIDE (Ours)	0.0012±0.0001	0.0121±0.0015	**0.0109±0.0009**	0.0011±0.0002	0.0021±0.0004
CRPS	beta-VAE	0.0201±0.0041	0.4955±0.0125	0.4985±0.0102	0.0914±0.0010	0.1425±0.0049
	c-beta-VAE	0.1984±0.0022	0.4205±0.0148	0.4514±0.0210	0.0899±0.0009	0.1388±0.0068
	TimeVAE	0.1848±0.0038	0.4841±0.085	0.4815±0.0189	0.0889±0.0009	0.1422±0.0077
	TimeGAN	0.2412±0.0019	0.3941±0.0115	0.4415±0.0171	0.0911±0.0008	0.1262±0.0062
	cGAN	0.1974±0.0012	0.4451±0.0201	0.3914±0.0211	0.0903±0.0007	0.1298±0.0056
	RealNVP	0.2511±0.0019	0.4254±0.0194	0.5125±0.0184	0.0919±0.0007	0.1405±0.0035
	Fourier-Flows	0.2214±0.0024	0.3814±0.0164	0.4514±0.0123	0.0912±0.0008	0.1281±0.0077
	DDPM	0.1595±0.0018	**0.2955±0.0144**	**0.3215±0.0154**	0.0875±0.0006	0.1028±0.0062
	Diffusion-TS	0.1565±0.0016	0.2985±0.0174	0.3285±0.0149	**0.0863±0.0007**	**0.1018±0.0045**
	FIDE (Ours)	**0.1541±0.0021**	0.3001±0.0191	0.3251±0.0177	0.0893±0.0007	0.1061±0.0054
Predictive Score	beta-VAE	0.8121±0.0410	1.0915±0.0215	0.8515±0.0104	0.8021±0.0109	0.9911±0.0133
	c-beta-VAE	0.7951±0.0555	1.0841±0.0121	0.8442±0.0110	0.7958±0.0089	0.9891±0.0151
	TimeVAE	0.7714±0.0345	1.0662±0.0211	0.8394±0.0089	**0.7821±0.0105**	**0.9822±0.0101**
	TimeGAN	**0.7514±0.0451**	**1.0621±0.0198**	**0.8379±0.0151**	0.7856±0.0098	0.9862±0.0125
	cGAN	0.7694±0.0354	1.0721±0.0188	0.8433±0.0181	0.7905±0.0122	0.9874±0.0151
	RealNVP	0.8011±0.0384	1.0914±0.0178	0.8533±0.0154	0.8033±0.0135	0.9981±0.0201
	Fourier-Flows	0.7985±0.0324	1.1008±0.0205	0.8501±0.0151	0.7988±0.0140	0.9954±0.0188
	DDPM	0.7711±0.0441	1.0751±0.0184	0.8488±0.0133	0.7912±0.0125	0.9925±0.0167
	Diffusion-TS	0.7684±0.0405	1.0722±0.0189	0.8501±0.0125	0.7889±0.0129	0.9910±0.0155
	FIDE (Ours)	0.7651±0.0488	1.0692±0.0192	0.8458±0.0151	0.7895±0.0135	0.9852±0.0158

Table 6.2 Comparison of the generated sample distribution for the distribution metrics and predictive score. **Bold** and Underlined entries denote the best and second-best result.

distribution plots for the synthetic AR(1) data generated by DDPM [13] and FIDE, as shown in Figure 6.7-(a) and (b), provide further evidence of FIDE's capabilities. While DDPM struggles to capture the block maxima distribution accurately, FIDE generates samples that more effectively maintain the fidelity of the distribution. This improvement is particularly noticeable in the upper tail

behavior, which is critical for applications that require precise modeling of extreme block maxima. This superior performance is not surprising as it directly results from our method's emphasis on block maxima distribution, achieved through the introduction of frequency inflation, conditional generation based on block maxima, and the incorporation of the GEV distribution into the generative modeling framework.

As FIDE prioritizes the accurate modeling of block maxima, we have also evaluated its efficacy in capturing the distribution of all (block maxima and non-block maxima) values in time series. The results are shown in Table 6.2. FIDE achieves comparable performance to state-of-the-art methods like DDPM [13] and Diffusion-TS [9]. This is further illustrated by the distribution plots of all values for DDPM and FIDE given in Figure 6.7-(c) and (d). The results in Table 6.2 also show that FIDE consistently outperforms VAE-based, GAN-based, and Flow-based alternatives. For Predictive Score, while TimeGAN and TimeVAE show marginally better results, FIDE maintains competitive performance against other baseline methods. These results suggest minimal performance trade-off when applying FIDE to time series data. Despite its emphasis on block maxima values, this does not significantly compromise its ability to model the overall distribution. This positions FIDE as a robust and versatile generative model for capturing extreme values in time series.

