DEEP LEARNING REGULARIZATION: THEORY AND DATA PERSPECTIVES

By

Xitong Zhang

A DISSERTATION

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

Computational Mathematics, Science and Engineering—Doctor of Philosophy

2024

ABSTRACT

Generalization is a central research topic in deep learning. To enhance the test performance of well-trained models on unseen data, it is essential to apply regularization techniques that refine the model's expressive capabilities and the training process. This thesis categorizes regularization into theory-driven and data-driven approaches.

Theory-driven regularization encompasses methods that are broadly applicable across various contexts, including conventional techniques such as weight decay and dropout. Conversely, data-driven regularization involves techniques specifically designed for particular data sets and applications. For instance, different neural network architectures can be developed to capture various useful patterns in data for specific applications. This dissertation explores both types of regularization, from the development of new training algorithms with theoretical guarantees to the design of deep learning architectures for data-driven approaches.

For theory-driven regularization, this dissertation discusses a training algorithm based on PAC-Bayes bound. PAC-Bayes bound evaluates the upper bound of the test error using only training data. However, minimizing the upper bound of the test error using existing PAC-Bayes bounds, which are theoretically tight and should intuitively benefit generalization, often results in compromised test performance compared to empirical risk minimization (ERM) with commonly used regularization techniques such as weight decay, large learning rates, and small batch sizes. The designed algorithm seeks to bridge the gap between theoretical tightness and practical effectiveness in boosting test performance for classification tasks.

For data-driven regularization, this dissertation discusses graph neural networks specifically designed for directed graphs and spatial-temporal seismic data. It also introduces a physics-informed deep learning framework for full-waveform inversion, which aims to estimate subsurface structures based on seismic data by integrating the governing acoustic wave equation with convolutional neural networks. Additionally, data augmentation is considered a specialized form of regularization. This thesis explores the design of generative neural networks for time-lapse full-waveform inversion to obtain more training samples and achieve lower test errors in the target inversion task.

The material presented in this dissertation incorporates several publications and preprints. For details on PAC-Bayes training, where the model is trained using the PAC-Bayes bound, review Zhang et al. (2023). For discussions on regularization through the design of graph neural networks, refer to Zhang et al. (2021b) and Zhang et al. (2022). For physics-informed regularization, see Jin et al. (2021). For approaches to data augmentation with generative models, check Yang et al. (2022).

To all warriors exploring in the darkness.

"There is only one heroism in the world: to see the world as it is and to love it."
— Romain Rolland

ACKNOWLEDGEMENTS

Pursuing a PhD degree has been a long and memorable journey, filled with many emotions, including happiness, excitement, hopefulness, and, at times, depression. In this dissertation, I sincerely thank all my friends, colleagues, advisors, and family who have supported me throughout this journey.

Over the course of my six-year journey at Michigan State University, while my research may not be the most acclaimed, I am confident that my experiences are among the most diverse of all the graduates. Throughout my PhD, I had the opportunity to work in several different research labs. After transferring from another department, my journey in the Department of Computational Mathematics, Science, and Engineering began under the guidance of Dr. Matthew Hirn and, later, Dr. Rongrong Wang. They introduced me to high-quality research practices, effective project management, and the essentials of good leadership and mentorship. Additionally, I was fortunate to intern with Dr. Youzuo Lin four years ago, who encouraged and guided me in exploring a completely new field in geoscience. Working with them has been a profoundly rewarding experience; words cannot fully express my gratitude. I would also like to extend my thanks to all my other committee members, Dr. Saiprasad Ravishankar and Dr. Jianrong Wang, for their invaluable guidance, advice, and constructive feedback.

I am deeply thankful to Jingwen Shi, who has been a steadfast companion throughout nearly my entire research career. I also appreciate Qi Wang, who provided encouragement and support as I explored new research paths during challenging times. I am grateful to Guangliang Liu, Haitao Mao, and Zhiyu Xue for their support during my job search. Additionally, I extend my gratitude to all my labmates and friends, including, but not limited to, He Lyu, Avrajit Ghosh, Ismail Alkhouri, Peng Jin, Will Reichard-Flynn, Yuxin Yang, Shihang Feng, Michael Perlmutter, Yixuan He, Xiaorui Liu, and Junyuan Hong. It is impossible to name everyone here; please accept my apologies if I have inadvertently omitted anyone. Most of all, I extend my deepest gratitude to my family, my unwavering supporters. Their understanding, encouragement, and love have been the foundation of my success throughout all that has happened these years.

TABLE OF CONTENTS

CHAPTER 1

OVERVIEW

1.1 Background

Before delving into the details of this dissertation, it is crucial to define what is meant by 'regularization,' a term with varied definitions in the field. For instance, a recent taxonomy has categorized regularization techniques into explicit and implicit types. According to Hernández-García and König (2018), explicit regularization involves techniques that reduce the representational capacity of a model class $\mathcal{H}_0$, such as a neural network, resulting in a sub-hypothesis set $\mathcal{H}_1 \subset \mathcal{H}_0$ based on specific assumptions. Examples of explicit regularization include weight decay and dropout. On the other hand, implicit regularization comprises techniques that reduce generalization error indirectly through characteristics of the network architecture, the training data, or the learning algorithm, as discussed in Zhang et al. (2021a).

Due to some ambiguity in these definitions, this dissertation adopts the taxonomy proposed by Kukačka et al. (2017), defining regularization as follows:

Definition 1.1.1. *Regularization is any supplementary technique that aims at enhancing the model's ability to generalize, i.e., to produce better results on the test set.*

Based on this definition, we can categorize different types of regularization with the following objectives:

An arbitrary deep learning model can be described as a function $f_\theta : x \mapsto y$ with trainable weights $\theta \in \Theta$. The objective of training is to find the optimal weights θ^* that minimize the loss function $\mathcal{L} : \Theta \mapsto \mathbb{R}$:

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta). \tag{1.1}$$

The loss function generally takes the form:

$$\mathcal{L} = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f_\theta(x), y) + R(\cdots)], \tag{1.2}$$

where $\mathcal{D}$ represents the data distribution, ℓ is the misfit function that measures the discrepancy between the network output $f_\theta(x)$ and the target label y, and R is the extra penalty term based on specific criteria, such as Occam's razor (e.g., weight decay).

Since the data distribution $\mathcal{D}$ is generally unknown, we evaluate the loss function using a training dataset $\mathcal{S} \sim \mathcal{D}$, defined as $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m$, where $\mathcal{S}$ comprises m pairs of training samples. Training can then be framed as solving the following optimization task:

$$\theta^* = \arg\min_\theta \frac{1}{|\mathcal{S}|} \sum_{(x_i,y_i)\sim\mathcal{S}} [\ell(f_\theta(x_i), y_i) + R(\cdots)]. \tag{1.3}$$

Based on the above equation, we can identify different sources of regularization:

- f: the neural network architecture, such as the use of pooling layers to achieve output invariance to slight spatial distortions in the input.
- R: the extra penalty term, for example, weight decay and sharpness minimization (Foret et al., 2020).
- $\mathcal{S}$: the training dataset, including techniques like data augmentation (Wang et al., 2017).
- ℓ: the misfit function, such as the dice coefficient which is robust to class imbalance (Milletari et al., 2016).
- $\arg\min_\theta$: the optimization procedure, for instance, implicit regularization in (stochastic) gradient descent (Barrett and Dherin, 2020; Ghosh et al., 2022), as well as early-stopping and warm-start methods.

1.2 Dissertation Contributions

This dissertation focuses on exploring all types of regularization—f, R, ℓ, $\mathcal{S}$, and $\arg\min_\theta$. More specifically:

1. In Section 2.2, we discuss a two-stage training algorithm for neural networks that minimizes the PAC-Bayes bound (Zhang et al., 2023), integrating both ℓ and $\arg\min_\theta$. Previous research

in PAC-Bayes learning theory primarily focused on establishing tight upper bounds for test errors, whereas PAC-Bayes training updates network weights to minimize these bounds for better generalization. While theoretically tight, practical implementations of PAC-Bayes bounds often fell short of achieving test errors as low as those obtained by empirical risk minimization (ERM) with optimally tuned hyperparameters of commonly used regularization, such as learning rate, dropout, and weight decay. Moreover, traditional PAC-Bayes training algorithms Pérez-Ortiz et al. (2021) typically require bounded loss functions and extensive searches over priors using additional datasets, limiting their applicability. Our new PAC-Bayes training algorithm, which allows for unbounded loss and involves a two-stage training process, minimizes reliance on prior tuning by training both the prior and posterior using the same dataset. Comprehensive evaluations across various classification tasks and neural network architectures show that our method not only surpasses existing PAC-Bayes algorithms but also achieves test accuracies comparable to ERM with optimal commonly chosen regularization settings.

2. For f, this dissertation presents the design of graph neural networks, MagNet (Zhang et al., 2021b) and STGNN (Zhang et al., 2022).

    - Section 3.1 introduces MagNet, a graph neural network for directed graphs. Unlike typical GNNs that focus on undirected graphs and require symmetrization, MagNet utilizes a complex Hermitian matrix, the magnetic Laplacian, to preserve direction information. This matrix captures undirected geometric structure in the magnitude of its entries and directional information in their phases. A "charge" parameter adjusts spectral information to account for variations among directed cycles. MagNet has been applied to various node classification and link prediction tasks, showing superior performance compared to other methods on most tasks. It is adaptable to other GNN architectures such as GCN (Kipf and Welling, 2016) and ChebNet (Defferrard et al., 2016).
    - Section 3.2 discusses the Spatiotemporal Graph Neural Network (STGNN) designed

for estimating earthquake locations and magnitudes. Traditional machine learning earthquake characterization methods use waveform information from a single station; STGNN, however, utilizes data from multiple stations to construct dynamic graphs via adaptive message passing. Tested on data from the Southern California Seismic Network and Oklahoma, STGNN has demonstrated more accurate earthquake location predictions than baseline models.

3. Combining f and R, Section 3.3 details an architecture for Full-Waveform Inversion (FWI) (Jin et al., 2021). FWI is typically used in geophysics to estimate subsurface velocity maps from seismic data, a challenging task formulated by a second-order partial differential equation (PDE). By using finite difference methods to approximate forward modeling of the PDE and modeling its inversion with a CNN, we transform the supervised inversion task into an unsupervised seismic data reconstruction task. The architecture effectively acts as an auto-encoder, with the decoder designed around governing physics (f). Perceptual loss (Johnson et al., 2016) has also been found to enhance generalization in this setting (R). Our results indicate that the model, utilizing only seismic data, achieves accuracy comparable to supervised methods and outperforms them when more unlabeled seismic data is included.

4. For the regularization of $\mathcal{S}$, Section 3.4 describes a data augmentation approach based on generative neural networks (Yang et al., 2022) for time-lapse full-waveform inversion (FWI). Traditional data augmentation techniques from computer vision often yield physically unacceptable samples that do not benefit FWI. We developed generative models that incorporate physics knowledge, such as governing equations and observable phenomena, to enhance the quality of the synthetic data. We applied these techniques to detect small CO_2 leakages and validated our methods through comprehensive numerical tests. Our analysis shows that data-driven seismic imaging can be significantly improved with our data augmentation techniques.

1.3 Dissertation Structure

The dissertation is organized as follows.

Chapter 2 explores theory-driven regularization, starting with a review of implicit regularization in gradient descent in Section 2.1. While numerous studies have investigated individual regularization techniques and recognized their advantages, the interactions among these techniques remain less understood. Consequently, extensive tuning of the hyperparameters associated with each regularization technique is often necessary to achieve optimal test performance in practical applications. Motivated by this challenge, Section 2.2 introduces a training algorithm based on PAC-Bayes theory. This algorithm proves effective across various in-domain classification tasks and different architectures, aiming to enhance generalization.

Chapter 3 focuses on data-driven regularization by designing deep learning architectures tailored for specific applications. Section 3.1 introduces MagNet, a graph neural network for directed graphs. Section 3.2 presents STGNN, a graph neural network that predicts earthquake locations and magnitudes based on waveforms collected from various seismic stations. Section 3.3 discusses full-waveform inversion conducted in an unsupervised manner, leveraging governing physical principles. Additionally, Section 3.4 describes a scientific data augmentation approach specifically for time-lapse full-waveform inversion.

Finally, Chapter 4 summarizes the discussed regularization techniques and discusses the potential future work.

CHAPTER 2

THEORY-DRIVEN REGULARIZATION

This chapter focuses on theory-driven regularization, which operates independently of data characteristics. The methods discussed here are applicable across a broader spectrum of applications, underpinned by theoretical frameworks aimed at enhancing generalization.

This chapter initially explores implicit regularization in Section 2.1, represented by the $\arg\min_{\theta}$ term in Equation 1.3. Building on the insights gained from studying implicit regularization effects, this chapter will then delve into a training algorithm that leverages the PAC-Bayes bound, detailed in Section 2.2. This approach illustrates how theoretical principles can guide the development of practical regularization techniques that enhance model performance across various domains.

2.1 Introduction of Implicit Regularization: The Gradient Descent Case

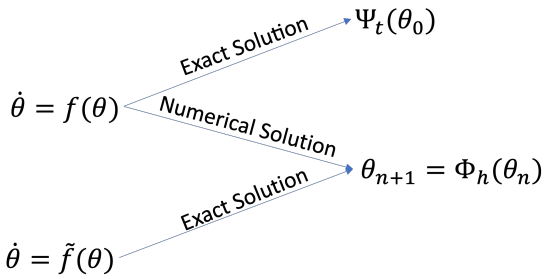


Figure 2.1 Backward error analysis. There is error in the numerical solution of the system $\dot{\theta} = f(\theta)$ because of discretization. Thus, the numerical solution becomes the exact solver of the other system $\dot{\theta} = \tilde{f}(\theta)$.

Barrett and Dherin (2020) pointed out that deep learning models trained with larger learning rates could achieve better generalization performance by analyzing the gradient descent algorithm based on the backward error analysis. The backward error analysis is used to measure the error from the discretization of ODE solvers. The general intention is visualized in Figure 2.1. In order to measure the error of ODE $\theta_{n+1} = \Phi_h(\theta_n)$, which is the numerical solution of the system $\dot{\theta} = f(\theta)$, we can compare $\dot{\theta} = \tilde{f}(\theta)$, the exact solution of $\theta_{n+1} = \Phi_h(\theta_n)$, with $\dot{\theta}$, the discretized solution, by Taylor expansion.

Consider the gradient descent, the iterative weight update process can be represented by the

learning rate h and the gradient of loss function $\nabla_\theta E(\cdot)$:

$$\theta_{n+1} = \theta_n + hf(\theta_n) = \theta_n - h\nabla_\theta E(\theta_n). \tag{2.1}$$

To obtain what $\tilde{f}$ will make the trajectory of $\dot{\theta} = \tilde{f}(\theta)$ coincides with that of (2.1) on the discrete time-grids, we can first write $\tilde{f}(\theta)$ in its Taylor series form

$$\dot{\theta} = \tilde{f}(\theta) = f(\theta) + hf_1(\theta) + h^2 f_2(\theta) + \cdots \tag{2.2}$$

Then for any fixed $t > 0$, by using numerical integration, we have

$$\begin{aligned}
\theta_{t+1} &= \theta_t + h(f(\theta_t) + hf_1(\theta_t) + h^2 f_2(\theta_t) + \cdots) \\
&\quad + \frac{h^2}{2}(f'(\theta_t) + hf_1'(\theta_t) + h^2 f_2'(\theta_t) + \cdots)(f(\theta_t) + hf_1(\theta_t) + h^2 f_2(\theta_t) + \cdots) \\
&\quad + \cdots \\
&\quad \approx \theta_t + hf(\theta_t) + \frac{h^2}{2}(f'(\theta_t)f(\theta_t) + 2f_1(\theta_t)).
\end{aligned} \tag{2.3}$$

To match the two trajectories, $\dot{\theta} = \tilde{f}(\theta)$ should match $\theta_{n+1} = \theta_n + hf(\theta_n)$ whenever $t = n$, $\frac{h^2}{2}f'(\theta_t)f(\theta_t) + h^2 f_1(\theta_t)$ should be 0. Thus, we have $f_1(\theta) = -\frac{1}{2}f'(\theta)f(\theta)$, which turns Equation 2.2 into:

$$\dot{\theta} \approx f(\theta) + hf_1(\theta) = f(\theta) - \frac{h}{2}f'(\theta)f(\theta) = -\nabla(E_\theta(\theta) + \frac{h}{4}||\nabla E_\theta(\theta)||^2). \tag{2.4}$$

Equation 2.4 indicates that the numerical solution of the gradient descent has the implicit regularization term $||\nabla E_\theta(\theta)||^2$. The exact loss that the gradient descent solves becomes:

$$\tilde{E}(\theta) = E(\theta) + \frac{h}{4}||\nabla E(\theta)||^2. \tag{2.5}$$

Consequently, it explains why the testing performance is better with a larger learning rate for the gradient descent method. Following the same analysis process, Smith et al. (2021) has the same conclusion for the stochastic gradient descent method.

There are several other implicit regularization techniques and scenarios. Keskar et al. (2016) notes that training with large batches often leads to convergence at sharp minima, which experimentally results in poorer generalization performance compared to flatter minima. Neyshabur et al. (2014) and

Nakkiran et al. (2021) empirically observe that test performance improves with the addition of more trainable weights, even after achieving 100% training accuracy. Kobak et al. (2020) discusses how features with independent components can act as implicit regularization in ridge regression, where the optimal regularization weight can sometimes be negative. Bishop (1995) shows that adding small Gaussian noise to features leads to a loss expectation equivalent to adding an implicit term of $||\frac{\partial \mathcal{L}}{\partial x}||^2$ to the original loss function. Additionally, Santurkar et al. (2018) empirically examines the effect of batch normalization, concluding that it smooths training without reducing internal covariate shift as initially proposed by Ioffe and Szegedy (2015).

Implicit regularization generally arises from mechanisms where the strength is difficult to tune, unlike traditional regularization techniques such as weight decay. For example, learning rate and momentum (Ghosh et al., 2022) are related to convergence: overly large values can hinder convergence. To enhance the effect of implicit regularization without impacting convergence, one can convert implicit terms to explicit ones. A notable method in this context is Sharpness-Aware Minimization (SAM) (Foret et al., 2020). However, despite SAM's effectiveness in achieving good test performance, Andriushchenko et al. (2023) argue that sharpness does not necessarily correlate well with generalization. Instead, it correlates more with training parameters like the learning rate, which may arbitrarily relate to generalization depending on the setup. This raises a pertinent question: Is there a more reliable metric strongly correlated with generalization? The answer lies in the PAC-Bayes bound, which directly measures the upper bound of the test error. If the PAC-Bayes bound is sufficiently tight, it should correlate perfectly with generalization. This concept will be elaborated in the next section.

2.2 Unlocking Tuning-free Generalization: Minimizing the PAC-Bayes Bound with Trainable Priors

The PAC-Bayes bound is instrumental in assessing the generalization capabilities of machine learning models by estimating the upper limits of test errors. This theoretical framework offers crucial insights into a model's generalization ability and provides a solid foundation for developing practical training algorithms (Shawe-Taylor and Williamson, 1997). PAC-Bayes bounds are particularly valuable as they elucidate the discrepancy between training and generalization errors, underline the importance of incorporating regularizers in empirical risk minimization, and demonstrate how larger datasets can enhance generalization. The efficacy of PAC-Bayes bounds in determining the generalization capabilities of machine learning models has been validated by extensive empirical evidence across various generalization metrics (Jiang et al., 2019).

Minimizing the upper bound of generalization error is inherently advantageous for generalization. This section introduces a training algorithm that leverages the PAC-Bayes bounds, aiming to optimize these theoretical limits through an approach involving trainable priors. The objective function, designed based on the proposed PAC-Bayes bound, corresponds to the ℓ term in Equation 1.3, while the complete practical training algorithm is represented by the $\arg\min_{\theta}$ term.

2.2.1 Introduction

Traditionally, PAC-Bayes bounds have been primarily used for quality assurance or model selection (McAllester, 1998, 1999; Herbrich and Graepel, 2000), particularly with smaller machine learning models. Recent work has introduced a framework that minimizes a PAC-Bayes bound during training large neural networks (Dziugaite and Roy, 2017). Ideally, the generalization performance of deep neural networks could be enhanced by directly minimizing its quantitative upper bounds, specifically the PAC-Bayes bounds, without incorporating any other regularization tricks. However, the effectiveness of applying PAC-Bayes training to deep neural networks is challenged by the well-known issue that PAC-Bayes bounds can become vacuous in highly over-parameterized settings (Livni and Moran, 2020). Additionally, selecting a suitable prior, which should be independent of training samples, is critical yet challenging. This often leads to conducting a parameter search for

the prior using separate datasets (Dziugaite et al., 2021). Furthermore, existing PAC-Bayes training methods are typically tailored for bounded loss (Dziugaite and Roy, 2017, 2018; Pérez-Ortiz et al., 2021), limiting their straightforward application to popular losses like Cross-Entropy.

On the other hand, the prevalent training methods for neural networks, which involve minimizing empirical risk with SGD/Adam, achieve satisfactory test performance. However, they often require integration with various regularization techniques to optimize generalization performance. For instance, research has shown that factors such as larger learning rates (Cohen et al., 2021; Barrett and Dherin, 2020), momentum (Ghosh et al., 2022; Cattaneo et al., 2023), smaller batch sizes (Lee and Jang, 2022), parameter noise injection (Neelakantan et al., 2015; Orvieto et al., 2022), and batch normalization (Luo et al., 2018) all induce higher degrees of *implicit regularization*, yielding better generalization. Besides, various *explicit regularization* techniques, such as weight decay (Loshchilov and Hutter, 2017), dropout (Wei et al., 2020), label noise (Damian et al., 2021) can also significantly affect generalization. While many studies have explored individual regularization techniques to identify their unique benefits, the interaction among these regularizations remains less understood. As a result, in practical scenarios, one has to extensively tune the hyperparameters corresponding to each regularization technique to obtain the optimal test performance.

Although further investigation is needed to fully understand the underlying mechanisms, training models using ERM with various regularization methods remains the prevalent choice and typically delivers state-of-the-art test performance. While PAC-Bayes training is built upon a solid theoretical basis for analyzing generalization, its wider adoption is limited by existing assumptions about loss and challenges in prior selection. Moreover, it is still an open question regarding how to enhance PAC-Bayes training to match the performance of ERM methods with well-tuned regularizations. This section introduces a training algorithm using a new PAC-Bayes bound for unbounded loss. The contribution is summarized as follows:

1. We introduce a new PAC-Bayes bound for unbounded loss complemented by a training algorithm. This algorithm simultaneously optimizes the prior and the posterior using the same dataset.
2. The test performance of the proposed algorithm is theoretically justified.