6.6.5 Ablation Study

In our ablation study depicted in Table 6.3, we systematically assessed the individual contributions of each component within our proposed framework. By selectively deactivating elements such as the GEV loss, conditional block maxima input, and high-frequency inflation module, we observed consistent performance degradation across all scenarios. Notably, the absence of the conditional block maxima input significantly impacted the Jenson-Shannon Divergence and KL Divergence metrics, while the lack of the GEV loss had the most pronounced effect on the CRPS metric. Surprisingly, the predictive score remained relatively resilient to the deactivation of any single component, suggesting a degree of redundancy or compensatory mechanisms among the remaining components. Overall, our ablation study highlights the indispensable role of each component in achieving optimal performance in our model. In summary, our findings underscore the holistic

Metrics	Methods	AR1	Stock	Energy	Temperature	ECG
Jensen–Shannon Divergence	FIDE - frequency inflation	0.0006+0.0001	0.0898+0.0054	0.0822+0.0084	0.0009+0.0001	0.0984+0.0058
	FIDE - conditional	0.0007+0.0001	0.1054+0.0089	0.0941+0.0098	0.0010+0.0002	0.1102+0.0098
	FIDE - GEV Loss	0.0005+0.0001	0.0813+0.0035	0.0715+0.0041	0.0008+0.0001	0.0922+0.0056
	FIDE	**0.0004+0.0001**	**0.0700+0.0061**	**0.0680+0.0092**	**0.0007+0.0001**	**0.0930+0.0082**
KL Divergence	FIDE - frequency inflation	0.0042+0.0008	0.1559+0.0161	0.1054+0.0049	0.0036+0.0006	0.1854+0.0064
	FIDE - conditional	0.0051+0.0010	0.1689+0.0210	0.1089+0.0095	0.0041+0.0010	0.1901+0.0063
	FIDE - GEV Loss	0.0039+0.0007	0.1551+0.0188	0.1021+0.0088	0.0032+0.0009	0.1823+0.0092
	FIDE	**0.0030+0.0009**	**0.1504+0.0128**	**0.0950+0.0098**	**0.0029+0.0008**	**0.1810+0.0084**
CRPS	FIDE - frequency inflation	0.0391+0.0078	0.2172+0.0158	0.2152+0.0791	0.0649+0.0081	0.2372+0.0181
	FIDE - conditional	0.0335+0.0089	0.2165+0.0132	**0.2082+0.0768**	0.0651+0.0047	0.2382+0.0184
	FIDE - GEV Loss	0.0415+0.0087	0.2232+0.0203	0.2189+0.0874	0.0815+0.0104	0.2456+0.0399
	FIDE	**0.0310+0.0098**	**0.2115+0.0152**	0.2085+0.0985	**0.0517+0.0082**	**0.2345+0.0204**
Predictive Score	FIDE - frequency inflation	**0.6070+0.0112**	0.8942+0.0158	0.7264+0.0069	0.6711+0.0091	0.9081+0.0154
	FIDE - conditional	0.6095+0.0079	0.8901+0.0141	0.7261+0.0081	0.6715+0.0078	0.9059+0.0122
	FIDE - GEV Loss	0.6089+0.0089	0.8891+0.0122	0.7269+0.0074	0.6712+0.0009	0.9062+0.0058
	FIDE	0.6081+0.0098	**0.8871+0.0104**	**0.7240+0.0087**	**0.6694+0.0082**	**0.9040+0.0112**

Table 6.3 Ablation Study of generated samples' block maxima distribution metrics and predictive score using the proposed FIDE model and without individual component of the model.

importance of the individual components, with their synergistic interplay contributing to the overall effectiveness of FIDE.

6.7 Conclusions

This framework entails a comprehensive strategy that involves refining the model's ability to capture and emphasize extreme event distributions. Through a comprehensive exploration of the constraints within current diffusion-based models, the introduced FIDE framework innovatively addresses these limitations by introducing strategies to maintain high-frequency components. Furthermore, it extends conventional diffusion models to enable conditional generation while integrating the Generalized Extreme Value (GEV) distribution. One limitation of the framework is that the theoretical justifications are based on some mild assumptions. However, we have justified those assumptions with empirical evidence and prior literature. The framework's superiority over baseline methods is confirmed through rigorous experiments on both real-world and synthetic data.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

The proposed thesis explores the subject of modeling extremes in time series, with a focus on predictive and generative modeling techniques. The initial three works of the thesis aim to explore and develop predictive modeling techniques for time series extremes, with the goal of accurate forecasting and identifying extreme events in time series data. In the final work of the thesis, attention shifts towards exploring conditional generative modeling for time series extremes, specifically diffusion-based conditional generative modeling.

This dissertation presents significant advancements in the field of time series modeling by developing deep learning frameworks that improve the prediction and generative modeling of extreme values. These contributions provide robust solutions to the challenges posed by extreme events in time series data, with applications spanning risk management, disaster preparedness, and various other domains. The methodologies introduced herein have the potential to advance both theoretical understanding and practical applications in time series forecasting and generative modeling. Here are the key contributions of this thesis:

DeepExtrema: Introduced in Chapter 3, this framework combines a deep neural network with a generalized extreme value (GEV) distribution to forecast block maxima in time series data. This approach addresses the challenge of maintaining positivity constraints on GEV parameters and demonstrates superior performance through extensive experiments on real-world and synthetic data.

Self-Recover: Presented in Chapter 4, this self-supervised learning framework addresses the challenge of disparate temporal coverage in time series predictors. It employs a combination of contrastive and generative self-supervised schemes to impute long-term missing values and a denoising autoencoder for random missing values, effectively enhancing block maxima predictions.

Self-Recover: Detailed in Chapter 5, this self-supervised learning framework focuses on robust representation learning of time series data to capture tail distributions. By integrating wavelet-based data augmentation and a loss function based on empirical cumulative distribution functions, `Self-Recover` improves the forecasting of extreme values.

FIDE: Introduced in Chapter 6, FIDE is a conditional diffusion model that addresses the generative modeling of extreme values in time series. By employing a high-frequency inflation strategy and incorporating the GEV distribution, FIDE enhances the time series generation focusing on extreme events, demonstrating its efficacy through experiments on both real-world and synthetic data.

7.1 Future Work

The application of deep learning to time series extremes presents numerous avenues for further research. Two broad areas warrant particular attention. First, extending research into modeling spatio-temporal relationships could significantly enhance the accuracy and comprehensiveness of forecasting models by capturing how extreme events evolve and interact across both space and time. This is especially relevant in fields like meteorology, environmental science, and public health. Second, improving the interpretability of models predicting time series extremes is crucial for gaining the trust of stakeholders and enabling informed decision-making. Enhancing model transparency and explainability through new visualization techniques and inherently interpretable model designs will facilitate a deeper understanding of the factors driving extreme events and improve the communication of these insights to decision-makers.

7.1.1 Spatio-Temporal Modeling of Extreme Events

A promising direction for future research is the development of frameworks that can effectively model the spatio-temporal dependencies and dynamics of extreme events. Extreme events, such as natural disasters, often exhibit complex spatial and temporal patterns, with their occurrences influenced by various environmental factors and their impacts propagating across different geographic regions over time. Capturing these intricate spatio-temporal relationships is essential for accurate forecasting and risk assessment.

Potential avenues for exploration include the integration of deep learning techniques with spatio-temporal models, such as convolutional LSTMs, graph neural networks, or transformer architectures. These approaches could leverage the ability of deep learning to learn complex representations from data, while explicitly accounting for spatial and temporal dependencies through the model structure. Additionally, incorporating domain knowledge and physics-based models

into the deep learning frameworks could further enhance their capability to capture the underlying mechanisms driving extreme events.

Furthermore, the development of multivariate and multi-task learning frameworks could enable the joint modeling of multiple extreme event types and their interactions. Such approaches could provide a more comprehensive understanding of the complex dynamics at play and facilitate more holistic risk assessment and decision-making.