3. The proposed PAC-Bayes training algorithm outperforms existing methods that minimize other PAC-Bayes bounds in terms of test performance.
4. Our training algorithm approaches the best test performance of the widely-used ERM using SGD/Adam, enhanced by standard regularizations like noise injection and weight decay.
5. Our training algorithm exhibits robustness to variation in hyperparameters such as learning rate and batch size. Besides, the same hyperparameter configuration is effective across various neural network architectures.

2.2.2 Preliminaries

This section outlines the PAC-Bayes framework. For any supervised learning problem, the goal is to find a proper model $\mathbf{h}$ from some hypothesis space $\mathcal{H}$, with the help of the training data $\mathcal{S} \equiv \{z_i\}_{i=1}^m$, where z_i is the training pair with sample $\mathbf{x}_i$ and its label y_i. Given the loss function $\ell(\mathbf{h}; z_i) : \mathbf{h} \mapsto \mathbb{R}^+$, which measures the misfit between the true label y_i and the predicted label by $\mathbf{h}$, the empirical and population/generalization errors are defined as:

$$\ell(\mathbf{h}; \mathcal{S}) = \frac{1}{m} \sum_{i=1}^{m} \ell(\mathbf{h}; z_i), \quad \ell(\mathbf{h}; \mathcal{D}) = \mathbb{E}_{\mathcal{S} \sim \mathcal{D}}(\ell(\mathbf{h}; \mathcal{S})),$$

by assuming that the training and testing data are i.i.d. sampled from the unknown distribution $\mathcal{D}$. PAC-Bayes bounds include a family of upper bounds on the generalization error of the following type.

Theorem 2.2.1. *(Maurer, 2004) Assume the loss function ℓ is **bounded** within the interval $[0, 1]$. Given a **preset** prior distribution $\mathcal{P}$ over the model space $\mathcal{H}$, and given a scalar $\delta \in (0, 1)$, for any choice of i.i.d m-sized training dataset $\mathcal{S}$ according to $\mathcal{D}$, and all posterior distributions $\mathcal{Q}$ over $\mathcal{H}$,*

$$\mathbb{E}_{\mathbf{h} \sim \mathcal{Q}} \ell(\mathbf{h}; \mathcal{D}) \leq \mathbb{E}_{\mathbf{h} \sim \mathcal{Q}} \ell(\mathbf{h}; \mathcal{S}) + \sqrt{\frac{\log(\frac{2\sqrt{m}}{\delta}) + \mathrm{KL}(\mathcal{Q}||\mathcal{P})}{2m}},$$

holds with probability at least $1 - \delta$. Here, KL stands for the Kullback-Leibler divergence.

A PAC-Bayes bound measures the gap between the expected empirical and generalization errors. It's worth noting that this bound holds for all posterior $\mathcal{Q}$ for any given data-independent prior $\mathcal{P}$

and, which enables optimization of the bound by searching for the best posterior. In practice, the posterior expectation corresponds to the trained model, and the prior expectation can be set to the initial model. In this section, we will use $||\cdot||$ to denote a generic norm, and $||\cdot||_2$ to denote L_2 norm.

2.2.3 Related Work

PAC-Bayes bounds were first used to train neural networks in Dziugaite and Roy (2017). Specifically, the bound McAllester (1999) has been employed for training shallow stochastic neural networks on binary MNIST classification with bounded 0-1 loss and has proven to be non-vacuous. Following this work, many recent studies (Letarte et al., 2019; Rivasplata et al., 2019; Pérez-Ortiz et al., 2021; Biggs and Guedj, 2021; Perez-Ortiz et al., 2021; Zhou et al., 2018b) expanded the applicability of PAC-Bayes bounds to a wider range of neural network architectures and datasets. However, most studies are limited to training shallow networks with binary labels using bounded loss, which restricts their broader application to deep network training. Although PAC-Bayes bounds for unbounded loss have been established (Audibert and Catoni, 2011; Alquier and Guedj, 2018; Holland, 2019; Kuzborskij and Szepesvári, 2019; Haddouche et al., 2021; Rivasplata et al., 2020; Rodríguez-Gálvez et al., 2023; Casado et al., 2024), it remains unclear whether these bounds can lead to enhanced test performance in training neural networks. This uncertainty arises partly because they usually include assumptions that are difficult to validate or terms that are hard to compute in real applications. For example, Kuzborskij and Szepesvári (2019) derived a PAC-Bayes bound under the second-order moment condition of the unbounded loss. However, as mentioned in the paper, that bound is semi-empirical, in the sense that it contains the population second order moment of the loss, in contrast to usual PAC-Bayes bounds that only contain empirical quantities that can be computed from the data. To the best of our knowledge, existing PAC-Bayes bounds built under the second-order moment condition all suffer from this issue.

Recently, Dziugaite et al. (2021) suggested that a tighter PAC-Bayes bound could be achieved with a data-dependent prior. They divide the data into two sets, using one to train the prior and the other to train the posterior with the optimized prior, thus making the prior independent from the

training dataset for the posterior. This, however, reduces the training data available for the posterior. Dziugaite and Roy (2018) and Rivasplata et al. (2020) justified the approach of learning the prior and posterior with the same set of data by utilizing differential privacy. However, the argument only holds for priors provably satisfying the so-called $DP(\epsilon)$-condition in differential privacy, which limits their practical application. Pérez-Ortiz et al. (2021) also empirically shows training with Dziugaite and Roy (2018) could sacrifice test accuracy if the bound is not tight enough. In this work, we advance the PAC-Bayes training approach, enhancing its practicality and showcasing its potential in realistic settings.

2.2.4 New PAC-Bayes Bound for Unbounded loss

Popular PAC-Bayes training algorithms (Dziugaite and Roy, 2017, 2018; Pérez-Ortiz et al., 2021) are limited to bounded loss. When dealing with unbounded Cross-Entropy loss[1], they require a clipping of the loss to small bounded regions before applying the training, leading to suboptimal performance. On the other hand, PAC-Bayes bounds for unbounded loss were also established in the literature (Germain et al., 2016; Rodríguez-Gálvez et al., 2023) where the requirement of bounded loss is replaced by the weaker requirement of the finite second-order moment of the loss or finite CGF (cumulant generating function). However, these bounds are often not non-vacuous when applied to deep neural networks (as shown in Figure 2.2 of Section 2.2.8), meaning that the numerical value of the bound is too large for the training to progress.

We propose a modified PAC-Bayes bound that imposes milder conditions, making it effective for training deep networks. The new bound is based on a modification of the existing assumption of the loss function, detailed as follows.

Definition 2.2.2 (Exponential moment on finite intervals)**.** Let X be a random variable defined on the probability space $(\Omega, \mathcal{F}, \mathrm{P})$ and $0 \leq \gamma_1 \leq \gamma_2$ be two numbers. We call any $K > 0$ an exponential moment bound of X over the interval $[\gamma_1, \gamma_2]$, when

$$\mathbb{E}[\exp(\gamma(\mathbb{E}[X] - X))] \leq \exp(\gamma^2 K) \tag{2.6}$$

[1]The MSE loss could also be unbounded when used in regression tasks

holds for all $\gamma \in [\gamma_1, \gamma_2]$.

By restricting the range of γ to a finite interval $[\gamma_1, \gamma_2]$, (2.6) is weaker than the usual exponential moment condition for sub-Gaussian distributions. Later, when we apply this condition to the PAC-Bayes analysis, the random variable X in Def. 2.2.2 will represent the loss function. **Since most loss functions in machine learning (e.g., Cross-Entropy, L_1, MSE, Huber loss, hinge loss, Log-cosh loss, quantile loss) are non-negative, it is of great interest to analyze the strength of Definition 2.2.2 under $X \geq 0$. In this case, we can show that our condition is weaker than the second-order moment condition, which is currently the weakest condition allowing the establishment of PAC-Bayes bounds**.

Lemma 2.2.3. *For non-negative random variable $X \geq 0$, the existence of K on the interval $\gamma \in [0, \infty)$ in Definition 2.2.2 can be implied by the existence of the second-order moment $\mathbb{E}X^2 < \infty$.*

This lemma suggests that for non-negative loss functions, our Definition 2.2.2 is weaker than the second-order moment condition. In addition, the assumption $X \geq 0$ can be further relaxed to $X \geq -M$ with $M > 0$, as in this case the random variable $X + M$ is non-negative to which we can apply Lemma 2.2.3.

Proof of Lemma 2.2.3. We show that $\mathbb{E}X^2 < \infty$ implies Definition 2.2.2 holding for any $\gamma \in [0, \infty)$ with some finite K. Since $\mathbb{E}X^2 < \infty$, we have $(\mathbb{E}X)^2 \leq \mathbb{E}X^2 < \infty$. If $\gamma \geq \frac{1}{\mathbb{E}X}$, then it suffices to take the K in

$$\mathbb{E}e^{\gamma(\mathbb{E}X - X)} \leq e^{\gamma^2 K}$$

to be $K = \frac{\mathbb{E}X}{\gamma} \leq (\mathbb{E}X)^2 \equiv K_1$. If $\gamma < \frac{1}{\mathbb{E}X}$, then using the inequality

$$e^x \leq 1 + x + x^2, \quad \forall x < 1$$

with $x := \gamma(\mathbb{E}X - X) \leq \gamma \mathbb{E}X < 1$, we have

$$\mathbb{E}e^{\gamma(\mathbb{E}X - X)} \leq \mathbb{E}(1 + \gamma(\mathbb{E}X - X) + \gamma^2(\mathbb{E}X - X)^2) = 1 + \gamma^2 \mathrm{Var}(X) \leq e^{\gamma^2 \mathrm{Var}(X)}$$

Therefore, it suffices to take $K = \mathrm{Var}(X) \equiv K_2$. Collecting the two cases, we see taking $K = \max\{K_1, K_2\}$ would be enough for Definition 4.1 to hold with $\gamma_1 = 0, \gamma_2 = \infty$. □

Remark 2.2.4 (Comparison with the first-order-moment condition). Still under the assumption $X \geq 0$, when the γ_1 in Definition 2.2.2 is finite (bounded away from 0), the existence of K can be implied by the existence of first-order moment. Indeed, by taking $K = \frac{\mathbb{E}[X]}{\gamma_1}$, the inequality $\mathbb{E}[X] - X \leq \mathbb{E}[X]$ (assumed $X \geq 0$) immediately implies (2.6). However, this argument does not hold when $\gamma_1 \to 0$. Hence we cannot say our condition is as weak as the first-order moment condition.

We want to emphasize that the main motivation for proposing Definition 4.1 is from an empirical perspective, where we want to have a bound with a smaller numerical value. Therefore, in practice, we always take γ_1, γ_2 to be positive scalars.

In addition, we propose to make the exponential moment bound depend on the prior distribution, which leads to a further reduction of the bound. For this purpose, we first extend Definition 2.2.2 from a single random variable to a family of random variables parameterized by models in a hypothesis space.

Let us first explain what we mean by random variables parameterized by models in a hypothesis space. In the network setting, let us define $X(\mathbf{h})$ as $X(\mathbf{h}) \equiv \ell(f_\theta(x), y)$, where ℓ is the loss and $\mathbf{h} = f_\theta$ is the model/network parametrized by weight θ. For a fixed model $\mathbf{h}$ (i.e. f_θ), we see $X(\mathbf{h})$ is a random variable whose randomness comes from the input pairs $(x, y) \sim \mathcal{D}$ ($\mathcal{D}$ is the data distribution). Since this random variable $X(\mathbf{h})$ varies with $\mathbf{h}$, we call it a random variable parameterized by models $\mathbf{h}$.

Definition 2.2.5 (Exponential moment over hypotheses)**.** Let $X(\mathbf{h})$ be a random variable parameterized by the hypothesis $\mathbf{h}$ in some space $\mathcal{H}$ (i.e., $\mathbf{h} \in \mathcal{H}$), and fix an interval $[\gamma_1, \gamma_2]$ with $0 < \gamma_1 < \gamma_2 < \infty$. Let $\{\mathcal{P}_{\boldsymbol{\lambda}}, \boldsymbol{\lambda} \in \Lambda\}$ be a family of distribution over $\mathcal{H}$ parameterized by $\boldsymbol{\lambda} \in \Lambda \subseteq \mathbb{R}^k$. Then, we call any non-negative function $K(\boldsymbol{\lambda})$ a uniform exponential moment bound for $X(\mathbf{h})$ over the priors $\{\mathcal{P}_{\boldsymbol{\lambda}}, \boldsymbol{\lambda} \in \Lambda\}$ and the interval $[\gamma_1, \gamma_2]$, if the following holds

$$\mathbb{E}_{\mathbf{h} \sim \mathcal{P}_{\boldsymbol{\lambda}}} \mathbb{E}[\exp(\gamma(\mathbb{E}[X(\mathbf{h})] - X(\mathbf{h})))] \leq \exp(\gamma^2 K(\boldsymbol{\lambda})),$$

for all $\gamma \in [\gamma_1, \gamma_2]$, and any $\boldsymbol{\lambda} \in \Lambda \subseteq \mathbb{R}^k$. The minimal such $K(\boldsymbol{\lambda})$ is

$$K_{\min}(\boldsymbol{\lambda}) = \sup_{\gamma \in [\gamma_1, \gamma_2]} \frac{1}{\gamma^2} \log(\mathbb{E}_{\mathbf{h} \sim \mathcal{P}_{\boldsymbol{\lambda}}} \mathbb{E}[\exp\left(\gamma(\mathbb{E}[X(\mathbf{h})] - X(\mathbf{h}))\right)]). \tag{2.7}$$

Similar to Definition 2.2.2, when dealing with non-negative loss, the existence of the exponential moment bound $K_{\min}$ is guaranteed, provided that the second-order moment of the loss is bounded, or provided that the first-order moment of the loss is bounded and γ_1 is bounded away from 0.

Now, we can establish the PAC-Bayes bound for losses that satisfy Definition 2.2.5.

Theorem 2.2.6 (PAC-Bayes bound for unbounded loss with a **preset** prior distribution)**.** *Given a prior distribution $\mathcal{P}_{\boldsymbol{\lambda}}$ over the hypothesis space $\mathcal{H}$, parametrized by $\boldsymbol{\lambda} \in \Lambda$. Assume the loss $\ell(\mathbf{h}, z_i)$ as a random variable parametrized by $\mathbf{h}$ satisfies Definition 2.2.5. Fix some $\delta \in (0, 1)$. For any $0 < \delta < 1$ and $\gamma \in [\gamma_1, \gamma_2]$, we have*

$$P_{\mathcal{S}}\left(\forall Q \in \mathbf{Q}, \mathbb{E}_{h \sim Q} \ell(\boldsymbol{h}; \mathcal{D}) \leq \mathbb{E}_{h \sim Q} \ell(\boldsymbol{h}; \mathcal{S}) + \frac{1}{\gamma m}(\log \frac{1}{\delta} + \mathrm{KL}(Q||\mathcal{P}_{\boldsymbol{\lambda}})) + \gamma K(\boldsymbol{\lambda})\right) \geq 1 - \delta$$

where $\mathbf{Q}$ is the set of all probability distributions.

Remark 2.2.7. **By setting $\gamma = O(m^{-1/2})$, we observe that the asymptotic behavior of this bound aligns with the $O(m^{-1/2})$ convergence rate of popular PAC-Bayes bounds in the literature. A corollary of this theorem and Lemma 2.2.3 is that this convergence rate can be achieved for CE loss under a bounded second-order moment condition. While bounds under the second-order moment condition were derived in the literature, as discussed in Section 2.2.3, our bound seems to be the first purely empirical bound (i.e., computable from data) that can be easily used for training. Moreover, our use of finite $\gamma_1 > 0$ and $\gamma_2 < \infty$ and the permission of K to depend on the prior parameter λ further reduce the value of the bound.**

The proof is available in Appendix A.1.1.

With the relaxed requirements on the loss function, our bound offers a basis for establishing effective optimization over both the posterior and the prior. We will first outline the training process, which focuses on jointly optimizing the prior and posterior to avoid the complex hyper-parameter search over the prior as Pérez-Ortiz et al. (2021), followed by a discussion of its theoretical guarantees.

The procedure is similar to the one in Dziugaite and Roy (2017), but has been adapted to align with our newly proposed bound.

We begin by parameterizing the posterior distribution as $Q_{\boldsymbol{\sigma}}(\mathbf{h})$, where $\mathbf{h} \in \mathbb{R}^d$ represents the mean of the posterior, and $\boldsymbol{\sigma} \in \mathbb{R}^d$ accounts for the variations in each model parameter from this mean (i.e., variance). Next, we parameterize the prior as $\mathcal{P}_{\boldsymbol{\lambda}}$, where $\boldsymbol{\lambda} \in \mathbb{R}^k$. We operate under the assumption that the prior has significantly fewer parameters than the posterior, that is, $k \ll d$; the relevance of this assumption will become apparent upon examining Theorem 2.2.10. For our PAC-Bayes training, we propose to optimize over all four variables: $\mathbf{h}$, γ, $\boldsymbol{\sigma}$, and $\boldsymbol{\lambda}$:

$$(\hat{\mathbf{h}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}}) = \arg \min_{\mathbf{h}, \boldsymbol{\lambda}, \boldsymbol{\sigma}, \gamma \in [\gamma_1, \gamma_2]} L_{PAC}(\mathbf{h}, \gamma, \boldsymbol{\sigma}, \boldsymbol{\lambda}), \tag{P}$$

where

$$L_{PAC}(\mathbf{h}, \gamma, \boldsymbol{\sigma}, \boldsymbol{\lambda}) = \mathbb{E}_{\tilde{\mathbf{h}} \sim Q_{\boldsymbol{\sigma}}(\mathbf{h})} \ell(\tilde{\mathbf{h}}; \mathcal{S}) + \frac{1}{\gamma m}(\log \frac{1}{\delta} + \mathrm{KL}(Q_{\boldsymbol{\sigma}}(\mathbf{h}) || \mathcal{P}_{\boldsymbol{\lambda}})) + \gamma K(\boldsymbol{\lambda}).$$

Compared to previous PAC-Bayes training, the most notable change in L_{PAC} is that we allow K to depend on the prior parameter $\boldsymbol{\lambda}$, and optimize it along with other terms.

We provide an end-to-end theorem that guarantees the performance of this optimization algorithm.

To derive our theorem, we need the following assumptions:

Assumption 2.2.8 (Continuity of the KL divergence)**.** Let $\mathfrak{Q}$ be a family of posterior distributions, let $\mathfrak{P} = \{P_{\boldsymbol{\lambda}}, \boldsymbol{\lambda} \in \Lambda \subseteq \mathbb{R}^k\}$ be a family of prior distributions parameterized by $\boldsymbol{\lambda}$. We say the KL divergence $\mathrm{KL}(Q||\mathcal{P}_{\boldsymbol{\lambda}})$ is continuous with respect to $\boldsymbol{\lambda}$ over the posterior family, if there exists some non-decreasing function $\eta_1(x) : \mathbb{R}_+ \mapsto \mathbb{R}_+$ with $\eta_1(0) = 0$, such that $|\mathrm{KL}(Q||\mathcal{P}_{\boldsymbol{\lambda}}) - \mathrm{KL}(Q||\mathcal{P}_{\tilde{\boldsymbol{\lambda}}})| \leq \eta_1(\|\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}}\|)$, for all pairs $\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}} \in \Lambda$ and for all $Q \in \mathfrak{Q}$.

Assumption 2.2.9 (Continuity of the exponential moment bound)**.** Let $K_{\min}(\boldsymbol{\lambda})$ be as defined in Definition 2.2.5. Assume it is continuous with respect to the parameter $\boldsymbol{\lambda}$ of the prior in the sense that there exists a non-decreasing function $\eta_2(x) : \mathbb{R}_+ \mapsto \mathbb{R}_+$ with $\eta_2(0) = 0$ such that $|K_{\min}(\boldsymbol{\lambda}) - K_{\min}(\tilde{\boldsymbol{\lambda}})| \leq \eta_2(\|\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}}\|)$, for all $\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}} \in \Lambda$.

These two assumptions are quite weak and can be satisfied by popular continuous distributions, such as the exponential family.

We will first present a general theorem applicable to all distribution families satisfying these assumptions. Then we demonstrate why the Gaussian prior/posterior distribution, commonly used in practice, satisfies these assumptions.

Theorem 2.2.10 (PAC-Bayes bound for unbounded losses and **trainable** priors)**.** *Assume the loss* $\ell(\mathbf{h}, z_i)$ *as a random variable parametrized by* $\mathbf{h}$ *satisfies Definition 2.2.5. Let* $\mathfrak{Q}$ *be a family of posterior distribution, let* $\mathfrak{P} = \{P_{\boldsymbol{\lambda}}, \boldsymbol{\lambda} \in \Lambda \subseteq \mathbb{R}^k\}$ *be a family of prior distributions parameterized by* $\boldsymbol{\lambda}$*. Let* $n(\varepsilon) := \mathcal{N}(\Lambda, \|\cdot\|, \varepsilon)$ *be the covering number of the set of the prior parameters. Under Assumption 2.2.8 and Assumption 2.2.9, the following inequality holds for the minimizer* $(\hat{\mathbf{h}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}})$ *of* (P) *and any* $\epsilon, \varepsilon > 0$ *with probability as least* $1 - \epsilon$*:*

$$\mathbb{E}_{\mathbf{h}\sim Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})} \ell(\mathbf{h}; \mathcal{D}) \leq L_{PAC}(\hat{\mathbf{h}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}}) + \eta, \tag{2.8}$$

where $\eta = B\varepsilon + C(\eta_1(\varepsilon) + \eta_2(\varepsilon)) + \frac{\log(n(\varepsilon)+\frac{\gamma_2-\gamma_1}{2\varepsilon})}{\gamma_1 m}$*, and* C *and* B *are constants depending on* γ_1, γ_2*,* η_2*,* m *and the upper bounds of the parameters in the prior and posterior.*

The proof is available in Appendix A.1.2.

The theorem provides a generalization bound on the model learned as the minimizer of (P) with data-dependent priors. This bound contains the PAC-Bayes loss L_{PAC} along with an additional correction term η, that is notably absent in the traditional PAC-Bayes bound with fixed priors. Given that $(\hat{\mathbf{h}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}})$ minimizes L_{PAC}, evaluating L_{PAC} at its own minimizer ensures that the first term is small. If the correction term is also small, then the test error remains low. In the next section, we will delve deeper into the condition for this term to be small. Intuitively, selecting a small ε helps to maintain low values for the first three terms in η. Although a smaller ε increases the $n(\varepsilon)$ in the last term, this increase is moderated because it is inside the logarithm and divided by the size of the dataset.

2.2.5 PAC-Bayes Training Algorithm with Gaussian Families

2.2.6 Gaussian prior and posterior

For the L_{PAC} objective to have a closed-form formula, in this section, we employ the Gaussian distribution family. For ease of illustration, we introduce a new notation. Consider a neural network model denoted as $f_{\boldsymbol{\theta}}$, where f represents the network's architecture, and $\boldsymbol{\theta}$ is the weight. In this context, $f_{\boldsymbol{\theta}}$ aligns with the $\mathbf{h}$ discussed in earlier sections. Moving forward, we will use $f_{\boldsymbol{\theta}}$ to refer to the model instead of $\mathbf{h}$.