There are some potential challenges in this direction as follows:

- Handling high-dimensional and multi-scale data: Extreme events often involve complex interactions across multiple spatial and temporal scales, requiring models to effectively process and integrate high-dimensional data from various sources.
- Accounting for non-stationarities and regime shifts: Many extreme events exhibit non-stationary behavior, with their patterns and drivers subject to abrupt changes or regime shifts over time, posing challenges for traditional spatio-temporal models.
- Incorporating physical constraints and domain knowledge: Integrating domain-specific knowledge, physical laws, and constraints into deep learning models can improve their ability to capture realistic spatio-temporal dynamics, but doing so in a principled and effective manner remains a challenge.
- Scalability and computational efficiency: Modeling intricate spatio-temporal dependencies across large geographic regions and extended time horizons can be computationally intensive, necessitating the development of scalable and efficient architectures.

Overcoming these challenges would enable the development of comprehensive spatio-temporal models that can accurately capture the complex interplay between extreme events and their environmental drivers across different spatial and temporal scales. Such models would have far-reaching applications in fields like meteorology, climate science, environmental monitoring, and disaster risk management. By providing accurate forecasts and insights into the spatial evolution and propagation of extreme events, these models could inform proactive mitigation strategies, resource

allocation decisions, and targeted interventions to minimize the adverse impacts on communities and ecosystems.

7.1.2 Interpretability and Explainability in Extreme Event Prediction

While deep learning models presented in this thesis have demonstrated remarkable predictive performance in modeling extreme events, their inherent complexity often hinders interpretability and explainability. Improving the transparency and interpretability of these models is crucial for gaining stakeholder trust, enabling effective communication with decision-makers, and facilitating a deeper understanding of the underlying factors driving extreme events.

One promising direction is the development of inherently interpretable deep learning architectures, such as attention-based models, disentangled representations, or prototype-based models. These approaches could provide insights into the specific features or patterns that contribute to the prediction of extreme events, enabling better interpretability.

Another avenue for exploration is the integration of post-hoc explanation techniques, such as saliency maps, concept activation vectors, or counterfactual explanations. These methods aim to provide human-understandable explanations for model predictions by highlighting the most relevant input features or by generating counterfactual examples that elucidate the decision boundaries.

Furthermore, the development of novel visualization techniques tailored for extreme event predictions could significantly enhance the communication and interpretation of model outputs. Interactive visualizations and dashboards that effectively convey the spatial and temporal characteristics of predicted extreme events, along with their associated uncertainties, could facilitate more informed decision-making by stakeholders.

Potential challenges in this area include:

- Quantifying and communicating uncertainty: Extreme event predictions are inherently uncertain, and effectively quantifying and communicating this uncertainty to stakeholders in an interpretable manner is crucial but challenging.
- Maintaining interpretability in complex models: As models become more sophisticated (e.g., incorporating spatio-temporal dependencies or multi-task learning), preserving interpretability

becomes increasingly challenging, requiring novel approaches to disentangle and explain the model's decision-making process.

- Evaluation of interpretability and explainability: Developing rigorous metrics and evaluation frameworks for assessing the quality and effectiveness of interpretability and explainability methods remains an open challenge.
- Balancing accuracy and interpretability: There is often a trade-off between model complexity (which improves accuracy) and interpretability, requiring careful model design and regularization techniques to strike an appropriate balance.

By addressing the interpretability and explainability challenges, these future research directions could contribute to building more trustworthy and actionable models for extreme event prediction, ultimately enabling better risk assessment, mitigation strategies, and decision-making processes.

BIBLIOGRAPHY

[1] Chenglei Peng, Yang Li, Yao Yu, Yu Zhou, and Sidan Du. Multi-step-ahead host load prediction with gru based encoder-decoder in cloud computing. In *2018 10th International Conference on Knowledge and Smart Technology (KST)*, pages 186–191. IEEE, 2018.

[2] Alaa Sagheer and Mostafa Kotb. Time series forecasting of petroleum production using deep lstm recurrent networks. *Neurocomputing*, 323:203–213, 2019.

[3] Shamsul Masum, Ying Liu, and John Chiverton. Multi-step time series forecasting of electric load using machine learning models. In *International conference on artificial intelligence and soft computing*, pages 148–159. Springer, 2018.

[4] Nikolay Laptev, Jason Yosinski, Li Erran Li, and Slawek Smyl. Time-series extreme event forecasting with neural networks at uber. In *International conference on machine learning*, volume 34, pages 1–5, 2017.

[5] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[6] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

[7] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32, 2019.

[8] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation. *arXiv preprint arXiv:2111.08095*, 2021.

[9] Xinyu Yuan and Yan Qiao. Diffusion-TS: Interpretable Diffusion for General Time Series Generation. *arXiv preprint arXiv:2403.01742*, 2024.