We define the posterior distribution of the weights as a Gaussian distribution centered around the trainable weight $\boldsymbol{\theta}$, with trainable variance $\boldsymbol{\sigma}$., i.e., the posterior weight distribution is $\mathcal{N}(\boldsymbol{\theta}, \text{diag}(\boldsymbol{\sigma}))$, denoted by $Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})$, where $\boldsymbol{\sigma}$ includes the anisotropic variance of the weights and $\boldsymbol{\theta}$ includes the mean. The assumption of a diagonal covariance matrix implies the independence of the weights. We consider two types of priors, both centered around the initial weight of the neural network $\boldsymbol{\theta}_0$ (as suggested by Dziugaite and Roy (2017)), but with different settings on the variance.

Scalar prior: we use a universal scalar to encode the variance of all the weights in the prior, i.e., the weight distribution of $\mathcal{P}_\lambda$ is $\mathcal{N}(\boldsymbol{\theta}_0, \lambda I_d)$, where λ is a scalar. With this prior, the KL divergence $\text{KL}(Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P}_\lambda(\boldsymbol{\theta}_0))$ in (P) is:

$$\frac{1}{2}\left[-\mathbf{1}_d^\top \log(\boldsymbol{\sigma}) + d(\log(\lambda) - 1) + \frac{(\|\boldsymbol{\sigma}\|_1 + \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2^2)}{\lambda}\right]. \tag{2.9}$$

Layerwise prior: weights in the ith layer share a common variance $\boldsymbol{\lambda}_i$, but different layers could have different variances. By setting $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_k)$ as the vector containing all the layerwise variances of a k-layer neural network, the weight distribution of prior $\mathcal{P}_{\boldsymbol{\lambda}}$ is $\mathcal{N}(\boldsymbol{\theta}_0, \text{BlockDiag}(\boldsymbol{\lambda}))$, where $\text{BlockDiag}(\boldsymbol{\lambda})$ is obtained by diagonally stacking all $\boldsymbol{\lambda}_i I_{d_i}$ into a $d \times d$ matrix, where d_i is the number of weights of the ith layer. The KL divergence for layerwise prior is in Appendix A.1.3. For shallow networks, it is enough to use the scalar prior; for deep neural networks and neural networks constructed from different types of layers, using the layerwise prior is more sensible.

By plugging in the closed-form (2.9) for $\text{KL}(Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P}_{\boldsymbol{\lambda}}(\boldsymbol{\theta}_0))$ into the PAC-Bayes bound in Theorem 2.2.10, we have the following corollary that justifies the usage of PAC-Bayes bound on large neural networks with the trainable prior.

Corollary 2.2.11. *Suppose the posterior and prior are Gaussian distributions as defined above. Assume all parameters for the prior and posterior are bounded, i.e., we restrict the model parameter* $\boldsymbol{\theta}$*, the posterior variance* $\boldsymbol{\sigma}$ *and the prior variance* $\boldsymbol{\lambda}$*, all to be searched over bounded sets,* $\Theta := \{\boldsymbol{\theta} \in \mathbb{R}^d : \|\boldsymbol{\theta}\|_2 \leq \sqrt{d}M\}$, $\Sigma := \{\boldsymbol{\sigma} \in \mathbb{R}^d_+ : \|\boldsymbol{\sigma}\|_1 \leq dT\}$, $\Lambda =: \{\boldsymbol{\lambda} \in [e^{-a}, e^b]^k\}$*, respectively, with fixed* $M, T, a, b > 0$*. Then,*

- *Assumption 2.2.8 holds with* $\eta_1(x) = L_1 x$*, where* $L_1 = \frac{1}{2}\max\{d, e^a(2\sqrt{d}M + dT)\}$
- *Assumption 2.2.9 holds with* $\eta_2(x) = L_2 x$*, where* $L_2 = \frac{1}{\gamma_1^2}\left(2dM^2e^{2a} + \frac{d(a+b)}{2}\right)$
- *With high probability, the PAC-Bayes bound for the minimizer of* (P) *has the form*

$$\mathbb{E}_{\boldsymbol{\theta}\sim Q_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})}\ell(f_{\boldsymbol{\theta}}; \mathcal{D}) \leq L_{PAC}(\hat{\boldsymbol{\theta}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}}) + \eta,$$

where $\eta = \frac{k}{\gamma_1 m}\left(1 + \log\frac{2(CL+B)\Delta\gamma_1 m}{k}\right)$, $L = L_1 + L_2$, $\Delta := \max\{b + a, 2(\gamma_2 - \gamma_1)\}$, $C = \frac{1}{\gamma_1 m} + \gamma_2$ *B is a constant depending on* $\gamma_1, \delta, M,\ d,\ T,\ a,\ b,\ m$[2]*.*

In the bound, the term $L_{PAC}(\hat{\boldsymbol{\theta}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}})$ is inherently minimized as it evaluates the function L_{PAC} at its own minimizer. The overall bound remains low if the correction term η can be deemed insignificant. The logarithm term in the definition of η grows very mildly with the dimension in general, so we can treat it (almost) as a constant. Thus, $\eta \sim \frac{k}{\gamma_1 m}$, from which we see that 1). η (and therefore the bound) would be small if prior's degree of freedom k is substantially less than the dataset size m 2). This bound still achieves the asymptotic rate of $O(m^{-1/2})$ after optimizing over γ_1. We note that even if the corollary assumes that the parameters (i.e., mean and variance) of the Gaussian distribution are bounded, the random variable itself is still unbounded, so the loss is still unbounded. The proof and more discussions can be found in Appendix A.1.4.

2.2.7 Training algorithm

Estimating $K_{\min}(\boldsymbol{\lambda})$**:** In practice, the function $K_{\min}(\boldsymbol{\lambda})$ must be estimated first. Since we showed in Corollary 2.2.11 and Remark 2.2.4 that $K_{\min}(\boldsymbol{\lambda})$ is Lipschtiz continuous and bounded, we can approximate it using piecewise-linear functions. Notably, since for each fixed $\boldsymbol{\lambda} \in \Lambda$, the prior is

[2]See Appendix A.1.4 for the explicit form of B.

Algorithm 2.1 PAC-Bayes training (scalar prior)

Input: initial weight $\boldsymbol{\theta}_0 \in \mathbb{R}^d, T_1 = 500, \lambda_1 = e^{-12}, \lambda_2 = e^2, \gamma_1 = 0.5, \gamma_2 = 10$. *//$T_1, \lambda_1, \lambda_2, \gamma_1, \gamma_2$ can be fixed in all experiments of Sec.2.2.8.*
Output: trained weight $\hat{\boldsymbol{\theta}}$, posterior noise level $\hat{\boldsymbol{\sigma}}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$, $\mathbf{v} \leftarrow \mathbf{1_d} \cdot \log(\frac{1}{d}\|\boldsymbol{\theta}_0\|_1)$, $b \leftarrow \log(\frac{1}{d}\|\boldsymbol{\theta}_0\|_1)$
Obtain $\hat{K}(\lambda)$ with $\Lambda = [\lambda_1, \lambda_2]$ using (A.20) (Appendix Algorithm A.1)
*/*Stage 1*/*
for epoch = $1 : T_1$ **do**
  for sampling one batch s from $\mathcal{S}$ **do**
    //Ensure non-negative variances
    $\lambda \leftarrow \exp(b)$, $\boldsymbol{\sigma} \leftarrow \exp(\mathbf{v})$
    $\mathcal{P}_\lambda \leftarrow \mathcal{N}(\boldsymbol{\theta}_0; \lambda I_d)$, $Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta}) \leftarrow \boldsymbol{\theta} + \mathcal{N}(\mathbf{0}; \mathrm{diag}(\boldsymbol{\sigma}))$
    //Get the stochastic version of $\mathbb{E}_{\tilde{\boldsymbol{\theta}} \sim Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})} \ell(f_{\tilde{\boldsymbol{\theta}}}; \mathcal{S})$
    Draw one $\tilde{\boldsymbol{\theta}} \sim Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})$ and evaluate $\ell(f_{\tilde{\boldsymbol{\theta}}}; \mathcal{S})$
    Compute the KL divergence as (2.9)
    Compute γ as (2.10)
    Compute the loss function $\mathcal{L}$ as L_{PAC} in (P)
    //Update all parameters
    $b \leftarrow b + \eta \frac{\partial \mathcal{L}}{\partial b}$, $\mathbf{v} \leftarrow \mathbf{v} + \eta \frac{\partial \mathcal{L}}{\partial \mathbf{v}}$, $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$
  end for
end for
//Fix the noise level from now on
$\hat{\boldsymbol{\sigma}} \leftarrow \exp(\mathbf{v})$
*/*Stage 2*/*
while not converge **do**
  for sampling one batch s from $\mathcal{S}$ **do**
    //Noise injection
    Draw one $\tilde{\boldsymbol{\theta}} \sim Q_{\hat{\boldsymbol{\sigma}}}(\boldsymbol{\theta})$ and evaluate $\ell(f_{\tilde{\boldsymbol{\theta}}}; \mathcal{S})$ as $\tilde{\mathcal{L}}$,
    //Update model parameters
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \frac{\partial \tilde{\mathcal{L}}}{\partial \boldsymbol{\theta}}$
  end for
end while
$\hat{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$

independent of the data, this procedure of estimating $K_{\min}(\boldsymbol{\lambda})$ can be carried out before training. More details are in Appendix A.2.1.

Two-stage PAC-Bayes training: Algorithm 2.1 outlines the proposed PAC-Bayes training algorithm that contains two stages. Stage 1 performs pure PAC-Bayes bound minimization, and Stage 2 is a refinement stage. The version of Algorithm 1 that uses a layerwise prior is detailed in Appendix A.2.2. For Stage 1, although there are several input parameters to be specified, one

can use the same choice of values across very different network architectures and datasets with minor modifications. Please see Appendix A.3.1 for more discussions. When everything else in the PAC-Bayes loss is fixed, $\gamma \in [\gamma_1, \gamma_2]$ has a closed-form solution,

$$\gamma^* = \min\left\{\max\left\{\gamma_1, \frac{1}{K_{\min}}\sqrt{\frac{\log\frac{1}{\delta} + \mathrm{KL}(Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||P_{\boldsymbol{\lambda}}(\boldsymbol{\theta}_0))}{m}}\right\}, \gamma_2\right\} \tag{2.10}$$

Therefore, we only need to perform gradient updates on the other three variables, $\boldsymbol{\theta}, \boldsymbol{\sigma}, \boldsymbol{\lambda}$.

The second stage of training: Gastpar et al. (2023); Nagarajan and Kolter (2019) showed that achieving high accuracy on certain distributions precludes the possibility of getting a tight generalization bound in overparameterized settings. This implies that it is less possible to use reasonable generalization bound to fully train one overparameterized model on a particular dataset. By minimizing the PAC-Bayes bound only, it is also observed in our PAC-Bayes training (Stage 1) that the training accuracy is hard to reach 100%. Therefore, we add a second stage to ensure convergence of the training loss. Specifically, in Stage 2, we continue to update the model by minimizing only $\mathbb{E}_{\boldsymbol{\theta}\sim Q_{\tilde{\boldsymbol{\sigma}}}}\ell(f_{\boldsymbol{\theta}}; \mathcal{S})$ over $\boldsymbol{\theta}$, and keep all other variables (i.e., $\boldsymbol{\lambda}$, $\boldsymbol{\sigma}$) fixed to the solution found by Stage 1. This is essentially a stochastic gradient descent with noise injection, the level of which has been learned from Stage 1. The two-stage training is similar to the idea of the learning-rate scheduler (LRS). In LRS, the initial large learning rate introduces an implicit bias that guides the solution path towards a flat region (Cohen et al., 2021; Barrett and Dherin, 2020), and the later lower learning rate ensures the convergence to a local minimizer in this region. Without the large learning rate stage, it cannot reach the flat region; without the small learning rate stage, it cannot converge to a local minimizer. For the two-stage PAC-Bayes training, Stage 1 (PAC-Bayes stage) guides the solution to flat regions by minimizing the generalization bound, and Stage 2 is necessary for an actual convergence to a local minimizer.

Regularizations in the PAC-Bayes training: By plugging the KL divergence (2.9) into P, we can see that in the case of Gaussian priors and posteriors, the PAC-Bayes loss is nothing but the original training loss augmented by a noise injection and a weight decay, except that strength of both of them are automatically learned. More discussions are available in Appendix A.2.3.

Prediction: After training, we use the mean of the posterior as the trained model and perform deterministic prediction on the test dataset. In Appendix A.2.4, we provide some mathematical intuition of why the deterministic predictor is expected to perform even better than the Bayesian predictor.

2.2.8 Experiments

In this section, we demonstrate the efficacy of the proposed PAC-Bays training algorithm through extensive numerical experiments. Specifically, we conduct comparisons between our algorithm and existing PAC-Bayes training algorithms, as well as conventional training algorithms based on Empirical Risk Minimization (ERM). Our approach yields competitive test accuracy in all settings and exhibits a high degree of robustness w.r.t. the choice of hyperparameters.

Comparison with different PAC-Bayes bounds and existing PAC-Bayes training algorithms: We compared our PAC-Bayes training algorithm using the layerwise prior with baselines in Pérez-Ortiz et al. (2021): *quad* (Rivasplata et al., 2019), *lambda* (Thiemann et al., 2017), *classic* (McAllester, 1999), and *bbb* (Blundell et al., 2015) in the context of deep convolutional neural networks. The baseline PAC-Bayes algorithms contain a variety of crucial hyperparameters, including variance of the prior (1e-2 to 5e-6), learning rate (1e-3 to 1e-2), momentum (0.95, 0.99), dropout rate (0 to 0.3) in the training of the prior, and the KL trade-off coefficient (1e-5 to 0.1) for *bbb*. These hyperparameters were chosen by grid search. The batch size is 250 for all methods. Our findings, as detailed in Table 2.1, show that our algorithm outperforms the other PAC-Bayes methods regarding test accuracy. It is important to note that all four baselines employed the PAC-Bayes bound for bounded loss. Therefore, they need to convert unbounded loss into bounded loss for training purposes. Various conversion methods were evaluated by Pérez-Ortiz et al. (2021), and the most effective one was selected for producing the results presented.

To demonstrate the necessity of our newly proposed PAC-Bayes bound for unbounded loss, we compared this new bound with two existing PAC-Bayes bounds for unbounded loss. One is based on the *subGaussian* assumption (Corollary 4 of Germain et al. (2016)), while the other (Theorem 9 of Rodríguez-Gálvez et al. (2023)) assumes the loss function is a bounded cumulant generating

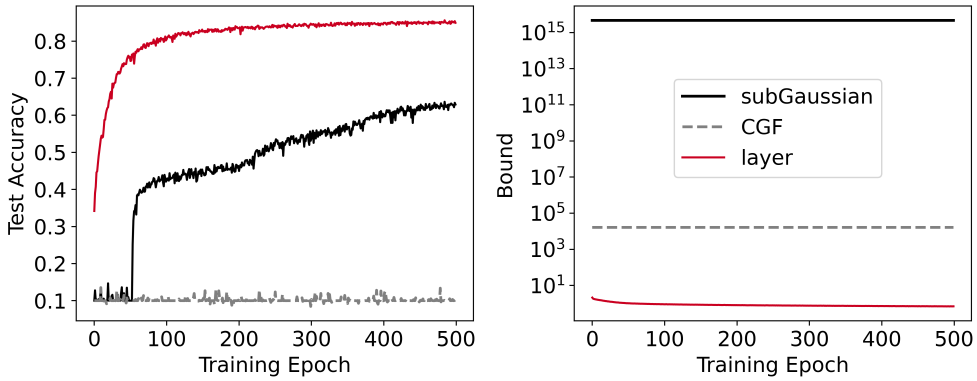


Figure 2.2 Training process when minimizing different PAC-Bayes bounds on CNN9 using CIFAR10 (Stage 1). Minimizing our bound (*layer*) achieves a tighter bound and better test accuracy compared with optimizing the other two (*subGaussian* and *CGF*).

Table 2.1 Test accuracy of convolution neural networks on CIFAR10. The test accuracy of baselines for bounded loss is from Table 5 of Pérez-Ortiz et al. (2021), calculated as 1-the zero-one error of the deterministic predictor. *subG* represents the subGaussian bound. Our proposed PAC-Bayes training with a layerwise prior (*layer*) achieves the best test accuracy across all models.

	bounded				unbounded		
	quad	lambda	classic	bbb	subG	CGF	layer
CNN9	78.63	79.39	78.33	83.49	81.49	80.02	**85.46**
CNN13	84.47	84.48	84.22	85.41	85.84	84.21	**88.31**
CNN15	85.31	85.51	85.20	85.95	85.63	84.36	**87.55**

function (*CGF*). It is important to note that, as of now, no training algorithms specifically leverage these PAC-Bayes bounds for unbounded loss. Therefore, for a fair comparison, we conducted an experiment by replacing our PAC-Bayes bound with the other two bounds and using the same two-stage training algorithm with the trainable layerwise prior.

We found that the two baseline bounds are not non-vacuous on CNN9/13/15; both are larger than 1e5. The subGaussian bound even explodes on CNN13 and CNN15. When using these bounds for training a model, it is expected that they deliver worse [3] performance than the proposed one as shown in Table 2.1. We also visualized the test accuracy when minimizing different PAC-Bayes bounds for unbounded loss in Stage 1. As shown in Figure 2.2, minimizing our PAC-Bayes bound can achieve better generalization performance. The details of the two baseline bounds are in Appendix A.3.2.

[3] Despite the vacuousness of the bound, the final results are still meaningful due to the use of Stage 2.

Table 2.2 Test accuracy of CNNs on C10 (CIFAR10) and C100 (CIFAR100) with batch size 128. Our PAC-Bayes training with scalar and layerwise prior are labeled *scalar* and *layer*. The best and second-best test accuracies are **highlighted** and underlined. Our PAC-Bayes training can approximately match the best performance of the baseline.

	VGG13		VGG19		ResNet18		ResNet34		Dense121	
	C10	C100	C10	C100	C10	C100	C10	C100	C10	C100
SGD	**90.2**	66.9	90.2	**64.5**	**89.9**	64.0	90.0	**70.3**	**91.8**	**74.0**
Adam	88.5	63.7	89.0	58.8	87.5	61.6	87.9	59.5	91.2	70.0
AdamW	88.4	61.8	89.0	62.3	87.9	61.4	88.3	59.9	91.5	70.1
scalar	88.7	**67.2**	89.2	61.3	88.0	68.8	89.6	69.5	91.2	71.4
layer	89.7	67.1	**90.5**	62.3	89.3	**68.9**	**90.9**	69.9	91.5	72.2

Comparison with ERM optimized by SGD/Adam with various regularizations: We tested our PAC-Bayes training on CIFAR10 and CIFAR100 datasets with no data augmentation[4] on various popular deep neural networks, VGG13, VGG19 (Simonyan and Zisserman, 2014), ResNet18, ResNet34 (He et al., 2016), and Dense121 (Huang et al., 2017) by comparing its performance with conventional empirical risk minimization by SGD/Adam enhanced by various regularizations (which we call baselines). The training of baselines involves a grid search for the best hyperparameters, including momentum for SGD (0.3 to 0.9), learning rate (1e-3 to 0.2), weight decay (1e-4 to 1e-2), and noise injection (5e-4 to 1e-2). The batch size was set to be 128. We reported the highest test accuracy obtained from this search as the baseline results. For all convolutional neural networks, our method employed Adam with a fixed learning rate of 1e-4.

Since the CIFAR10 and CIFAR100 datasets do not have a published validation dataset, **we used the test dataset to find the best hyperparameters of baselines during the grid search, which might lead to a slightly inflated performance for baselines.** Nevertheless, as presented in Table 2.2, the test accuracy of our method is still competitive. Please refer to Appendix A.3.5 for more details.

Evaluation on graph neural networks: To demonstrate the broad applicability of the proposed PAC-Bayes training algorithm to different network architectures, we evaluated it on graph neural networks (GNNs). Unlike CNNs, optimal GNN performance has been reported using the AdamW

[4]Result with data augmentation can be found in Appendix A.3.4

Table 2.3 Test accuracy of GNNs trained with AdamW versus our proposed method with scalar prior *scalar*. The best test accuracies are **highlighted**. The performance of our training can almost match the best results of the baseline obtained after carefully tuning hyperparameters.

		CoraML	Citeseer	PubMed	Cora	DBLP
GCN	AdamW	85.7±0.7	**90.3**±0.4	**85.0**±0.6	60.7±0.7	**80.6**±1.4
	scalar	**86.1**±0.7	90.0±0.4	84.9±0.8	**62.0**±0.4	80.5±0.6
GAT	AdamW	85.7±1.0	**90.8**±0.3	84.0±0.4	**63.5**±0.4	**81.8**±0.6
	scalar	**85.9**±0.8	90.6±0.5	**84.4**±0.5	60.9±0.6	81.0±0.5
SAGE	AdamW	85.7±0.5	**90.5**±0.5	83.5±0.4	60.6±0.5	**80.7**±0.6
	scalar	**86.5**±0.5	90.0±0.5	**84.4**±0.6	**61.2**±0.2	79.9±0.5
APPNP	AdamW	86.6±0.7	**91.0**±0.4	85.1±0.5	62.5±0.4	80.6±2.8
	scalar	**87.1**±0.6	90.4±0.5	**85.7**±0.4	**63.5**±0.4	**81.8**±0.5

optimizer for ERM and enabling dropout. To ensure the best baseline results, we conducted a hyperparameter search over learning rate (1e-3 to 1e-2), weight decay (0 to 1e-2), noise injection (0 to 1e-2), and dropout (0 to 0.8) and reported the highest test accuracy as the baseline result. For our method, we used Adam and fixed the learning rate to be 1e-2 for all graph neural networks. We follow the convention for graph datasets by randomly assigning 20 nodes per class for training, 500 for validation, and the remaining for testing.

We tested four architectures GCN (Kipf and Welling, 2016), GAT (Veličković et al., 2018), SAGE (Hamilton et al., 2017), and APPNP (Gasteiger et al., 2018) on 5 benchmark datasets CoraML, Citeseer, PubMed, Cora and DBLP (Bojchevski and Günnemann, 2017). Since there are only two convolution layers for GNNs, applying our algorithm with the scalar prior is sensible. For our PAC-Bayes training, we retained the dropout layer in the GAT as is, since it differs from the conventional dropout and essentially drops the edges of the input graph. Other architectures do not have this type of dropout; hence, our PAC-Bayes training for these architectures does not include dropout.

Table 2.3 demonstrates that the performance of our algorithm closely approximates the best outcome of the baseline. Appendix A.3.6 provides additional details and more results. Extra analysis on few-shot text classification with transformers is in Appendix A.3.7.

Evaluation on the sensitivity of hyperparameters: In previous experiments, we selected

Table 2.4 The test accuracy for CNNs on CIFAR10 (C10) and CIFAR100 (C100) using a batch size of 2048. Values in $(\cdot)$ indicate how much the results differ from using a batch size (128). Our PAC-Bayes training with scalar and layerwise prior are labeled as *scalar* and *layer*. The most robust results w.r.t. the increase of batch size are **highlighted**, indicating the elevated robustness of our method compared to the baseline regarding batch sizes.

	VGG13		ResNet18	
	C10	C100	C10	C100
SGD	87.7 (-2.5)	60.1 (-6.8)	85.4 (-4.5)	61.5 (-2.6)
Adam	90.7 (+2.2)	66.2 (+2.5)	87.7 (+0.2)	65.4 (+3.8)
AdamW	87.2 (-1.1)	61.0 (-0.8)	84.9 (-2.9)	58.9 (-2.5)
scalar	88.9 (**+0.2**)	66.0 (-1.2)	88.9 (+0.9)	68.7 (**-0.1**)
layer	89.4 (-0.3)	67.1 (**0.0**)	89.2 (**-0.1**)	69.3 (+0.3)

Table 2.5 Test accuracy of ResNet18 and VGG13 trained with different learning rates on CIFAR10. The best test accuracies are **highlighted**. Our method is more robust to learning rate variations.

Model	Method	3e-5	5e-5	1e-4	2e-4	3e-4	5e-4	1e-3
ResNet18	layer	**88.4**	**88.8**	**89.3**	**88.6**	**88.3**	**89.2**	87.3
	Adam	66.6	73.9	81.2	85.3	86.4	87.0	**87.5**
VGG13	layer	**88.6**	**88.9**	**89.7**	**89.6**	**89.6**	**89.5**	**88.7**
	Adam	84.3	84.8	85.8	87.4	87.9	88.3	88.5

specific batch sizes and learning rates as the only two tunable hyperparameters of our algorithm, with all other parameters remaining constant across all experiments. We further demonstrate that batch size and learning rate variations do not significantly impact our final performance. This suggests a general robustness of our method to hyperparameters, reducing the necessity for extensive tuning. More specifically, with a fixed learning rate 5e-4 in our method, Table 2.4 shows that changing the batch size from 128 to a very large one, 2048, for VGG13 and ResNet18 does not significantly affect the performance of the PAC-Bayes training compared to ERM with extensive tuning as before. Also, as shown by Table 2.5, our algorithm is more robust to learning rate changes than ERM, which utilizes the optimal weight decay and noise injection settings from Table 2.2. Please refer to Appendix A.3.8 for more results.

2.2.9 Summary

In this section, the objective function is designed by the proposed PAC-Bayes bound for the ℓ term in Equation 1.3, and the comprehensive implementation of the training process is encapsulated by the $\arg\min_{\theta}$ term. This integration of theoretical principles and practical application forms a robust algorithm for reducing generalization error of machine learning models. Specifically, we presented the practical deployment of the PAC-Bayes bound, expanding its use for effectively training neural networks with satisfactory test performance. To realize this, we proposed a new PAC-Bayes bound for unbounded loss with a trainable prior. This new bound overcomes the limitations inherent in the assumptions of bounded loss and extensive prior selection.

CHAPTER 3

DATA-DRIVEN REGULARIZATION

This chapter presents data-dependent regularization approaches. According to the regularization taxonomy in Equation (1.3), the terms f, R, ℓ, and $\mathcal{S}$ can all be data-dependent. To regulate learning and directly learn desired patterns, a new architecture, termed f, is required. Depending on the task, a specific R term, such as the total variation (TV) loss—popular in full-waveform inversion (FWI)—can be incorporated into the loss function. Specialized ℓ functions can be designed to encode physical constraints relevant to scientific data. Furthermore, data augmentation techniques, tailored to the unique characteristics of the data, can be implemented as part of $\mathcal{S}$.

3.1 MagNet: A Neural Network for Directed Graphs

Introducing graph structures to a collection of objects allows the encoding of pairwise relationships, which often possess inherent directional properties. For instance, the WebKB dataset Pei et al. (2020) comprises a list of university websites interconnected by hyperlinks, where one website might link to another without reciprocal linking, typifying directed graphs. In this context, the section introduces *MagNet*, a graph convolutional neural network designed for directed graphs. This network, represented by the f term in Equation (1.3), leverages the magnetic Laplacian to effectively model and learn from the directional relationships present in such datasets.

3.1.1 Introduction

Most graph neural networks fall into one of two families, *spectral networks* or *spatial networks*. Spatial methods define graph convolution as a localized averaging operation with iteratively learned weights. Spectral networks, on the other hand, define convolution on graphs via the eigendecompositon of the (normalized) graph Laplacian. The eigenvectors of the graph Laplacian assume the role of Fourier modes, and convolution is defined as entrywise multiplication in the Fourier basis. For a comprehensive review of both spatial and spectral networks, we refer the reader to Zhou et al. (2018a) and Wu et al. (2020b).

Many spatial graph CNNs have natural extensions to directed graphs. However, these extensions typically only consider the outgoing neighbors of each vertex and neglect the incoming neighbors.

Therefore, they run the risk of discarding potentially important information. Consider, for example, a directed social network such as Twitter, where the nodes are Twitter accounts and a directed edge $(u, v) \in E$ means that account u mentions account v (using the @ functionality). To infer something about account v, there is important information to be gathered both from other accounts that v mentions, and accounts that mention v. Therefore, it is common for spatial methods to preprocess the data by symmetrizing the adjacency matrix, effectively creating an undirected graph. For example, while Veličković et al. (2018) explicitly notes that their network is well-defined on directed graphs, their experiments treat all citation networks as undirected for improved performance.

Extending spectral methods to directed graphs is not straightforward since the adjacency matrix is asymmetric and, thus, there is no obvious way to define a symmetric, real-valued Laplacian with a full set of real eigenvalues that uniquely encodes any directed graph. We overcome this challenge by constructing a network based on the magnetic Laplacian $\mathbf{L}^{(q)}$ defined in Section 3.1.2. Unlike the directed graph Laplacians used in works such as Ma et al. (2019); Monti et al. (2018); Tong et al. (2020a,b), the magnetic Laplacian is not a real-valued symmetric matrix. Instead, it is a *complex-valued Hermitian* matrix that encodes the fundamentally asymmetric nature of a directed graph via the complex phase of its entries.

Since $\mathbf{L}^{(q)}$ is Hermitian, the spectral theorem implies it has an orthonormal basis of complex eigenvectors corresponding to real eigenvalues. Moreover, Theorem B.6.1, stated in Section B.6 of the appendix, shows that $\mathbf{L}^{(q)}$ is positive semidefinite, similar to the traditional Laplacian. Setting $q = 0$ is equivalent to symmetrizing the adjacency matrix and no importance is given to directional information. When $q = .25$, on the other hand, we have that $\mathbf{L}^{(.25)}(u, v) = -\mathbf{L}^{(.25)}(v, u)$ whenever there is an edge from u to v but not from v to u. Different values of q highlight different graph motifs Fanuel et al. (2018, 2017); Guo and Mohar (2017); Mohar (2020), and therefore the optimal choice of q varies. Learning the appropriate value of q from data allows MagNet to adaptively incorporate directed information. We also note that $\mathbf{L}^{(q)}$ has been applied to graph signal processing Furutani et al. (2020), community detection Fanuel et al. (2017), and clustering Cloninger (2017); Fanuel et al. (2018); F. de Resende and F. Costa (2020).

In Section 3.1.4, we show how the networks constructed in Bruna et al. (2014); Defferrard et al. (2016); Kipf and Welling (2016) can be adapted to directed graphs by incorporating complex Hermitian matrices, such as the magnetic Laplacian. When $q = 0$, we effectively recover the networks constructed in those previous works. Therefore, our work generalizes these networks in a way that is suitable for directed graphs. Our method is very general and is not tied to any particular choice of network architecture. Indeed, the main ideas of this work could be adapted to nearly any spectral graph neural network, and some spatial ones.

In Section 3.1.3, we summarize related work on directed graph neural networks as well as other papers studying the magnetic Laplacian and its applications in data science. In Section 3.1.5, we apply our network to node classification and link prediction tasks. We compare against several spectral and spatial methods as well as networks designed for directed graphs. We find that MagNet obtains the best or second-best performance on five out of six node-classification tasks and has the best performance on seven out of eight link-prediction tasks tested on real-world data, in addition to providing excellent node-classification performance on difficult synthetic data. The full implementation details, theoretical results concerning the magnetic Laplacian, extended examples, and further numerical details are in the appendix Section B.

3.1.2 The magnetic Laplacian

Spectral graph theory has been remarkably successful in relating geometric characteristics of undirected graphs to properties of eigenvectors and eigenvalues of graph Laplacians and related matrices. For example, the tasks of optimal graph partitioning, sparsification, clustering, and embedding may be approximated by eigenvectors corresponding to small eigenvalues of various Laplacians (see, e.g., Chung and Graham (1997); Shi and Malik (1997); Belkin and Niyogi (2003); Spielman and Teng (2004); Coifman and Lafon (2006)). Similarly, the graph signal processing research community leverages the full set of eigenvectors to extend the Fourier transform to these structures Ortega et al. (2018). Furthermore, numerous papers Bruna et al. (2014); Defferrard et al. (2016); Kipf and Welling (2016) have shown that this eigendecomposition can be used to define neural networks on graphs. In this section, we provide the background needed to extend these

constructions to directed graphs via complex Hermitian matrices such as the magnetic Laplacian.

We let $G = (V, E)$ be a directed graph where V is a set of N vertices and $E \subseteq V \times V$ is a set of directed edges. If $(u, v) \in E$, then we say there is an edge from u to v. For the sake of simplicity, we will focus on the case where the graph is unweighted and has no self-loops, i.e., $(v, v) \notin E$, but our methods have natural extensions to graphs with self-loops and/or weighted edges. If both $(u, v) \in E$ and $(v, u) \in E$, then one may consider this pair of directed edges as a single undirected edge.

A directed graph can be described by an adjacency matrix $(\mathbf{A}(u, v))_{u,v \in V}$ where $\mathbf{A}(u, v) = 1$ if $(u, v) \in E$ and $\mathbf{A}(u, v) = 0$ otherwise. Unless G is undirected, $\mathbf{A}$ is not symmetric, and, indeed, this is the key technical challenge in extending spectral graph neural networks to directed graphs. In the undirected case, where the adjacency matrix $\mathbf{A}$ is symmetric, the (unnormalized) graph Laplacian can be defined by $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is a diagonal degree matrix. It is well-known that $\mathbf{L}$ is a symmetric, positive-semidefinite matrix and therefore has an orthonormal basis of eigenvectors associated with non-negative eigenvalues. However, when $\mathbf{A}$ is asymmetric, direct attempts to define the Laplacian this way typically yield complex eigenvalues. This impedes the straightforward extension of classical methods of spectral graph theory and graph signal processing to directed graphs.

A key point of this project is to represent the directed graph through a complex Hermitian matrix $\boldsymbol{\mathcal{L}}$ such that: (1) the magnitude of $\boldsymbol{\mathcal{L}}(u, v)$ indicates the presence of an edge, but not its direction; and (2) the phase of $\boldsymbol{\mathcal{L}}(u, v)$ indicates the direction of the edge, or if the edge is undirected. Such matrices have been explored in the directed graph literature (see Section 3.1.3), but not in the context of graph neural networks. They have several advantages over their real-valued matrix counterparts. In particular, a single symmetric real-valued matrix will not uniquely represent a directed graph. Instead, one must use several matrices, as in Tong et al. (2020b), but this increases the complexity of the resulting network. Alternatively, one can work with an asymmetric, real-valued matrix, such as the adjacency matrix or the random walk matrix. However, the spatial graph filters that result from such matrices are typically limited by the fact that they can only aggregate information from the vertices that can be reached in one hop from a central vertex, but ignore the equally important subset

of vertices that can reach the central vertex in one hop. Complex Hermitian matrices, however, lead to filters that aggregate information from both sets of vertices. Finally, one could use a real-valued skew-symmetric matrix but such matrices do not generalize well to graphs with both directed and undirected edges.

The optimal choice of complex Hermitian matrix is an open question. Here, we utilize a parameterized family of magnetic Laplacians, which have proven to be useful in other data-driven contexts Fanuel et al. (2017); Cloninger (2017); Fanuel et al. (2018); F. de Resende and F. Costa (2020). We first define the symmetrized adjacency matrix and corresponding degree matrix by,

$$\mathbf{A}_s(u,v) \coloneqq \frac{1}{2}(\mathbf{A}(u,v) + \mathbf{A}(v,u)), \quad 1 \le u, v \le N, \quad \mathbf{D}_s(u,u) \coloneqq \sum_{v \in V} \mathbf{A}_s(u,v), \quad 1 \le u \le N\,,$$

with $\mathbf{D}_s(u,v) = 0$ for $u \neq v$. We capture directional information via a phase matrix,[1] $\mathbf{\Theta}^{(q)}$,

$$\mathbf{\Theta}^{(q)}(u,v) \coloneqq 2\pi q(\mathbf{A}(u,v) - \mathbf{A}(v,u))\,, \quad q \ge 0\,,$$

where $\exp(i\mathbf{\Theta}^{(q)})$ is defined component-wise by $\exp(i\mathbf{\Theta}^{(q)})(u,v) \coloneqq \exp(i\mathbf{\Theta}^{(q)}(u,v))$. Letting $\odot$ denote component-wise multiplication, we define the complex Hermitian adjacency matrix $\mathbf{H}^{(q)}$ by

$$\mathbf{H}^{(q)} \coloneqq \mathbf{A}_s \odot \exp(i\mathbf{\Theta}^{(q)})\,.$$

Since $\mathbf{\Theta}^{(q)}$ is skew-symmetric, $\mathbf{H}^{(q)}$ is Hermitian. When $q = 0$, we have $\mathbf{\Theta}^{(0)} = \mathbf{0}$ and so $\mathbf{H}^{(0)} = \mathbf{A}_s$. This effectively corresponds to treating the graph as undirected. For $q \neq 0$, the phase of $\mathbf{H}^{(q)}(u,v)$ encodes edge direction and the value $\mathbf{H}^{(q)}(u,v)$ separates four possible cases: no edge, edge from u to v, edge from v to u, and undirected edge. If there is no edge, we will have $\mathbf{H}^q(u,v) = 0$. In the case of a directed edge, the Hermitian adjacency will be complex valued, and changing the direction of an edge will correspond to complex conjugation. For example, in the case where $q = .25$, if there is an edge from u to v but not from v to u we have

$$\mathbf{H}^{(.25)}(u,v) = \frac{i}{2} = -\mathbf{H}^{(.25)}(v,u)\,.$$

[1] Our definition of $\mathbf{\Theta}^{(q)}$ coincides with that used in Furutani et al. (2020). However, another definition (differing by a minus sign) also appears in the literature. These resulting magnetic Laplacians have the same eigenvalues and the corresponding eigenvectors are complex conjugates of one another. Therefore, this difference does not affect the performance of our network since our final layer separates the real and imaginary parts before multiplying by a trainable weight matrix (see Section 3.1.4 for details on the network structure).

Thus, in this setting, an edge from u to v is treated as the opposite of an edge from v to u. On the other hand, if $(u, v), (v, u) \in E$ (which can be interpreted as a single undirected edge), then $\mathbf{H}^{(q)}(u, v) = \mathbf{H}^{(q)}(v, u) = 1$, and we see the phase, $\mathbf{\Theta}^{(q)}(u, v) = 0$, encodes the lack of direction in the edge. For the rest of this section, we will assume that q lies in between these two extreme values, i.e., $0 \leq q \leq .25$. We define the normalized and unnormalized magnetic Laplacians by

$$\mathbf{L}_U^{(q)} \coloneqq \mathbf{D}_s - \mathbf{H}^{(q)} = \mathbf{D}_s - \mathbf{A}_s \odot \exp(i\mathbf{\Theta}^{(q)}), \quad \mathbf{L}_N^{(q)} \coloneqq \mathbf{I} - \left(\mathbf{D}_s^{-1/2}\mathbf{A}_s\mathbf{D}_s^{-1/2}\right) \odot \exp(i\mathbf{\Theta}^{(q)}) \,. \quad (3.1)$$

Note that when G is undirected, $\mathbf{L}_U^{(q)}$ and $\mathbf{L}_N^{(q)}$ reduce to the standard undirected Laplacians.

$\mathbf{L}_U^{(q)}$ and $\mathbf{L}_N^{(q)}$ are Hermitian. Theorem 1 (Section B.6 of the appendix) shows they are positive-semidefinite and thus are diagonalized by an orthonormal basis of complex eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_N$ associated to real, nonnegative eigenvalues $\lambda_1, \ldots, \lambda_N$. Similar to the traditional normalized Laplacian, Theorem 2 (Section B.6 of the appendix) shows the eigenvalues of $\mathbf{L}_N^q$ lie in $[0, 2]$, and we may factor $\mathbf{L}_N^{(q)} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\dagger$, where $\mathbf{U}$ is the $N \times N$ matrix whose k-th column is $\mathbf{u}_k$, $\mathbf{\Lambda}$ is the diagonal matrix with $\mathbf{\Lambda}(k, k) = \lambda_k$, and $\mathbf{U}^\dagger$ is the conjugate transpose of $\mathbf{U}$ (a similar formula holds for $\mathbf{L}_U^{(q)}$). Furthermore, recall $\mathbf{L} = \mathbf{B}\mathbf{B}^\top$, where $\mathbf{B}$ is the signed incidence matrix. Similarly, Theorem 3 (Section B.6 of the appendix) shows that $\mathbf{L}_U^{(q)} = \mathbf{B}^{(q)}(\mathbf{B}^{(q)})^\dagger$, where $\mathbf{B}^{(q)}$ is a modified incidence matrix. The magnetic Laplacian encodes geometric information in its eigenvectors and eigenvalues. In the directed star graph (Section B.7 of the appendix), for example, directional information is contained in the eigenvectors only, whereas the eigenvalues are invariant to the direction of the edges. On the other hand, for the directed cycle graph the magnetic Laplacian encodes the directed nature of the graph solely in its spectrum. In general, both the eigenvectors and eigenvalues may contain important information, which we leverage in MagNet.

3.1.3 Related work

In Section 3.1.3.1, we describe other graph neural networks designed specifically for directed graphs. Notably, none of these methods encode directionality with complex numbers, instead opting for real-valued, symmetric matrices. In Section 3.1.3.2, we review other work studying the magnetic Laplacian which has been studied for several decades and lately has garnered interest in the network science and graph signal processing communities. However, to the best of our knowledge, this is the

first work to use it to construct a graph neural network. We also note there are numerous approaches to graph signal processing on directed graphs. Many of these rely on a natural analog of Fourier modes. These Fourier modes are typically defined through either a factorization of a graph shift operator or by solving an optimization problem. For further review, we refer the reader to Marques et al. (2020).

3.1.3.1 Neural networks for directed graphs

In Ma et al. (2019), the authors construct a directed Laplacian, via identities involving the random walk matrix and its stationary distribution $\mathbf{\Pi}$. When G is undirected, one can use the fact that $\mathbf{\Pi}$ is proportional to the degree vector to verify this directed Laplacian reduces to the standard normalized graph Laplacian. However, this method requires G to be strongly connected, unlike MagNet. The authors of Tong et al. (2020b) use a first-order proximity matrix $\mathbf{A}_F$ (equivalent to $\mathbf{A}_s$ here), as well as two second-order proximity matrices $\mathbf{A}_{S_{\text{in}}}$ and $\mathbf{A}_{S_{\text{out}}}$. $\mathbf{A}_{S_{\text{in}}}$ is defined by $\mathbf{A}_{S_{\text{in}}}(u, v) \neq 0$ if there exists a w such that $(w, u), (w, v) \in E$, and $\mathbf{A}_{S_{\text{out}}}$ is defined analogously. These three matrices collectively describe and distinguish the neighborhood of each vertex and those vertices that can reach a vertex in a single hop. The authors construct three different Laplacians and use a fusion operator to share information across channels. Similarly, inspired by Benson et al. (2016), in Monti et al. (2018), the authors consider several different symmetric Laplacian matrices corresponding to a number of different graph motifs.

The method of Tong et al. (2020a) builds upon the ideas of both Ma et al. (2019) and Tong et al. (2020b) and considers a directed Laplacian similar to the one used in Ma et al. (2019), but with a PageRank matrix in place of the random-walk matrix. This allows for applications to graphs which are not strongly connected. Similar to Tong et al. (2020b), they use higher-order receptive fields (analogous to the second-order adjacency matrices discussed above) and an inception module to share information between receptive fields of different orders. We also note Klicpera et al. (2019a), which uses an approach based on PageRank in the spatial domain. There are also some related methods for directed graphs that are not based on the graph Laplacian, such as the directed graph embedding Sim et al. (2021), and directed message passing for molecular graphs Klicpera et al.

(2019b).

3.1.3.2 Related work on the magnetic Laplacian and Hermitian adjacency matrices

The magnetic Laplacian has been studied since at least Lieb and Loss (1993). The name originates from its interpretation as a quantum mechanical Hamiltonian of a particle under magnetic flux. Early works focused on d-regular graphs, where the eigenvectors of the magnetic Laplacian are equivalent to those of the Hermitian adjacency matrix. The authors of Guo and Mohar (2017), for example, show that using a complex-valued Hermitian adjacency matrix rather than the symmetrized adjacency matrix reduces the number of small, non-isomorphic cospectral graphs. Topics of current research into Hermitian adjacency matrices include clustering tasks Cucuringu et al. (2020) and the role of the parameter q Mohar (2020).

The magnetic Laplacian is also the subject of ongoing research in graph signal processing Furutani et al. (2020), community detection Fanuel et al. (2017), and clustering Cloninger (2017); Fanuel et al. (2018); F. de Resende and F. Costa (2020). For example, Fanuel et al. (2018) uses the phase of the eigenvectors to construct eigenmap embeddings analogous to Belkin and Niyogi (2003). The role of q is highlighted in the works of Fanuel et al. (2018, 2017); Guo and Mohar (2017); Mohar (2020), which show how particular choices of q may highlight various graph motifs. In our context, this indicates that q should be carefully tuned via cross-validation. Lastly, we note that numerous other directed graph Laplacians have been studied and applied to data science Chung (2005); Chung and Kempton (2013); Palmer and Zheng (2021). However, as alluded to in Section 3.1.2, these methods typically do not use complex Hermitian matrices.