[10] Stuart Coles, Joanna Bawa, Lesley Trenner, and Pat Dorazio. *An introduction to Statistical Modeling of Extreme Values*, volume 208. Springer, 2001.

[11] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation. *Advances in Neural Information Processing Systems*, 34:24804–24816, 2021.

[12] Marin Biloš, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, and Stephan Günnemann. Modeling Temporal Data as Continuous Functions with Stochastic Process Diffusion. In *International Conference on Machine Learning*, pages 2452–2470. PMLR, 2023.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

[16] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

[17] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[18] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, 2017.

[19] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv:1803.01271 [cs]*, April 2018. arXiv: 1803.01271.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[21] Hao Li, Yanyan Shen, and Yanmin Zhu. Stock price prediction using attention-based multi-input lstm. In *Asian conference on machine learning*, pages 454–469. PMLR, 2018.

[22] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021.

[23] Majid Moradi Aliabadi, Hajar Emami, Ming Dong, and Yinlun Huang. Attention-based recurrent neural network for multistep-ahead prediction of process performance. *Computers & Chemical Engineering*, 140:106931, 2020.

[24] Xuan Zhang, Xun Liang, Aakas Zhiyuli, Shusen Zhang, Rui Xu, and Bo Wu. At-lstm: An attention-based lstm model for financial time series prediction. In *IOP Conference Series: Materials Science and Engineering*, volume 569, page 052037. IOP Publishing, 2019.

[25] Manuel R Vargas, Beatriz SLP De Lima, and Alexandre G Evsukoff. Deep learning for stock market prediction from financial news articles. In *2017 IEEE international conference on computational intelligence and virtual environments for measurement systems and applications (CIVEMSA)*, pages 60–65. IEEE, 2017.

[26] Mohammed F Alsharekh, Shabana Habib, Deshinta Arrova Dewi, Waleed Albattah, Muhammad Islam, and Saleh Albahli. Improving the efficiency of multistep short-term electricity load forecasting via r-cnn with ml-lstm. *Sensors*, 22(18):6913, 2022.

[27] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[28] Pradeep Hewage, Ardhendu Behera, Marcello Trovati, Ella Pereira, Morteza Ghahremani, Francesco Palmieri, and Yonghuai Liu. Temporal convolutional neural (tcn) network for an effective weather forecasting using time-series data from the local weather station. *Soft Computing*, 24:16453–16482, 2020.

[29] Daniel J Gauthier, Erik Bollt, Aaron Griffith, and Wendson AS Barbosa. Next generation reservoir computing. *Nature communications*, 12(1):5564, 2021.

[30] Francis Wyffels and Benjamin Schrauwen. A comparative study of reservoir computing strategies for monthly time series prediction. *Neurocomputing*, 73(10-12):1958–1964, 2010.

[31] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. Spectral temporal graph neural network for multivariate time-series forecasting. *Advances in neural information processing systems*, 33:17766–17778, 2020.

[32] Bin Wang, Tianrui Li, Zheng Yan, Guangquan Zhang, and Jie Lu. Deeppipe: A distribution-free uncertainty quantification approach for time series forecasting. *Neurocomputing*, 397:11–19, 2020.

[33] Viatcheslav V Kharin and Francis W Zwiers. Estimating extremes in transient climate change simulations. *Journal of Climate*, 18(8):1156–1173, 2005.

[34] Daizong Ding, Mi Zhang, Xudong Pan, Min Yang, and Xiangnan He. Modeling extreme events in time series prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1114–1122, 2019.

[35] Michael Polson and Vadim Sokolov. Deep learning for energy markets. *Applied Stochastic Models in Business and Industry*, 36(1):195–209, 2020.

[36] Andrew McDonald, Pang-Ning Tan, and Lifeng Luo. Comet flows: Towards generative modeling of multivariate extremes and tail dependence. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3328–3334. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.

[37] Tyler Wilson, Andrew McDonald, Asadullah Hill Galib, Pang-Ning Tan, and Lifeng Luo. Beyond point prediction: Capturing zero-inflated & heavy-tailed spatiotemporal data with deep extreme mixture models. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2020–2028, 2022.

[38] Tyler Wilson, Pang-Ning Tan, and Lifeng Luo. DeepGPD: A Deep Learning Approach for Modeling Geospatio-Temporal Extreme Events. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, 2022.