3.1.4 MagNet

Most graph neural network architectures can be described as being either *spectral* or *spatial*. Spatial networks such as Veličković et al. (2018); Hamilton et al. (2017); Atwood and Towsley (2016); Duvenaud et al. (2015) typically extend convolution to graphs by performing a weighted average of features over neighborhoods $\mathcal{N}(u) = \{v : (u, v) \in E\}$. These neighborhoods are well-defined even when E is not symmetric, so spatial methods typically have natural extensions to directed graphs. However, such simplistic extensions may miss important information in the

directed graph. For example, filters defined using $\mathcal{N}(u)$ are not capable of assimilating the equally important information contained in $\{v : (v, u) \in E\}$. Alternatively, these methods may also use the symmetrized adjacency matrix, but they cannot learn to balance directed and undirected approaches.

In this section, we show how to extend spectral methods to directed graphs using the magnetic Laplacian introduced in Section 3.1.2. To highlight the flexibility of our approach, we show how three spectral graph neural network architectures can be adapted to incorporate the magnetic Laplacian. Our approach is very general, and so for most of this section, we will perform our analysis for a general complex Hermitian, positive semidefinite matrix. However, we view the magnetic Laplacian as our primary object of interest (and use it in all of our experiments in Section 3.1.5) because of the large body of literature studying its spectral properties and applying it to data science (see Section 3.1.3).

3.1.4.1 Spectral convolution via the magnetic Laplacian

In this section, we let $\boldsymbol{\mathcal{L}}$ denote a Hermitian, positive semidefinite matrix, such as the normalized or unnormalized magnetic Laplacian introduced in Section 3.1.2, on a directed graph $G = (V, E)$, $|V| = N$. We let $\mathbf{u}_1 \ldots, \mathbf{u}_N$ be an orthonormal basis of eigenvectors for $\boldsymbol{\mathcal{L}}$ and let $\mathbf{U}$ be the $N \times N$ matrix whose k-th column is $\mathbf{u}_k$. We define the directed graph Fourier transform for a signal $\mathbf{x} : V \to \mathbb{C}$ by $\widehat{\mathbf{x}} = \mathbf{U}^\dagger \mathbf{x}$, so that $\widehat{\mathbf{x}}(k) = \langle \mathbf{x}, \mathbf{u}_k \rangle$. We regard the eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_N$ as the generalizations of discrete Fourier modes to directed graphs. Since $\mathbf{U}$ is unitary, we have the Fourier inversion formula

$$\mathbf{x} = \mathbf{U}\widehat{\mathbf{x}} = \sum_{k=1}^{N} \widehat{\mathbf{x}}(k)\mathbf{u}_k \,. \tag{3.2}$$

In Euclidean space, convolution corresponds to pointwise multiplication in the Fourier basis. Thus, we define the convolution of $\mathbf{x}$ with a filter $\mathbf{y}$ in the Fourier domain by $\widehat{\mathbf{y} * \mathbf{x}}(k) = \widehat{\mathbf{y}}(k)\widehat{\mathbf{x}}(k)$. By (3.2), this implies $\mathbf{y} * \mathbf{x} = \mathbf{U}\mathrm{Diag}(\widehat{\mathbf{y}})\widehat{\mathbf{x}} = (\mathbf{U}\mathrm{Diag}(\widehat{\mathbf{y}})\mathbf{U}^\dagger)\mathbf{x}$, and so we say $\mathbf{Y}$ is a convolution matrix if

$$\mathbf{Y} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^\dagger \,, \tag{3.3}$$

for a diagonal matrix $\boldsymbol{\Sigma}$. This is the natural generalization of the class of convolutions used in Bruna et al. (2014).

Next, following Defferrard et al. (2016) (see also Hammond et al. (2011)), we show that a spectral network can be implemented in the spatial domain via polynomials of $\boldsymbol{\mathcal{L}}$ by having $\boldsymbol{\Sigma}$ be a polynomial of $\boldsymbol{\Lambda}$ in (3.3). This reduces the number of trainable parameters to prevent overfitting, avoids explicit diagonalization of the matrix $\boldsymbol{\mathcal{L}}$, (which is expensive for large graphs), and improves stability to perturbations Levie et al. (2019). As in Defferrard et al. (2016), we define a normalized eigenvalue matrix, with entries in $[-1, 1]$, by $\widetilde{\boldsymbol{\Lambda}} = \frac{2}{\lambda_{\max}}\boldsymbol{\Lambda} - \mathbf{I}$ and assume

$$\boldsymbol{\Sigma} = \sum_{k=0}^{K} \theta_k T_k(\widetilde{\boldsymbol{\Lambda}}) \,,$$

for some real-valued $\theta_1, \ldots, \theta_k$, where T_k is the Chebyshev polynomial defined by $T_0(x) = 1, T_1(x) = x$, and $T_k(x) = 2xT_{k-1}(x) + T_{k-2}(x)$ for $k \geq 2$. With $(\mathbf{U}\widetilde{\boldsymbol{\Lambda}}\mathbf{U}^\dagger)^k = \mathbf{U}\widetilde{\boldsymbol{\Lambda}}^k\mathbf{U}^\dagger$, one has

$$\mathbf{Y}\mathbf{x} = \mathbf{U}\sum_{k=0}^{K} \theta_k T_k(\widetilde{\boldsymbol{\Lambda}})\mathbf{U}^\dagger\mathbf{x} = \sum_{k=0}^{K} \theta_k T_k(\widetilde{\boldsymbol{\mathcal{L}}})\mathbf{x} \,, \tag{3.4}$$

where, analogous to $\widetilde{\boldsymbol{\Lambda}}$, we define $\widetilde{\boldsymbol{\mathcal{L}}} \coloneqq \frac{2}{\lambda_{\max}}\boldsymbol{\mathcal{L}} - \mathbf{I}$. It is important to note that, due to the complex Hermitian structure of $\widetilde{\boldsymbol{\mathcal{L}}}$, the value $\mathbf{Y}\mathbf{x}(u)$ aggregates information both from the values of $\mathbf{x}$ on $\mathcal{N}_k(u)$, the k-hop neighborhood of u, and the values of $\mathbf{x}$ on $\{v : \mathrm{dist}(v, u) \leq k\}$, which consists of those of vertices that can reach u in k-hops. While in an undirected graph these two sets of vertices are the same, that is not the case for general directed graphs. Furthermore, due to the difference in phase between an edge (u, v) and an edge (v, u), the filter matrix $\mathbf{Y}$ is also capable of aggregating information from these two sets in different ways. This capability is in contrast to any single, symmetric, real-valued matrix, as well as any matrix that encodes just $\mathcal{N}(u)$.

To obtain a network similar to Kipf and Welling (2016), we set $K = 1$, assume that $\boldsymbol{\mathcal{L}} = \mathbf{L}_N^{(q)}$, using $\lambda_{\max} \leq 2$ make the approximation $\lambda_{\max} \approx 2$, and set $\theta_1 = -\theta_0$. With this, we obtain

$$\mathbf{Y}\mathbf{x} = \theta_0(\mathbf{I} + (\mathbf{D}_s^{-1/2}\mathbf{A}_s\mathbf{D}_s^{-1/2}) \odot \exp(i\boldsymbol{\Theta}^{(q)}))\mathbf{x} \,.$$

As in Kipf and Welling (2016), we substitute $\mathbf{I} + (\mathbf{D}_s^{-1/2}\mathbf{A}_s\mathbf{D}_s^{-1/2}) \odot \exp(i\boldsymbol{\Theta}^{(q)}) \to \widetilde{\mathbf{D}}_s^{-1/2}\widetilde{\mathbf{A}}_s\widetilde{\mathbf{D}}_s^{-1/2} \odot \exp(i\boldsymbol{\Theta}^{(q)})$. This renormalization helps avoid instabilities arising from vanishing/exploding gradients and yields

$$\mathbf{Y}\mathbf{x} = \theta_0\widetilde{\mathbf{D}}_s^{-1/2}\widetilde{\mathbf{A}}_s\widetilde{\mathbf{D}}_s^{-1/2} \odot \exp(i\boldsymbol{\Theta}^{(q)}) \,, \tag{3.5}$$

where $\widetilde{\mathbf{A}}_s = \mathbf{A}_s + \mathbf{I}$ and $\widetilde{\mathbf{D}}_s(i,i) = \sum_j \widetilde{\mathbf{A}}_s(i,j)$.

In theory, the matrix $\exp(i\boldsymbol{\Theta}^{(q)})$ is dense. However, in practice, one only needs to compute a small fraction of its entries. In most real-world datasets, the symmetrized adjacency matrix will be sparse. Since the Hermitian adjacency matrix is constructed via pointwise multiplication between the symmetrized adjacency matrix and the phase matrix, it is only necessary to compute the phase matrix for entries (u,v) where $\mathbf{A}_s(u,v) \neq 0$. Thus, the efficiency of the proposed algorithm is comparable to standard GCN algorithms, and can leverage any existing developments such as Fey et al. (2021) that increase efficiency of standard GCNs (although the computational complexity our method does differ by a factor of four because of the computational complexity of complex-valued multiplication).

3.1.4.2 The MagNet architecture

Let L be the number of convolution layers in our network, and let $\mathbf{X}^{(0)}$ be an $N \times F_0$ input feature matrix with columns $\mathbf{x}_1^{(0)}, \ldots \mathbf{x}_{F_0}^{(0)}$. Since our filters are complex, we use a complex version of ReLU defined by $\sigma(z) = z$, if $-\pi/2 \leq \arg(z) < \pi/2$, and $\sigma(z) = 0$ otherwise (where $\arg(z)$ is the complex argument of $z \in \mathbb{C}$). Let F_ℓ be the number of channels in layer ℓ, and for $1 \leq \ell \leq L$, $1 \leq i \leq F_{\ell-1}$, and $1 \leq j \leq F_\ell$, we let $\mathbf{Y}_{ij}^{(\ell)}$ be a convolution matrix defined in the sense of either (3.3), (3.4), or (3.5). Define the ℓ^{th} layer feature matrix $\mathbf{X}^{(\ell)}$ with columns $\mathbf{x}_1^{(\ell)}, \ldots \mathbf{x}_{F_\ell}^{(\ell)}$ as:

$$\mathbf{x}_j^{(\ell)} = \sigma\left(\sum_{i=1}^{F_{\ell-1}} \mathbf{Y}_{ij}^{(\ell)} \mathbf{x}_i^{(\ell-1)} + \mathbf{b}_j^{(\ell)}\right), \tag{3.6}$$

with $\mathbf{b}_j^{(\ell)}(v) = b_j^{(\ell)}$ and $\text{real}(b_j^{(\ell)}) = \text{imag}(b_j^{(\ell)})$. In matrix form we write $\mathbf{X}^{(\ell)} = \mathbf{Z}^{(\ell)}\left(\mathbf{X}^{(\ell-1)}\right)$, where $\mathbf{Z}^{(\ell)}$ is a hidden layer of the form (3.6). In the numerical experiments reported in Section 3.1.5, we utilize formulation (3.4) with $\mathcal{L} = \mathbf{L}_N^{(q)}$. In most cases we set $K = 1$, for which

$$\mathbf{X}^{(\ell)} = \sigma\left(\mathbf{X}^{(\ell-1)}\mathbf{W}_{\text{self}}^{(\ell)} + \widetilde{\mathbf{L}}_N^{(q)}\mathbf{X}^{(\ell-1)}\mathbf{W}_{\text{neigh}}^{(\ell)} + \mathbf{B}^{(\ell)}\right),$$

where $\mathbf{W}_{\text{self}}^{(\ell)}$ and $\mathbf{W}_{\text{neigh}}^{(\ell)}$ are learned weight matrices corresponding to the filter weights in (3.4), and $\mathbf{B}^{(\ell)}(v,\cdot) = (b_1^{(\ell)}, \ldots, b_{F_\ell}^{(\ell)})$ for each $v \in V$.

After the convolutional layers, we unwind the complex $N \times F_L$ matrix $\mathbf{X}^{(L)}$ into a real-valued $N \times 2F_L$ matrix, apply a linear layer, consisting of right-multiplication by a $2F_L \times n_c$ weight matrix

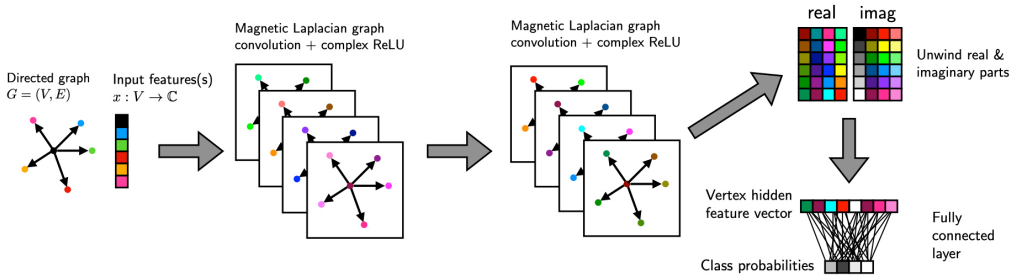
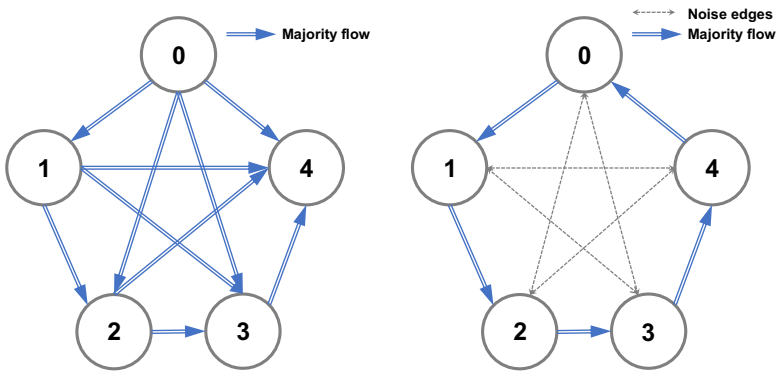


Figure 3.1 MagNet ($L = 2$) applied to node classification.



(a) Ordered meta-graph. (b) Cyclic meta-graph.

Figure 3.2 Meta-graphs for the synthetic data sets.

$\mathbf{W}^{(L+1)}$ (where n_c is the number of classes) and apply softmax. In our experiments, we set $L = 2$ or 3. When $L = 2$, our network applied to node classification, as illustrated in Figure 3.1, is given by

$$\text{softmax}(\text{unwind}(\mathbf{Z}^{(2)}(\mathbf{Z}^{(1)}(\mathbf{X}^{(0)})))\mathbf{W}^{(3)}) .$$

For link-prediction, we apply the same method through the unwind layer, and then concatenate the rows corresponding to pairs of nodes to obtain the edge features.

3.1.5 Experiments

3.1.5.1 Datasets

Directed Stochastic Block Model

We construct a directed stochastic block (DSBM) model as follows. First we divide N vertices into n_c equally-sized clusters $C_1, \ldots, C_{n_c}$. We define $\{\alpha_{i,j}\}_{1 \le i,j \le n_c}$ to be a collection of probabilities, $0 < \alpha_{i,j} \le 1$ with $\alpha_{i,j} = \alpha_{j,i}$, and for an unordered pair $u \ne v$ create an undirected edge between u and v with probability $\alpha_{i,j}$ if $u \in C_i, v \in C_j$. To turn this undirected graph into a directed graph, we define $\{\beta_{i,j}\}_{1 \le i,j \le n_c}$ to be a collection of probabilities such that $0 \le \beta_{i,j} \le 1$ and $\beta_{i,j} + \beta_{j,i} = 1$. For

each undirected edge $\{u, v\}$, we assign that edge a direction by the rule that the edge points from u to v with probability $\beta_{i,j}$ if $u \in C_i$ and $v \in C_j$, and points from v to u otherwise. If $\alpha_{i,j}$ is constant, then the only way to determine the clusters will be from the directional information.

In Figure 3.3, we plot the performance of MagNet and other methods on variations of the DSBM. In each of these, we set $n_c = 5$ and the goal is to classify the vertices by cluster. We set $N = 2500$, except in Figure 3.3d where $N = 500$. In Figure 3.3a, we plot the performance of our model on the DSBM with $\alpha_{i,j} \coloneqq \alpha^* = .1, .08$, and $.05$ for $i \neq j$, which varies the density of inter-cluster edges, and set $\alpha_{i,i} = .1$. Here we set $\beta_{i,i} = .5$ and $\beta_{i,j} = .05$ for $i > j$. This corresponds to the ordered meta-graph in Figure 3.2a. Figure 3.3b also uses the ordered meta-graph, but here we fix $\alpha_{i,j} = .1$ for all i, j, and set $\beta_{i,j} = \beta^*$, for $i > j$, and allow β^* to vary from .05 to .4, which varies the net flow (related to flow imbalance in He et al. (2021)) from one cluster to another. The results in Figure 3.3c utilize a cyclic meta-graph structure as in Figure 3.2b (without the gray noise edges). Specifically, we set $\alpha_{i,j} = .1$ if $i = j$ or $i = j \pm 1 \mod 5$ and $\alpha_{i,j} = 0$ otherwise. We define $\beta_{i,j} = \beta^*$, $\beta_{j,i} = 1 - \beta^*$ when $j = (i - 1) \mod 5$, and $\beta_{i,j} = 0$ otherwise. In Figure 3.3d we add noise to the cyclic structure of our meta-graph by setting $\alpha_{i,j} = .1$ for all i, j and $\beta_{i,j} = .5$ for all (i, j) connected by a gray edge in Figure 3.2b (keeping $\beta_{i,j}$ the same as in Figure 3.3c for the blue edges).

Real datasets

Texas, *Wisconsin*, and *Cornell* are WebKB datasets modeling links between websites at different universities Pei et al. (2020). We use these datasets for both link prediction and node classification with nodes labeled as student, project, course, staff, and faculty in the latter case. *Telegram* Bovet and Grindrod (2020) is a pairwise influence network between 245 Telegram channels with $8,912$ links. To the best of our knowledge, this dataset has not previously been studied in the graph neural network literature. Labels are generated from the method discussed in Bovet and Grindrod (2020), with a total of four classes. The datasets *Chameleon* and *Squirrel* Rozemberczki et al. (2019) represent links between Wikipedia pages related to chameleons and squirrels. We use these datasets for link prediction. Likewise, *WikiCS* Mernyei and Cangea (2020) is a collection of Computer Science articles. *Cora-ML* and *CiteSeer* are popular citation networks with node labels corresponding to

scientific subareas. We use the versions of these datasets provided in Bojchevski and Günnemann (2017). Further details are given in the appendix.

3.1.5.2 Training and implementation details

Node classification is performed in a semi-supervised setting (i.e., access to the test data, but not the test labels, during training). For the datasets *Cornell*, *Texas*, *Wisconsin*, and *Telegram* we use a 60%/20%/20% training/validation/test split, which might be viewed as more akin to supervised learning, because of the small graph size. For *Cora-ML* and *CiteSeer*, we use the same split as Tong et al. (2020a). For all of these datasets we use 10 random data splits. For the DSBM datasets, we generated 5 graphs randomly for each type and for each set of parameters, each with 10 different random node splits. We use 20% of the nodes for validation and we vary the proportion of training samples based on the classification difficulty, using 2%, 10%, and 60% of nodes per class for the ordered, cyclic, and noisy cyclic DSBM graphs, respectively, during training, and the rest for testing. Hyperpameters were selected using one of the five generated graphs, and then applied to the other four generated graphs.

In the main text, there are two types of link prediction tasks conducted for performance evaluation. The first type is to predict the edge direction of pairs of vertices u, v for which either $(u, v) \in E$ or $(v, u) \in E$. The second type is existence prediction. The model is asked to predict if $(u, v) \in E$ by considering ordered pairs of vertices (u, v). For both types of link prediction, we removed 15% of edges for testing, 5% for validation, and use the rest of the edges for training. The connectivity was maintained during splitting. 10 splits were generated randomly for each graph and the input features are in-degree and out-degree of nodes. In the appendix, we report on two additional link prediction tasks based on a three-class classification setup: $(u, v) \in E$, $(v, u) \in E$, or $(u, v), (v, u) \notin E$. Full details are provided in the appendix.

In all experiments, we used the normalized magnetic Laplacian and implement MagNet with convolution defined as in (3.4), meaning that our network may be viewed as the magnetic Laplacian generalization of ChebNet. The setting of the hyperparameter q and other network hyperparameters is obtained by cross-validation. Since currently complex tensors are still in beta in PyTorch, we

did not use them, and instead we stored any complex tensor as two real tensors (one for the real part, one for the imaginary part), and carried out complex multiplication using the standard formula: $(a + ib)(c + id) = (ac - bd) + i(bc + ad)$ (note, a, b, c, d can be real numbers or real matrices). We compare with multiple baselines in three categories: (i) spectral methods: ChebNet Defferrard et al. (2016), GCN Kipf and Welling (2016); (ii) spatial methods: APPNP Klicpera et al. (2019a), SAGE Hamilton et al. (2017), GIN Xu et al. (2018), GAT Veličković et al. (2018); and (iii) methods designed for directed graphs: DGCN Tong et al. (2020b), and two variants of Tong et al. (2020a), a basic version (DiGraph) and a version with higher order inception blocks (DiGraphIB). All baselines in the experiments have two graph convolutional layers, except for the node classification on the DSBM using the cyclic meta-graphs (Figures 3.3c, 3.3d, and 3.2b) for which we also tested three layers during the hyperparameter search. For ChebNet, we use the symmetrized adjacency matrix. For the spatial networks we apply both the symmetrized and asymmmetric adjacency matrix for node classification. The results reported are the better of the two results. The appendix provides full details, as well as results for two other types of baselines: (i) BiGCN, BiSAGE, BiGAT which are obtained by applying GCN, SAGE, GAT on both the original adjacency matrix and the transposed adjacency matrix; and (ii) a k-nearest neighbors classifier based on the eigenvector with the smallest eigenvalue of the magnetic Laplacian Fanuel et al. (2017).

3.1.5.3 Results

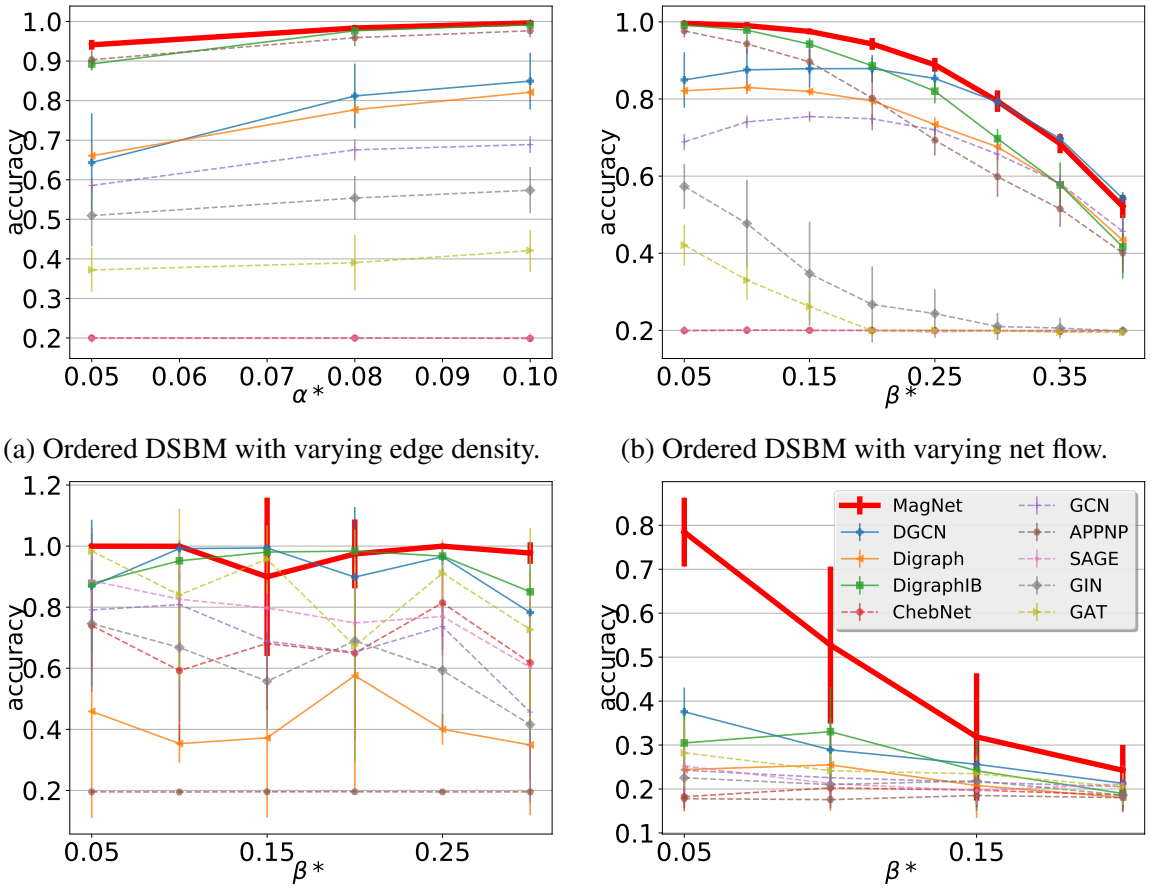
We see that MagNet performs well across all tasks. As indicated in Table 3.1, our cross-validation procedure selects $q = 0$ for node classification on the citation networks *Cora-ML* and *CiteSeer*. This means we achieved the best performance when regarding directional information as noise, suggesting symmetrization-based methods are appropriate in the context of node classification on citation networks. This matches our intuition. For example, in *Cora-ML*, the task is to classify research papers by scientific subarea. If the topic of a given paper is "machine learning," then it is likely to both cite and be cited by other machine learning papers. For all other datasets, we find the optimal value of q is nonzero, indicating that directional information is important. Our network exhibits the best performance on three out of six of these datasets and is a close second on *Texas*

Table 3.1 Node classification accuracy (%). The best results are in **bold** and the second are underlined.

Type	Method	Cornell	Texas	Wisconsin	Cora-ML	CiteSeer	Telegram	Score
Spectral	ChebNet	79.8±5.0	79.2±7.5	81.6±6.3	80.0±1.8	66.7±1.6	70.2 ±6.8	6.94
	GCN	59.0±6.4	58.7±3.8	55.9±5.4	82.0±1.1	66.0±1.5	73.4 ±5.8	19.16
Spatial	APPNP	58.7±4.0	57.0±4.8	51.8±7.4	**82.6±1.4**	66.9±1.8	67.3±3.0	18.75
	SAGE	80.0±6.1	**84.3±5.5**	83.1±4.8	82.3±1.2	66.0±1.5	66.4±6.4	5.76
	GIN	57.9±5.7	65.2±6.5	58.2±5.1	78.1±2.0	63.3±2.5	86.4±4.3	16.53
	GAT	57.6±4.9	61.1±5.0	54.1±4.2	81.9±1.0	67.3±1.3	72.6±7.5	16.39
Directed	DGCN	67.3±4.3	71.7±7.4	65.5±4.7	81.3±1.4	66.3±2.0	**90.4±5.6**	8.55
	Digraph	66.8±6.2	64.9±8.1	59.6±3.8	79.4±1.8	62.6±2.2	82.0±3.1	15.70
	DiGraphIB	64.4±9.0	64.9±13.7	64.1±7.0	79.3±1.2	61.1±1.7	64.1±7.0	16.36
Ours	MagNet	**84.3±7.0**	83.3±6.1	**85.7±3.2**	79.8±2.5	**67.5±1.8**	87.6±2.9	**1.10**
	Best q	0.25	0.15	0.05	0.0	0.0	0.15	-

and *Telegram*. We also achieve an at least four percentage point improvement over both ChebNet and GCN on the four data sets for which $q > 0$. These networks are similar to ours but with the classical graph Laplacian. This isolates the effects of the magnetic Laplacian and shows that it is a valuable tool for encoding directional information. MagNet also compares favorably to non-spectral methods on the WebKB networks (*Cornell*, *Texas*, *Wisconsin*). Indeed, MagNet obtains a $\sim 4\%$ improvement on *Cornell* and a $\sim 2.5\%$ improvement on *Wisconsin*, while on *Texas* it has the second best accuracy, close behind SAGE. We also see the other directed methods have relatively poor performance on the WebKB networks, perhaps since these graphs are fairly small and have very few training samples. To make this analysis more quantitative, we computed the absolute difference of the classification accuracy of each method from the classification accuracy of the top performing method (in percentage points) on each data set, and averaged over the six data sets. In this context, lower scores are better, and a method with a score of zero indicates the method is the top performing method on each data set. As reported in Table 3.1, MagNet achieved a best score of 1.1 percent.

On the DSBM datasets, as illustrated in Figure 3.3, we see that MagNet generally performs quite well and is the best performing network in the vast majority of cases. The networks DGCN and DiGraphIB rely on second order proximity matrices. As demonstrated in Figure 3.3c, these methods



(a) Ordered DSBM with varying edge density.

(b) Ordered DSBM with varying net flow.

(c) Cyclic DSBM with varying net flow.

(d) Noisy Cyclic DSBM with varying net flow.

Figure 3.3 Node classification accuracy. Error bars are one standard error. MagNet is bold red.

are well suited for networks with a cyclic meta-graph structure since nodes in the same cluster are likely to have common neighbors. MagNet, on the other hand, does not use second-order proximity, but can still learn the clusters by stacking multiple layers together. This improves MagNet's ability to adapt to directed graphs with different underlying topologies. This is illustrated in Figure 3.3d where the network has an approximately cyclic meta-graph structure. In this setting, MagNet continues to perform well, but the performance of DGCN and DiGraphIB deteriorate significantly. Interestingly, MagNet performs well on the DSBM cyclic meta-graph (Figure 3.3c) with $q \approx .1$, whereas $q \geq .2$ is preferred for the other three DSBM tests; we leave a more in-depth investigation for future work.

For link prediction, we achieve the best performance on seven out of eight tests as shown in Table 3.2. We also note that Table 3.2 reports optimal non-zero q values for each task. This indicates that incorporating directional information is important for link prediction, even on citation networks

Table 3.2 Link prediction accuracy (%). The best results are in **bold** and the second are underlined.

	Direction prediction				Existence prediction			
	Cornell	Wisconsin	Cora-ML	CiteSeer	Cornell	Wisconsin	Cora-ML	CiteSeer
ChebNet	71.0±5.5	67.5±4.5	72.7±1.5	68.0±1.6	80.1±2.3	82.5±1.9	80.0±0.6	77.4±0.4
GCN	56.2±8.7	71.0±4.0	79.8±1.1	68.9±2.8	75.1±1.4	75.1±1.9	81.6±0.5	76.9±0.5
APPNP	69.5±9.0	75.1±3.5	83.7±0.7	77.9±1.6	74.9±1.5	75.7±2.2	82.5±0.6	78.6±0.7
SAGE	75.2±11.0	72.0±3.5	68.2±0.8	68.7±1.5	79.8±2.4	77.3±2.9	75.0±0.0	74.1±1.0
GIN	69.3±6.0	74.8±3.7	83.2±0.9	76.3±1.4	74.5±2.1	76.2±1.9	82.5±0.7	77.9±0.7
GAT	67.9±11.1	53.2±2.6	50.0±0.1	50.6±0.5	77.9±3.2	74.6±0.0	75.0±0.0	75.0±0.0
DGCN	80.7±6.3	74.5±7.2	79.6±1.5	78.5±2.3	80.0±3.9	**82.8±2.0**	82.1±0.5	81.2±0.4
DiGraph	79.3±1.9	82.3±4.9	80.8±1.1	81.0±1.1	80.6±2.5	**82.8±2.6**	81.8±0.5	**82.2±0.6**
DiGraphIB	79.8±4.8	82.0±4.9	83.4±1.1	82.5±1.3	80.5±3.6	82.4±2.2	82.2±0.5	81.0±0.5
MagNet	**82.9±3.5**	**83.3±3.0**	**86.5±0.7**	**84.8±1.2**	**81.1±3.3**	**82.8±2.2**	**82.7±0.7**	79.9±0.6
Best q	0.20	0.10	0.20	0.15	0.25	0.05	0.05	0.05

such as *Cora* and *CiteSeer*. This matches our intuition, since there is a clear difference between a paper with many citations and one with many references.

3.1.6 Summary

In this section, we introduced MagNet, a neural network specifically designed for directed graphs, utilizing the magnetic Laplacian. This network represents an advancement in spectral graph convolutional networks by extending their application to directed graphs, thereby regularizing learning through the f term in Equation (1.3). We have demonstrated the effectiveness of MagNet, particularly highlighting the crucial role of incorporating directional information via a complex Hermitian matrix. Our results, based on both real and synthetic datasets for tasks such as link prediction and node classification, confirm that MagNet provides substantial improvements in handling directional data compared to traditional methods.

3.2 Spatio-Temporal Graph Convolutional Networks for Earthquake Source Characterization

This section discusses the application of a graph neural network designed specifically for earthquake source characterization. This network exemplifies the regularization of the f term in Equation (1.3), focusing on the unique challenges presented by spatio-temporal seismic data inherent in earthquake events.

3.2.1 Introduction

Earthquake source characterization plays a fundamental role in various seismic studies, including earthquake early-warning, hazard assessment, subsurface energy exploration, etc. Li et al. (2020a). Characterization of an earthquake source can be posed as a classical inverse problem. Its purpose is to infer the source information (location, magnitude, etc.) from seismic recordings. Various approaches have been developed to characterize earthquake sources, the most well-established being traveltime-based inversion Zhang et al. (2017); Li and van der Baan (2016); Lin et al. (2015); Zhang and Thurber (2003) and waveform-based inversion Beskardes et al. (2018); Zhebel and Eisner (2015); Pesicek et al. (2014); Gajewski et al. (2007). Traveltime-based methods implement a multi-step process, in which the arrival times of P and S waves are determined through phase detection and associated to specific earthquakes; earthquake locations are estimated as an inversion process given arrival times, station locations, and a velocity model. Magnitudes are calculated based on waveform amplitudes and source-receiver distances. Though traveltime-based methods are commonly used in seismic applications, they are susceptible to noise-related errors, particularly when estimating low-magnitude events, and fail to utilize abundant phase and amplitude information in the complete waveform. In contrast, waveform-based inversion integrates all phase and amplitude information recorded in seismographs, resulting in high quality source characterization. However, waveform-based inversion is computationally expensive. Both methods require domain expertise to properly tune parameters in the inversion process. Deep learning for source characterization provides a data-driven alternative, where integrated location and magnitude predictions extract full-waveform features with less computational expense than waveform inversion.

Advances in algorithms and computing, and the availability of large, high-quality datasets have

allowed machine learning techniques to attain spectacular success in seismological applications Kong et al. (2019); Bergen et al. (2019) including phase picking Zhu and Beroza (2019), seismic discrimination Li et al. (2018), waveform denoising Zhu et al. (2019a), phase association Ross et al. (2019), earthquake location Perol et al. (2018), as well as magnitude estimation Mousavi and Beroza (2020b). Although machine learning has long been applied to seismic event detection Wang and Teng (1995); Tiira (1999), the first work to leverage recent advances in deep learning was developed by Perol et al. (2018), where convolutional neural networks (CNN's) were trained to detect earthquakes from single station recordings and predict the source locations from among six regions. Though successful in establishing foundational research in machine learning for earthquake location, the CNN model is restricted to waveforms from a single seismic station and can only classify earthquakes into broad geographic groups without providing specific location information. Since then, more advanced single-station approaches have been developed to improve location accuracy. Mousavi and Beroza (2020a) build Bayesian neural networks to learn epicenter distance, P-wave travel time, and associated uncertainty from single-station data.

Recently, multi-station based machine learning methods have shown promising results. For instance, Kriegerowski et al. (2019) develop a CNN structure that combines three-component waveforms from multiple stations to predict hypocenter locations, resulting in more accurate source parameters than single station methods. Zhang et al. (2020) developed an end-to-end fully convolutional network (FCN) to predict the probability distribution of earthquake location directly from input data recorded at multiple stations, which was extended to determine earthquake locations and magnitudes from continuous waveforms for earthquake early warning Zhang et al. (2021c). Shen and Shen (2021) also adopt a CNN framework, extracting the location, magnitude, and origin time from continuous waveforms collected across a seismic network.

Though multiple-station approaches improve upon single-station methods, the use of standard convolutional layers is limited in several ways: (1) CNN's are designed to function on evenly-spaced grids (i.e. photographs) where information is exclusively shared between adjacent cells, and (2) CNN's require the input of station locations to be static (i.e. recordings from station N must always

be found at position N of the input file) in order to learn positional mapping. These assumptions are inappropriate for seismic networks, which are not regularly-spaced and may record information related to non-adjacent stations. Additionally, station outages, the addition/removal of stations to seismic networks, and the ability to select a localized array for the detection of small-magnitude events makes dynamic station input highly desirable for source characterization.

To solve this problem, recently several graph-based machine learning methods have been developed. Münchmeyer et al. (2020) developed an attention-based transformer model for earthquake early warning, which was extended to predict hypocenters and magnitudes of events in Münchmeyer et al. (2021). While this model is successful in implementing a multistation approach that allows for dynamic inputs, high computational complexity restricts inputs to a relatively small number of stations. Another method for implementing flexible, multi-station input that avoids high complexity for large networks is through graph convolution. This method is implemented by van den Ende and Ampuero (2020), who develop a multi-station source characterization model. This model regards features as nodes on an edgeless graph, implementing single-station convolution and global pooling. However, global pooling may not sufficiently extract all useful information from multiple seismic stations, as the pooling layer is ideally applied after global features are obtained by feature fusion along the spatial dimension. Yano et al. (2021) introduce a multi-station technique in which edges are selected and held fixed for all inputs. While this model allows for more meaningful features to be constructed than in global pooling, station inputs are required to be fixed during training and implementation, introducing the same limitation inherent to CNN's. McBrearty and Beroza (2022) propose a GNN framework using multiple pre-defined graphs constructed on both labels and station locations. The model allows for variation in the set of input stations, but the inputs are waveform amplitudes and phase arrival times rather than whole waveforms.

To harness the full functionality of Graph Neural Networks (GNN's) while maintaining flexibility in the location and number of seismic stations, we design a data-driven framework, spatio-temporal graph neural network (STGNN), that creates edges automatically to combine waveform features and spatial information. In order to evaluate the performance of our approach, we compare STGNN

to two baselines: the GNN model designed by van den Ende and Ampuero (2020) and the Fully Convolutional Network (FCN) designed by Zhang et al. (2020). We apply all three models to the two datasets upon which the baselines were originally tested and trained: (1) regional $2.5 < M < 6$ earthquakes recorded by 185 seismic stations in Southern California from 2000 to 2019, and (2) local $0 < M < 4$ earthquakes recorded by 30 seismic stations in Oklahoma from 2014 to 2015.

3.2.2 Methodology

3.2.2.1 Overview

Graph neural networks (GNN's) are designed to handle graphical data, or data that can be represented by vertices connected by edges. In GNN's, convolution and pooling operates along connecting edges. In CNN's, on the other hand, convolution and pooling operates on regions closest together on a Euclidean grid, meaning that input order directly impacts information-sharing and featurization. This is not the case for GNN's, in which edges are not restricted to Euclidean grids but may instead be constructed by any criteria. Two major advantages of GNN architectures are that they do not require a fixed input order, and can handle graphs with different sets of vertices. These properties of GNN's are well-suited for seismic data analysis with inputs from multiple stations. It is common for stations in a seismic network to be added, removed, or repositioned, or for the recording quality of individual stations to fluctuate over time due to operation and/or equipment issues. It is therefore beneficial to dynamically select relevant seismic stations for source characterization. We therefore propose a dynamic GNN framework as the basis for STGNN, in which seismic stations act as nodes, connected by dynamically defined edges.

Inspired by Wang et al. (2019), our graph convolutions follow the design of EdgeConv layers to automatically generate edges between nodes. Instead of manually constructing fixed edges or implementing an edgeless graph, our framework learns to combine useful information from multiple stations implicitly during the training process. Our framework consists of three major components as shown in Figure 3.4:

1. Waveform feature extraction: We first extract temporal features from the waveform recorded at each seismic station using a CNN-based encoder. The three-channel seismic recordings are

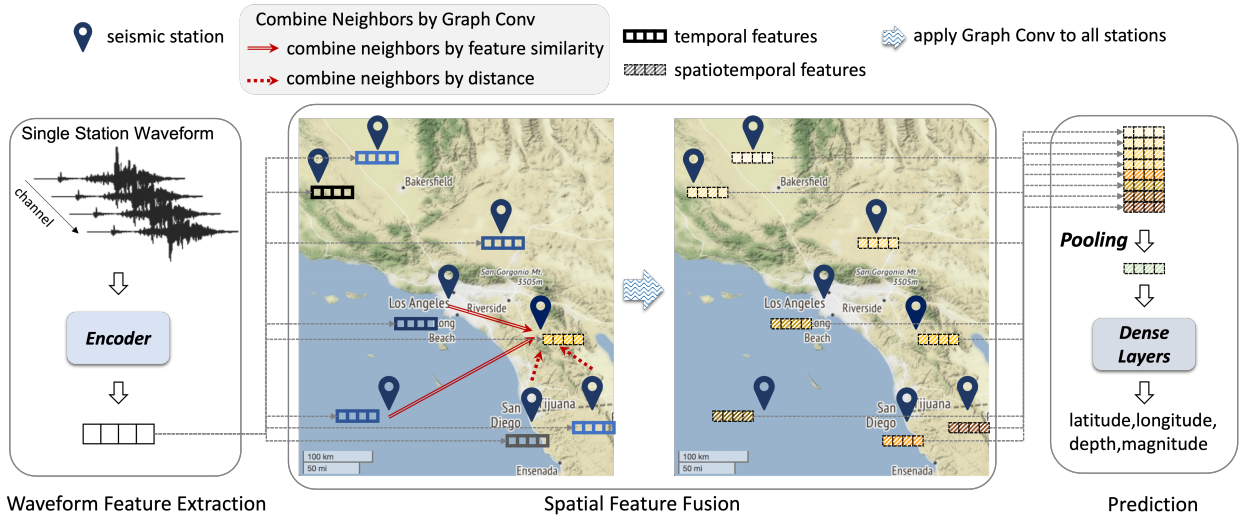


Figure 3.4 The overview of STGNN. There are three major components in STGNN: (1) Waveform feature extraction for obtaining time domain feature from each station independently. (2) Spatial feature fusion for time domain feature integration from different stations based on their geographic locations and extracted feature similarity. (3) Earthquake location and magnitude prediction given spatiotemporal features from the previous step. See Fig 3.5 for a more detailed summary of the network architecture.

reduced to a low dimensional representation.

2. Spatial feature fusion: We then represent the seismic station network as a graph, in which each node (i.e. station) is connected to other nodes by automatically generated edges. Through iterative steps of edge generation and convolution, the perceptive field is gradually enlarged. The model integrates and fuses features from different stations to obtain a high-order spatiotemporal representation of the recorded wavefield over the seismic network. The graph convolutional architecture considers both geographic locations and waveform feature similarity among multiple seismic stations.

3. Prediction: The last component is the prediction module. A fully-connected neural network outputs four normalized scalars corresponding to latitude, longitude, depth and magnitude based on features learned from the previous steps.

3.2.2.2 Graph Convolutional Layers

The spatial feature fusion process consists of multiple graph convolution layers. The goal of each graph convolution layer is to enlarge the perceptive field by combining the extracted feature of each seismic stations and auto-selected neighbor stations. Each graph convolution layer can be broken down into two steps: edge generation and feature update.

Edge generation. Each station node is connected to several other station nodes which show maximum similarity to the node. Similarity measurements are based on two criteria:

1. *Geographic distance:* The geographic distance is the intuitive choice, since adjacent stations tend to record related signals due to similar wave paths. Additionally, events are more likely to be mutually recorded by stations in close proximity, especially in the case of small-magnitude events.

2. *Feature similarity:* As the same earthquake event can be recorded by distant stations in a large area, waveform similarity provides a complimentary perspective to geographic distance. We compare l_2 distance of features from station i and j directly by $||x_i - x_j||^2$, and thus we can combine waveform features from distant stations, where x_i and x_j are the extracted feature vectors.

In edge generation, we link every station with its K-nearest neighbors based on their similarity, where K is a tunable hyperparapeter. In our framework, both geographic proximity and waveform feature similarity are considered.

By ignoring the feature channel and batch dimensions, we assume the feature for neighbors selection is $X \in \mathbb{R}^{N \times d}$, where N is the number of stations and d is the feature dimension. d equals to 2 when the criterion of generating edges is geographic distance (longitude and latitude). Let $x_i = X(i)$ and $x_j = X(j)$, the process of selecting K-nearest neighbors can be explained with the following equations:

1. Compute the pair-wise distance matrix $D \in \mathbb{R}^{N \times N}$:

$$D(i, j) = ||x_i - x_j||^2, 1 \leq i \leq N, 1 \leq j \leq N \tag{3.7}$$

2. Get K-nearest neighbors $\mathcal{N} \in \mathbb{R}^{N \times K}$ by sorting each row of D:

$$\mathcal{N}(i) = \text{smallest-K}(D(i)), 1 \leq i \leq N \tag{3.8}$$

In practice, the similarity between waveforms can also be affected by other factors, such as wave path and signal to noise ratio. By training with a large amount of samples with different sets of seismic stations with distinct spatial distributions, the network will learn to embed these implicit and complex factors to low dimensional features automatically in order to minimize the misfit between labels and predictions.