[39] Nenad Tomasev, Ioana Bica, Brian McWilliams, Lars Buesing, Razvan Pascanu, Charles Blundell, and Jovana Mitrovic. Pushing the limits of self-supervised resnets: Can we outperform supervised learning without labels on imagenet? *arXiv preprint arXiv:2201.05119*, 2022.

[40] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *Proceedings of the ieee/cvf International Conference on computer vision*, pages 6391–6400, 2019.

[41] Jeff Z HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *Advances in Neural Information Processing Systems*, 34:5000–5011, 2021.

[42] Xinrui Lyu, Matthias Hueser, Stephanie L Hyland, George Zerveas, and Gunnar Rätsch. Improving clinical predictions through unsupervised time series representation learning. *arXiv preprint arXiv:1812.00490*, 2018.

[43] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*, 2019.

[44] Yao-Hung Hubert Tsai, Yue Wu, Ruslan Salakhutdinov, and Louis-Philippe Morency. Self-supervised learning from a multi-view perspective. *arXiv preprint arXiv:2006.05576*, 2020.

[45] Zixin Wen and Yuanzhi Li. Toward understanding the feature learning process of self-supervised contrastive learning. In *International Conference on Machine Learning*, pages 11112–11122. PMLR, 2021.

[46] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *International Conference on Machine Learning*, pages 10268–10278. PMLR, 2021.

[47] Jason D Lee, Qi Lei, Nikunj Saunshi, and Jiacheng Zhuo. Predicting what you already know helps: Provable self-supervised learning. *Advances in Neural Information Processing Systems*, 34:309–323, 2021.

[48] Alaa Sagheer and Mostafa Kotb. Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems. *Scientific reports*, 9(1):1–16, 2019.

[49] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[50] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[51] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8980–8987, 2022.

[52] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. *arXiv preprint arXiv:2202.01575*, 2022.

[53] Xiang Zhang, Ziyuan Zhao, Theodoros Tsiligkaridis, and Marinka Zitnik. Self-supervised contrastive pre-training for time series via time-frequency consistency. *arXiv preprint arXiv:2206.08496*, 2022.

[54] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. *arXiv preprint arXiv:2106.14112*, 2021.

[55] Xinyu Yang, Zhenguo Zhang, and Rongyi Cui. Timeclr: A self-supervised contrastive learning framework for univariate time series representation. *Knowledge-Based Systems*, 245:108606, 2022.

[56] Sven Festag, Joachim Denzler, and Cord Spreckelsen. Generative adversarial networks for biomedical time series forecasting and imputation a systematic review. *Journal of Biomedical Informatics*, page 104058, 2022.

[57] Duc Hoang Tran, Van Linh Nguyen, Huy Nguyen, and Yeong Min Jang. Self-supervised learning for time-series anomaly detection in industrial internet of things. *Electronics*, 11(14):2146, 2022.

[58] Mingda Zhang. Time series: Autoregressive models ar, ma, arma, arima. *University of Pittsburgh*, 2018.

[59] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[60] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[61] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[62] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[63] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

[64] Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. GP-VAE: Deep Probabilistic Time Series Imputation. In *International conference on artificial intelligence and statistics*, pages 1651–1661. PMLR, 2020.

[65] Andrew McDonald, Pang-Ning Tan, and Lifeng Luo. COMET Flows: Towards Generative Modeling of Multivariate Extremes and Tail Dependence. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3328–3334, 2022.

[66] Magnus Wiese, Robert Knobloch, and Ralf Korn. Copula & marginal flows: Disentangling the marginal from its joint. *arXiv preprint arXiv:1907.03361*, 2019.

[67] Priyank Jaini, Ivan Kobyzev, Yaoliang Yu, and Marcus Brubaker. Tails of lipschitz triangular flows. In *International Conference on Machine Learning*, pages 4673–4681. PMLR, 2020.

[68] V Chandola, A Banerjee, and V Kumar. Anomaly detection: A survey. acm computing surveys. *vol*, 41:15, 2009.

[69] Tung Kieu, Bin Yang, Chenjuan Guo, and Christian S Jensen. Outlier detection for time series with recurrent autoencoder ensembles. In *IJCAI*, pages 2725–2732, 2019.

[70] Shuyu Lin, Ronald Clark, Robert Birke, Sandro Schönborn, Niki Trigoni, and Stephen Roberts. Anomaly detection for time series using vae-lstm hybrid model. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4322–4326. Ieee, 2020.