Feature update. Given the edges, we update the features of each stations by

$$\widetilde{x_i} = \underset{j \in \mathcal{N}_{\text{distance}}(i)}{\text{maxpool}} \left(g(x_i - x_j) + f(x_i)\right) + \underset{j' \in \mathcal{N}_{\text{similarity}}(i)}{\text{maxpool}} \left(g'(x_i - x_{j'}) + f'(x_i)\right), \tag{3.9}$$

where x_i, x_j and x'_j are features of station i, j and j', respectively. j is a neighbor of i based on geographic distance and j' is a neighbor of i by measuring feature similarity from the previous edge generation step. $g(\cdot)$, $f(\cdot)$, $g'(\cdot)$, and $f'(\cdot)$ are trainable fully connected neural networks. $\widetilde{x_i}$ is the updated feature of station i. Max pooling is conducted along the constructed edges to combine information from the K-nearest neighbors of i, so that each station is once more associated with a single feature vector. The update is asymmetric for station i and j to encourage the update processes of i and j to be different, as it is possible that only one of the stations records the event.

3.2.2.3 Architecture

The model takes as input (1) A list of station coordinates, and (2) waveforms recorded by each station. Each input can contain an arbitrary set of stations, limited by a trainable maximum. If the number of functioning stations is less than the maximum number of stations for which the model is trained, the input is padded with zeroed channels and the coordinates of the missing stations are set to $(-1, -1)$.

A graphical illustration of the architecture is presented in Figure 3.5. In Temporal Feature Extraction, time domain waveform features are extracted from each station independently using an encoder with eleven convolutional layers. These features are used to construct the initial inputs for Spatial Feature Fusion.

Two graphs are generated for each layer of graph convolution: one in which edges are generated based on geographic distance, and one in which edges are generated based on waveform feature similarity. For graphs in which geographic distance dictates edges, two scalars containing station coordinates, normalized between -1 and 1, are concatenated to the station's feature vector for neighbor selection and graph convolution. Spatial Feature Fusion uses four layers of graph convolutional layers to obtain spatially hierarchical features.

The features from all four graph convolutions are concatenated together along the feature dimension for final source characterization regression. A fully-connected layer transforms the features in each station, and the features are then compressed with adaptive max pooling along the station dimension. The compressed features are regressed to scalar predictions of latitude, longitude, depth, and magnitude using a fully-connected neural network. The objective function is

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{4}||y_i - \widehat{y_i}||, \tag{3.10}$$

where $\widehat{y_i}$ and y_i are the prediction and ground truth values of ith sample, respectively, represented as vectors of latitude, longitude, depth and magnitude.

The maximum number of stations and number of edges are architectural parameters set during training. The model design in PyTorch allows retraining to new architectural parameters without fundamental alteration of the code.

3.2.3 Experiments

In this section, the data, experiment settings, and results are discussed. We evaluate STGNN in two ways: (1) performance on two datasets compared to GNN and CNN baselines, and (2) stability analysis of STGNN with various settings.

3.2.3.1 Data Description

The Southern California dataset uses waveforms and catalogue information collected by the Southern California Earthquake Data Center (SCEDC) Hutton et al. (2010), and was used for training and testing in the GNN baseline van den Ende and Ampuero (2020). The selected dataset contains events from January 2000 to June 2019 within a geographic subset from 32° to 36° latitude

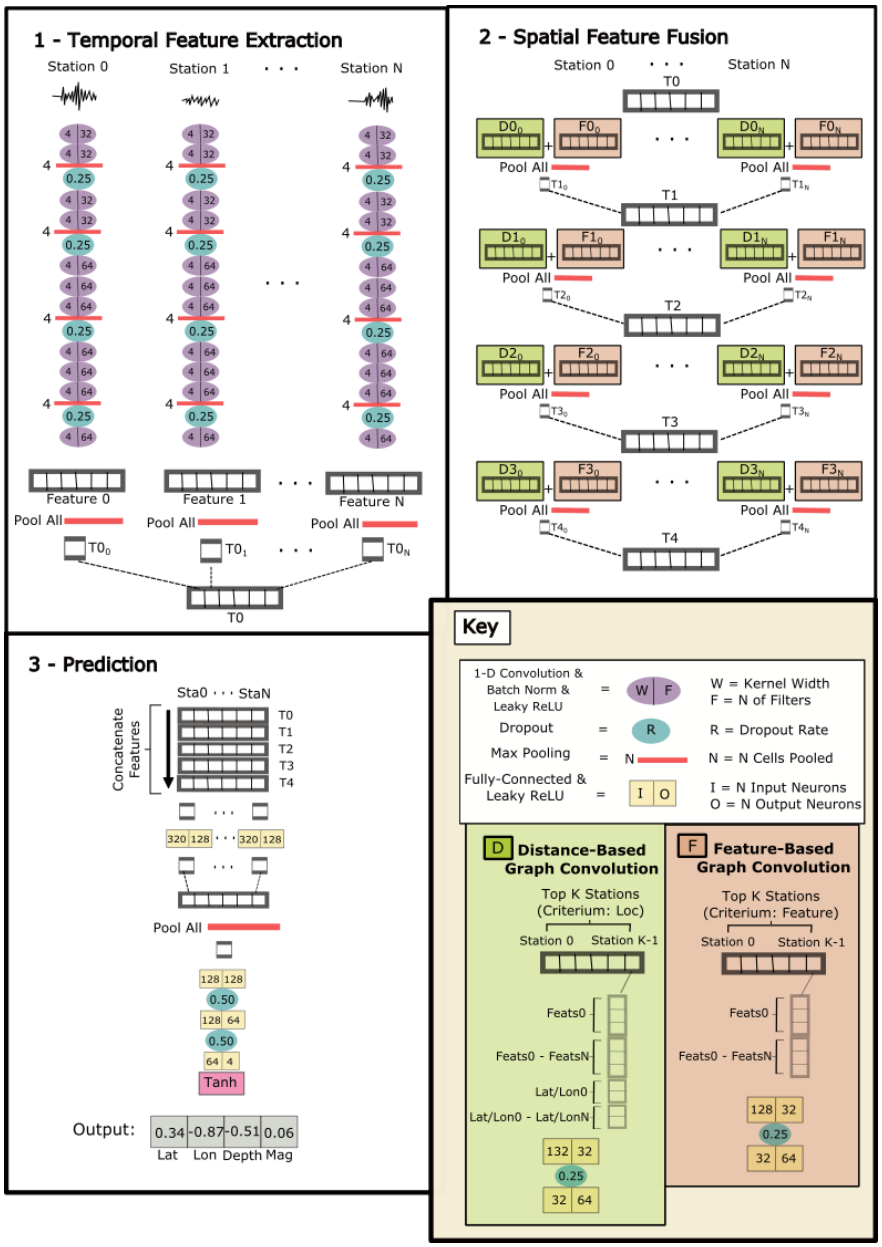
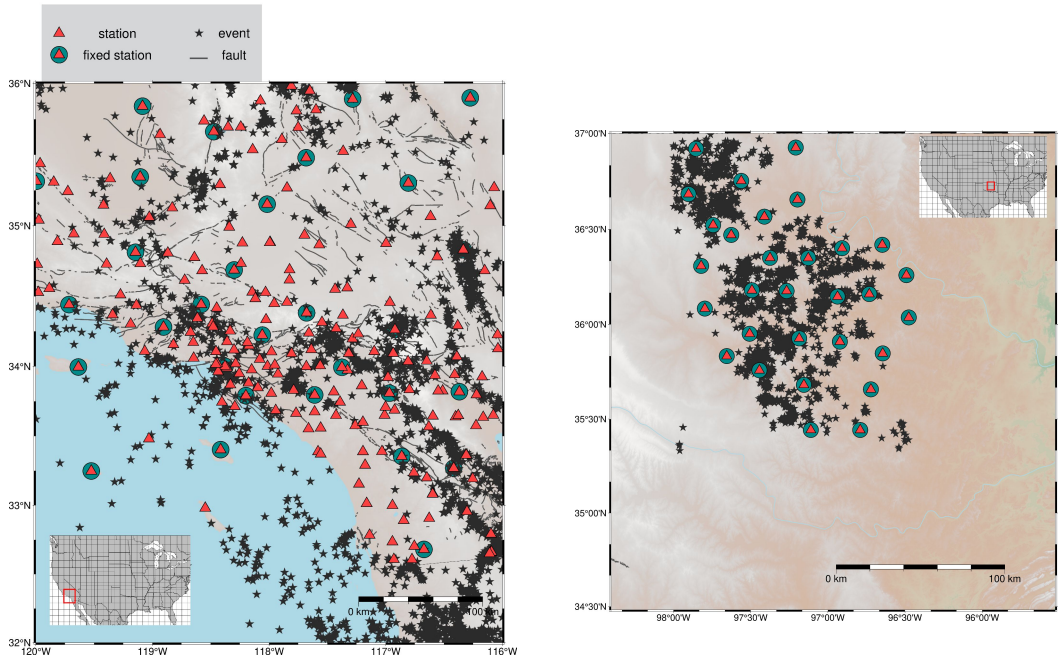


Figure 3.5 Overview of STGNN, including three components outlined in Figure 3.4. (1) In Temporal Feature Extraction, standard convolution and maximum pooling along the time dimension reduces 3-component waveforms to a feature vector of length 64. These feature vectors are concatenated to form T0. (2) In Spatial Feature Fusion, four layers of graph convolution are performed. In each layer, two graphs are constructed: Distance-Based (D), and Feature-Based (F), which are combined through element-wise summation, and max pooled along the K-nearest neighbors dimension. (3) In Prediction, feature tensors T0...T4 are concatenated and regressed to normalized predictions of latitude, longitude, depth, and magnitude. The pooling is applied along the station dimension.

and $-120°$ to $-116°$ longitude, a depth range of 0-30 km, and a magnitude range of $2.5 < M < 6$.

The final dataset contains $2,209$ events recorded by 185 broadband seismic stations.



(a) Southern California

(b) Oklahoma

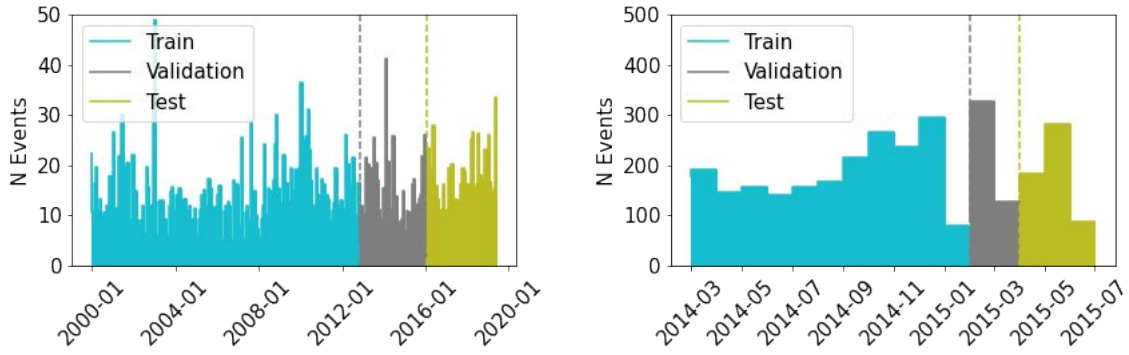
Figure 3.6 Maps of the two target regions used in this study: (a) Southern California and (b) Oklahoma. The distribution of all seismic stations (red triangles) and earthquakes (black stars) are shown. The 30 stations selected for fixed input testing are surrounded by a green circle. Black lines indicate seismic faults United States Geological Survey and California Geological Survey (2022).

The Oklahoma dataset uses waveforms and catalogue information collected by the Nanometrics Research Network, and was used for training and testing in the FCN baseline Zhang et al. (2020). The selected dataset contains events from March 2014 to July 2015 Nanometrics Seismological Instruments (2013) within a geographic subset from 34.482° to 37° latitude and −98.405° to −95.527° longitude, a depth range of 0-12 km, and a magnitude range of $1.5 < M < 4$. The final dataset contains $3,456$ events recorded by 30 broadband seismic stations.

All waveforms and catalogues were accessed using ObsPy Beyreuther et al. (2010). Each trace contains 200 sec of seismic displacement collected by three orthogonal channels, which is interpolated into $4,096$ evenly spaced samples, resulting in a sampling rate of approximately 20 Hz. For both datasets, the instrument response was removed, and waveforms were bandpass filtered from $1 - 8$ Hz. As the recorded displacement amplitudes are very small, the waveforms are multiplied by a constant scaling factor of $1e7$ to raise the input data to a numerically stable range close to $[-1, 1]$

without eliminating magnitude information. A map of events and stations is shown in Figure 3.6.

One advantage of the graph neural network is its ability to make predictions using dynamic inputs (i.e., the selected stations and their order in the input file are not necessarily the same for each sample). To demonstrate this ability, we perform tests with STGNN and the GNN baseline using Southern California data with dynamic inputs, in which functioning stations are randomly selected for each event. However, the FCN baseline requires a fixed input, in which the same stations must occupy the same position for each sample. To make a fair comparison, we train STGNN as well as both baselines on thirty fixed stations to compare the performance of all methods. This results in three datasets: (1) Dynamic Southern California Dataset, in which 100 stations are randomly selected for each sample, as well as (2) Fixed Southern California Dataset and (3) Fixed Oklahoma Dataset, in which a static set of 30 stations are used for every sample. For all datasets, events are omitted where < 25 stations are functioning.



(a) Southern California

(b) Oklahoma

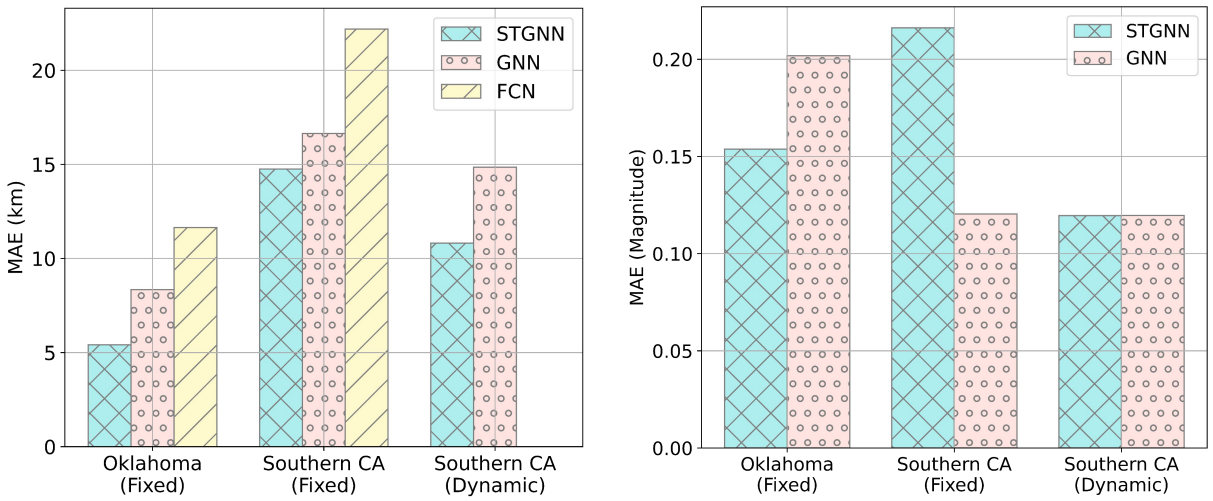
Figure 3.7 The monthly earthquake frequency distribution for (a) Southern California and (b) Oklahoma. The temporal boundaries between the training, validation, and testing data are indicated by color.

3.2.3.2 Training Procedure

In the experiments, we use AdamW as the optimizer with a learning rate of $3e{-}4$. The l_2 regularization term λ is $1e{-}4$. Models are trained for 400 epochs with early stopping after 50 epochs without validation error improvement, from which we select the model with the best validation performance. We use a 20-80 split to divide each dataset into testing and training data, and reserve

20% of the training data for validation. The datasets are not randomly shuffled, but rather separated by time in which training data precedes testing data. This approach avoids potential information leakage Kaufman et al. (2012) which might occur from spatially and temporally localized swarms. This method of splitting data also better simulates a real-use case, in which historic earthquakes would be used to train a model to detect more recent events on a network where station configuration and seismic characteristics may evolve over time. Figure 3.7 shows the monthly event frequency distribution in the training and testing dataset.

We use a sliding window with a length of 100 sec and a stride of 5 sec to create ten 100 sec samples from each 200 sec recording. This augments the data by increasing the sample size and cropping different portions of the wavetrain, assuring that the model can be used without known origin and arrival times. Within each sample, the same time shift is applied to all stations.



(a) MAE of Location Prediction

(b) MAE of Magnitude Prediction

Figure 3.8 (a) MAE of each tested model where the location error is measured in km. Location error refers to the euclidean distance between the predicted location and the true event location. (b) MAE of the magnitude predictions from the graph convolutional neural networks when applied to the Oklahoma dataset with 30 fixed stations, the Southern California dataset with 30 fixed stations, and the Southern California dataset with 100 dynamically selected stations.

3.2.3.3 Performance Comparison

To evaluate our developed framework, we compare the performance of our model against two baselines (1) van den Ende and Ampuero (2020) (referenced as GNN for graph neural network), and (2) Zhang et al. (2020) (referenced as FCN for fully convolutional network). The performance of each model is evaluated using the following metrics:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}|, \tag{3.11}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2, \tag{3.12}$$

$$\text{R}^2 = 1 - \sum_{i=1}^{n}\frac{(\widehat{y_i} - y_i)^2}{(y_i - \bar{y})^2}, \tag{3.13}$$

where $\widehat{y_i}$ is the model's prediction, y_i is the true value, $\bar{y}$ is the average true value, and n is the total number of predictions.

Both STGNN and GNN make normalized predictions between -1 and 1. When calculating the above metrics, the values are first reverted from the normalized scalars to degrees of latitude and longitude, kilometers of depth, and magnitude values. Degrees of latitude and longitude are then converted to kilometers using conversions of 110 km/degree and 92 km/degree, respectively.

Testing is conducted across three datasets:

1. *Dynamic Southern California Dataset:* The performance is tested for the STGNN and GNN models. Five neighbors (K=5) were selected for feature update for STGNN. Results are detailed in Table 3.3.

2. *Fixed Southern California Dataset:* The performance is tested for the STGNN, GNN, and FCN models. Seven neighbors (K=7) were selected for feature update for STGNN. Results are detailed in Table 3.4.

3. *Fixed Oklahoma Dataset:* The performance is tested for the STGNN, GNN, and FCN models. Seven neighbors (K=7) were selected for feature update for STGNN. Results are detailed in Table 3.5.

The performance overview (Figure 3.8) demonstrates that our proposed model achieves a higher location accuracy than baselines for all datasets. STGNN makes predictions with an average of 6.8 km less location error, a 40% improvement across all tested datasets when compared to the FCN baseline. Across all datasets, STGNN makes predictions with an average of 3.0 km less location error than the GNN baseline, a 22% improvement. The improved location is primarily due to epicentral location accuracy. All tested models demonstrate low R^2 values for depth prediction. STGNN and FCN achieve comparable magnitude prediction, and FCN does not support magnitude prediction. STGNN appears to incorporate a consistent bias, underpredicting magnitude values.

Figure 3.9, 3.10 and 3.11 plot all predictions to give a richer understanding of model capacity beyond individual quality metrics. Observation of individual predictions makes it clear that while both models succeed in learning a meaningful mapping to latitude and longitude predictions, depth predictions are highly scattered and are little better than predictions of the mean.

Table 3.3 Performance of STGNN and the GNN baseline when applied to the Southern California dataset with dynamic inputs. MAE refers to the mean absolute error (Equation 3.11) and MSE refers to the mean squared error (Equation 3.12), where a lower value indicates less error. The R^2 value (Equation 3.13) is a measure of how strongly variation in the predicted values are related to variation in the ground truth value, where a value close to 1 is indicative of high accuracy.

Latitude	MAE (km)	MSE (10^2 km)	R^2
STGNN	**7.548 ± 9.841**	**1.538 ± 9.255**	**0.980**
GNN	10.201 ± 11.791	2.431 ± 12.438	0.969
Longitude	MAE (km)	MSE (10^2 km)	R^2
STGNN	**6.931 ± 8.152**	**1.145 ± 5.589**	**0.986**
GNN	10.095 ± 12.086	2.480 ± 11.865	0.970
Depth	MAE (km)	MSE (10^2 km)	R^2
STGNN	**3.472 ± 2.928**	**0.206 ± 0.358**	**0.266**
GNN	3.837 ± 3.166	0.247 ± 0.399	0.120
Magnitude	MAE	MSE	R^2
STGNN	**0.120 ± 0.114**	**0.027 ± 0.085**	**0.826**
GNN	0.120 ± 0.126	0.030 ± 0.105	0.807

Table 3.4 Performance of STGNN, GNN and FCN baselines when applied to the Southern California dataset with fixed inputs. MAE refers to the mean absolute error (Equation 3.11) and MSE refers to the mean squared error (Equation 3.12), where a lower value indicates less error. The R^2 value (Equation 3.13) is a measure of how strongly variation in the predicted values are related to variation in the ground truth value, where a value close to 1 is indicative of high accuracy.

Latitude	MAE (km)	MSE (10^2 km)	R^2
STGNN	**10.396 ± 11.388**	**2.378 ± 10.067**	**0.954**
GNN	11.263 ± 11.696	2.637 ± 8.010	0.949
FCN	14.415 ± 21.827	6.842 ± 34.697	0.869
Longitude	MAE (km)	MSE (10^2 km)	R^2
STGNN	**9.663 ± 13.048**	**2.636 ± 15.761**	**0.962**
GNN	11.485 ± 12.199	2.807 ± 10.252	0.960
FCN	16.369 ± 24.872	8.865 ± 47.323	0.874
Depth	MAE (km)	MSE (10^2 km)	R^2
STGNN	**4.030 ± 3.396**	**0.278 ± 0.405**	**−0.069**
GNN	4.264 ± 3.384	0.296 ± 0.403	−0.141
FCN	4.105 ± 3.324	0.279 ± 0.431	-0.074
Magnitude	MAE	MSE	R^2
STGNN	0.216 ± 0.151	0.069 ± 0.083	0.583
GNN	**0.120 ± 0.118**	**0.028 ± 0.088**	**0.830**

3.2.3.4 Stability Analysis

There are three critical hyper-parameters in STGNN: the number of neighbors (K) considered for edge generation, the maximum amount of observed stations, and the random selection of seismic stations when creating datasets. We use the Southern California Dasaset to vary these hyperparameters in order to assess the stability of STGNN. The results of the paramater permutation are shown in Figure 3.12. When a parameter is not permuted, 100 stations, 5 edges, 4 graph convolutions, and a random seed of 0 is used.

For each prediction, a random subset of functional stations was selected. We permute the random seed during sample selection to alter the set of stations used for training. We find that the random subsets return similar results for all predictions except for magnitude, which shows a higher degree of variation. Prediction accuracy improves as more stations are used. Accuracy is moderately impacted by the number of edges selected, with magnitude predictions fluctuating most

Table 3.5 Performance of STGNN, GNN and FCN baselines when applied to the Oklahoma dataset with fixed inputs. MAE refers to the mean absolute error (Equation 3.11) and MSE refers to the mean squared error (Equation 3.12), where a lower value indicates less error. The R^2 value (Equation 3.13) is a measure of how strongly variation in the predicted values are related to variation in the ground truth value, where a value close to 1 is indicative of high accuracy.