[71] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, Puneet Agarwal, et al. Long short term memory networks for anomaly detection in time series. In *ESANN*, volume 2015, page 89, 2015.

[72] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*, 2018.

[73] Yang Jiao, Kai Yang, Dongjing Song, and Dacheng Tao. Timeautoad: Autonomous anomaly detection with self-supervised contrastive loss for multivariate time series. *IEEE Transactions on Network Science and Engineering*, 9(3):1604–1619, 2022.

[74] US GAO. Natural disasters: Economic effects of hurricanes katrina, sandy, harvey, and irma. https://www.gao.gov/products/gao-20-633r, 2020. Accessed: 2021-04-09.

[75] Ding Wang and Pang-Ning Tan. Johan: A joint online hurricane trajectory and intensity forecasting framework. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1677–1685, 2021.

[76] Christopher M Bishop. *Pattern recognition*. Springer, New York, 2006.

[77] Richard L. Smith. Maximum likelihood estimation in a class of nonregular cases. *Biometrika*, 72(1):67–90, 1985.

[78] Christopher W Landsea and James L Franklin. Atlantic hurricane database uncertainty and presentation of a new database format. *Monthly Weather Review*, 141(10):3576–3592, 2013.

[79] Elizabeth L Ratnam, Steven R Weller, Christopher M Kellett, and Alan T Murray. Residential load and rooftop pv generation: an australian distribution network dataset. *International Journal of Sustainable Energy*, 36(8):787–806, 2017.

[80] J Muthukumar. Weather dataset. https://www.kaggle.com/muthuj7/weather-dataset, Dec 2017. Accessed: 2021-09-30.

[81] Asadullah Hill Galib, Andrew McDonald, Tyler Wilson, Lifeng Luo, and Pang-Ning Tan. DeepExtrema: A Deep Learning Approach for Forecasting Block Maxima in Time Series Data. *arXiv preprint arXiv:2205.02441*, 2022.

[82] Julie A Winkler, Galina S Guentchev, Malgorzata Liszewska, and Pang-Ning Tan. Climate scenario development and applications for local/regional climate change impact assessments: An overview for the non-climate scientist: Part ii: Considerations when using climate change scenarios. *Geography Compass*, 5(6):301–328, 2011.

[83] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12, 2018.

[84] Linus Ericsson, Henry Gouk, Chen Change Loy, and Timothy M Hospedales. Self-supervised representation learning: Introduction, advances, and challenges. *IEEE Signal Processing Magazine*, 39(3):42–62, 2022.

[85] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

[86] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[87] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[88] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[89] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[90] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020.

[91] Gordon J Ross. Modelling financial volatility in the presence of abrupt changes. *Physica A: Statistical Mechanics and its Applications*, 392(2):350–360, 2013.

[92] Dilip Kumar. Sudden changes in extreme value volatility estimator: Modeling and forecasting with economic significance analysis. *Economic Modelling*, 49:354–371, 2015.

[93] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. *arXiv preprint arXiv:2002.12478*, 2020.

[94] Brian Kenji Iwana and Seiichi Uchida. An empirical survey of data augmentation for time series classification with neural networks. *Plos one*, 16(7):e0254841, 2021.

[95] Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[96] HARALD CRAMAR. On the composition of elementary errors. *Skand. Aktuarietids*, 11:13–74, 1928.

[97] CD Broad. Wahrscheinlichkeit, statistik, und wahrheit., 1937.

[98] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. Unsupervised representation learning for time series with temporal neighborhood coding. *arXiv preprint arXiv:2106.00750*, 2021.

[99] Alexander Nikitin, Letizia Iannucci, and Samuel Kaski. Tsgm: A flexible framework for generative modeling of synthetic time series. *arXiv preprint arXiv:2305.11567*, 2023.

[100] Jonathan Crabbé, Nicolas Huynh, Jan Stanczuk, and Mihaela van der Schaar. Time series diffusion in the frequency domain. *arXiv preprint arXiv:2402.05933*, 2024.

[101] Luis Candanedo. Appliances Energy Prediction. UCI Machine Learning Repository, 2017. DOI: https://doi.org/10.24432/C5VC8G.

[102] Australian Bureau of Meteorology, 2016.

[103] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[104] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2016.

[105] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density Estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

[106] Ahmed Alaa, Alex James Chan, and Mihaela van der Schaar. Generative Time-series Modeling with Fourier Flows. In *International Conference on Learning Representations*, 2021.