Latitude	MAE (km)	MSE (10^2 km)	R^2
STGNN	**3.574 ± 5.755**	**0.459 ± 3.665**	**0.975**
GNN	7.166 ± 12.414	2.055 ± 14.820	0.897
FCN	9.219 ± 16.418	3.545 ± 23.070	0.822
Longitude	MAE (km)	MSE (10^2 km)	R^2
STGNN	**3.697 ± 4.936**	**0.380 ± 2.365**	**0.942**
GNN	5.934 ± 8.144	1.015 ± 5.547	0.904
FCN	9.308 ± 11.883	2.279 ± 8.244	0.785
Depth	MAE (km)	MSE (10^2 km)	R^2
STGNN	**1.686 ± 1.427**	**0.049 ± 0.082**	0.036
GNN	1.701 ± 1.423	0.049 ± 0.078	**0.090**
FCN	1.865 ± 1.546	0.059 ± 0.084	-0.086
Magnitude	MAE	MSE	R^2
STGNN	**0.154 ± 0.126**	**0.040 ± 0.066**	**0.790**
GNN	0.195 ± 0.142	0.058 ± 0.083	0.681

significantly. Overall, the model appears to be generally stable, with magnitude demonstrating the greatest sensitivity to hyperparameter tuning.

We also examine the impact of the number of GNN layers on the model's performance. A depth of four convolutions produces the best balance of location and magnitude accuracy, and is used for this study.

3.2.4 Discussion

3.2.4.1 Architecture Strengths and Weaknesses

Our STGNN has several advantages over the FCN baseline model. One of the primary advantages is the ability to make predictions on a dynamic set of inputs, allowing the model to adapt to station outages, network alterations, and station subsetting. As STGNN featurizes individual stations rather than an ordered network image, the model can be easily trained to predict using any number of stations without architectural alteration.

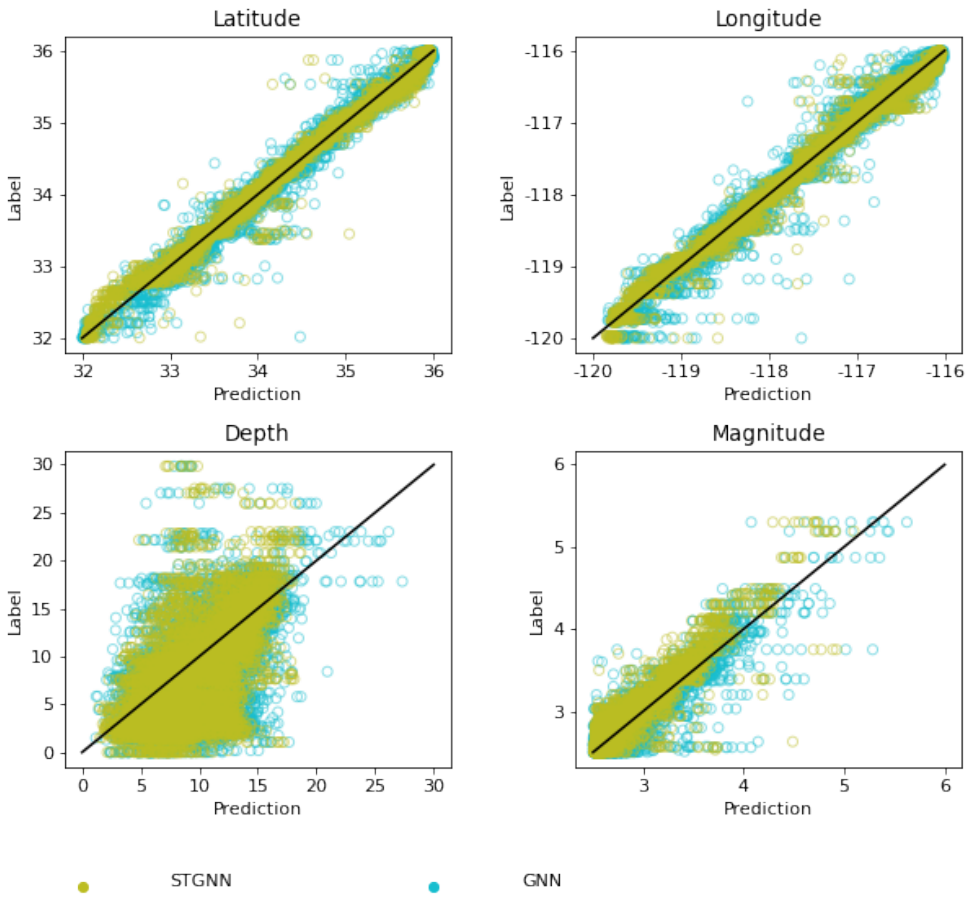


Figure 3.9 Testing comparison on 100 dynamically selected stations from the Southern California dataset with 5 convolutional edges. "STGNN" and "GNN" denote the performance of our framework and van den Ende and Ampuero (2020), respectively. In the scatter plot, each point represents an event, and a position on the diagonal line corresponds to perfect agreement between the predicted value (x-axis) and the true value (y-axis). Latitude and longitude values are displayed in degrees and depth values are displayed in kilometers.

The FCN baseline uses an image-to-image strategy, outputting a probability volume in which the highest values correspond to the event location. This has the advantage of predicting a probability amplitude, which Zhang et al. (2020) demonstrate as a useful measure of prediction uncertainty, especially in cases where earthquakes occur outside the bounds of the modeled region. However, the volumetric output comes at the cost of resolution limitation due to discretization. The gridded, three-dimensional output also requires a high degree of model complexity. The FCN baseline consequently comprises approximately 27 million parameters while our STGNN with scalar predictions comprises fewer than 0.24 million parameters.

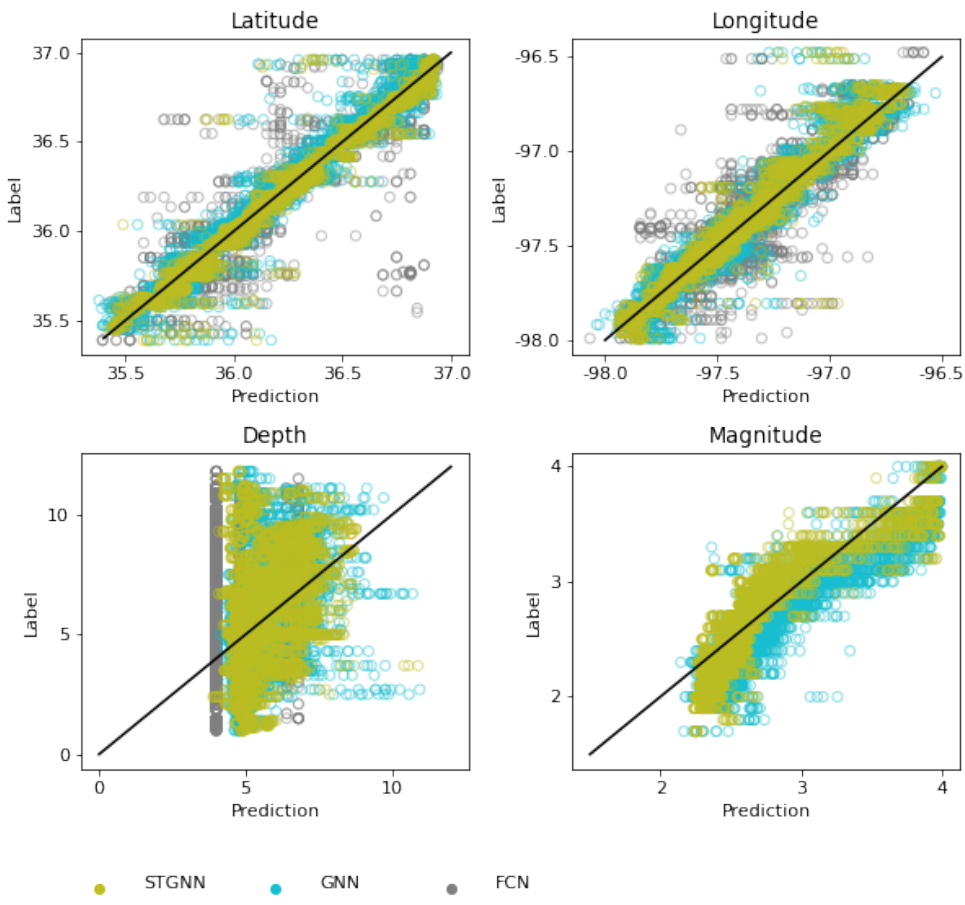


Figure 3.10 Testing comparison on 30 fixed stations from the Oklahoma dataset with 7 convolutional edges. "STGNN", "GNN", and "FCN" denote the performance of our framework, van den Ende and Ampuero (2020), and Zhang et al. (2020), respectively. In the scatter plot, each point represents an event, and a position on the diagonal line corresponds to perfect agreement between the predicted value (x-axis) and the true value (y-axis). Latitude and longitude values are displayed in degrees and depth values are displayed in kilometers. Magnitude is omitted for the FCN, as this model makes only location predictions.

The baseline GNN van den Ende and Ampuero (2020) implements edgeless graph convolution (i.e. station-by-station convolutions with global pooling) while the STGNN implements convolution and pooling over dynamically-generated edges. Figure 3.13 gives insight into the edge generation process. For clear visualization, we select a case with 50 seismic stations with $K = 5$. In the edges generated by waveform similarity, stations that have recorded an event are generally connected to other recording stations, forming distinct clusters from edges generated by geographic proximity. This indicates that the model is able to successfully extract waveform information and associate

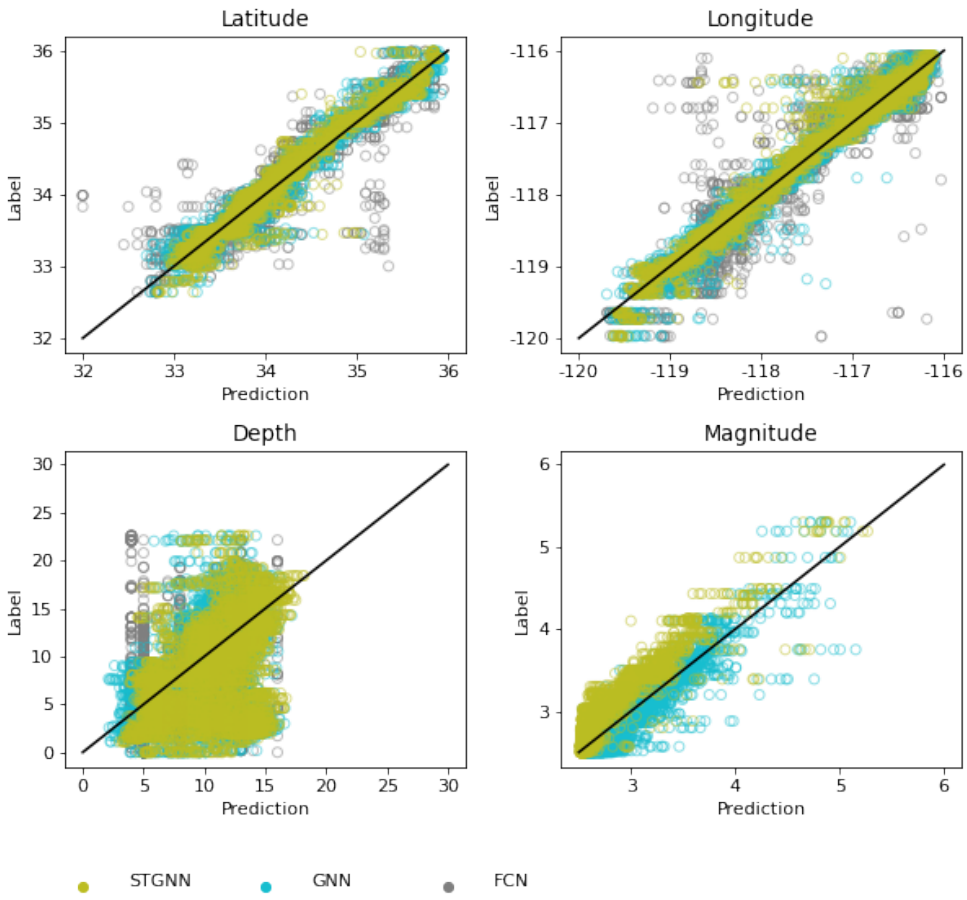
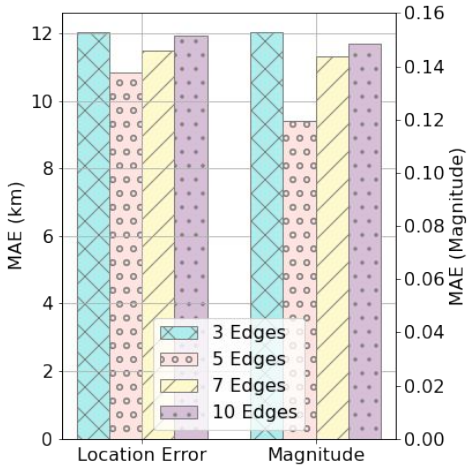


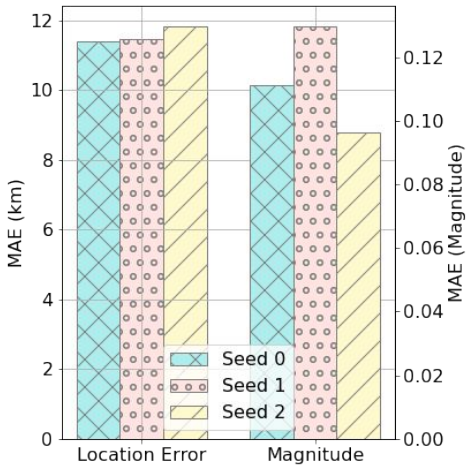
Figure 3.11 Testing comparison on 30 fixed stations from the Southern California dataset with 7 convolutional edges. "STGNN", "GNN", and "FCN" denote the performance of our framework, van den Ende and Ampuero (2020), and Zhang et al. (2020), respectively. In the scatter plot, each point represents an event, and a position on the diagonal line corresponds to perfect agreement between the predicted value (x-axis) and the true value (y-axis). Latitude and longitude values are displayed in degrees and depth values are displayed in kilometers. Magnitude is omitted for the FCN, as this model makes only location predictions.

stations in order to characterize an event. The graph generated based on the feature similarity is different from that created based on the location, showing that the feature similarity is a complement of location during aggregating features from different seismic stations.

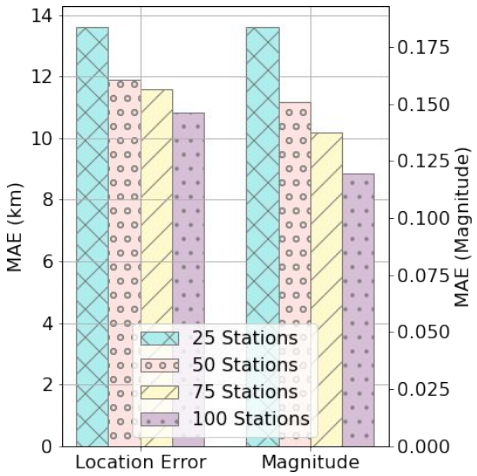
A limitation that STGNN shares with the baselines is the ability to make predictions only within a certain range of area, depth, and magnitude. The model outputs normalized values between -1 and 1 which correspond to a range selected at the beginning of training. The spatial restrictions are similar to the bounds set in inversion-based methods and are arguably less limiting, as the



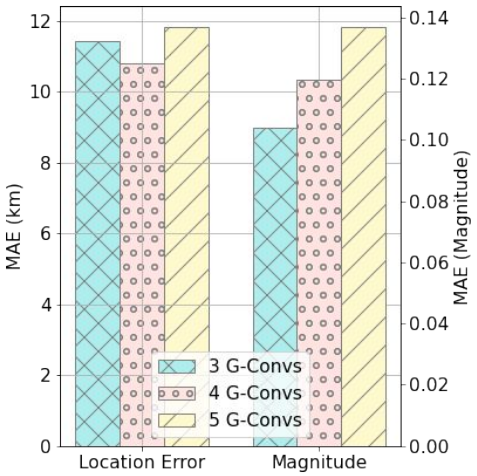
(a) Number of Edges



(b) Random Seed



(c) Number of Stations



(d) Number of Graph Convolutions

Figure 3.12 Stability analysis permuting (a) the number of edges used to connect nodes during graph convolution, (b) the random seed used to select stations for the model input, (c) the number of stations used for prediction, and (d) the number of graph convolutions implemented. When a parameter is not permuted, 100 stations, 5 edges, 4 graph convolutions, and a random seed of 0 is used.

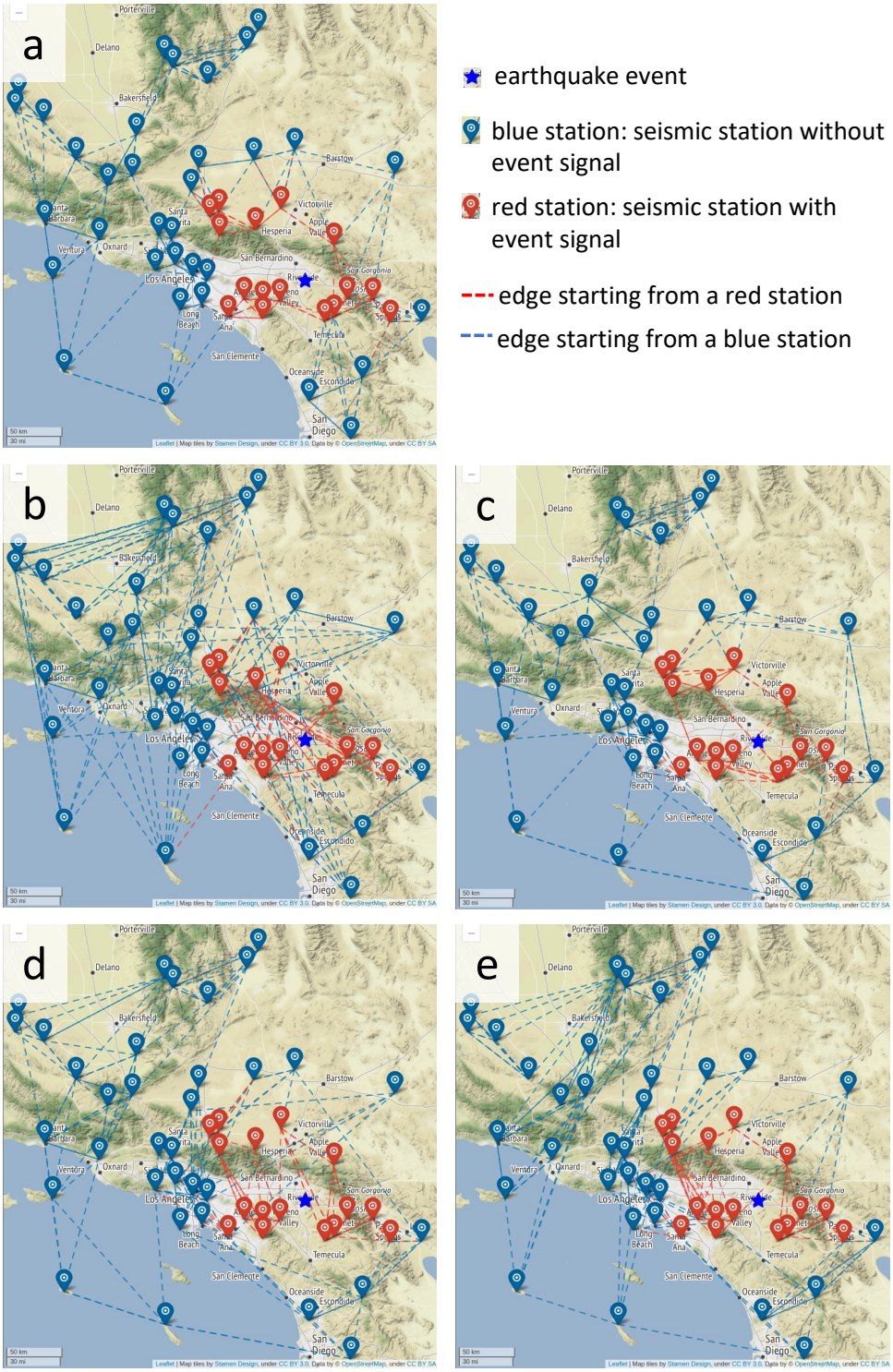


Figure 3.13 Graphs constructed by different layers of the graph neural network, (a) graph convolution layer based on geographic distance among seismic stations (b) 1st, (c) 2nd, (d) 3rd and (e) 4th graph convolution layer based on the extracted feature similarity. The information from stations with the event signal are clustered in deeper layers.

predictions made by our model are continuous and therefore not bound by grid-spacing.

However, STGNN is more limited than non-machine learning methods with regard to magnitude prediction. Magnitudes falling above or below the training range cannot be predicted by STGNN or the deep learning baselines. The limited range of predictions adversely impacts the usefulness of the deep learning methods for applications such as Earthquake Early Warning, where magnitude saturation must be avoided. The limitations posed by fixed prediction ranges are made less severe by STGNN's ability to be tuned to new ranges with small amounts of training data. However, the fixed prediction ranges nonetheless represent a weakness in our framework.

3.2.4.2 Impacts on Location Prediction

Overall location error for the STGNN model is 5.41 km for the Fixed Oklahoma Dataset and 14.75 km for the Fixed Southern California Dataset. The higher loss for the Southern California dataset may be attributable to the larger size of the region. As locations in both the smaller and larger regions are normalized to values between -1 and 1, errors in the initial prediction will result in larger errors when converted to kilometers in larger regions. In addition, larger regions may include a greater range of structural complexity that may be more challenging for the model to learn.

Location error for the Dynamic California Dataset was 3.93 km lower than that of the Fixed California Dataset. This supports the assumption that dynamic inputs improve not only the flexibility, but the performance of prediction models.

3.2.4.3 Synthetic Testing

While substantial improvements have been made in the prediction of latitude and longitude, magnitude does not improve in every dataset, and depth predictions are inaccurate for all models. To test the capacity of our model under ideal circumstances, we train our model using synthetic data. The synthetic waveforms are generated using Green's Functions created with PyFK Xi et al. (2021) from a 1-D sedimentary half-space model, with an epicentral resolution of 1 km and a depth resolution of 0.5 km and a sampling rate of 20.48 Hz. Thirty recording sensors are used with the same configuration as the Fixed Oklahoma Dataset. No label or waveform noise is applied. The high degree of accuracy suggests that the fundamental architecture of the model has the capacity to

learn depth and magnitude estimation (Fig 3.14). The difference between the simulated waveforms and the recorded data are (1) label noise, (2) waveform noise, and (3) subsurface complexity. While the fundamental structure holds promise, STGNN must be improved to address these factors before it can be effectively applied to real seismic datasets for depth and magnitude prediction.

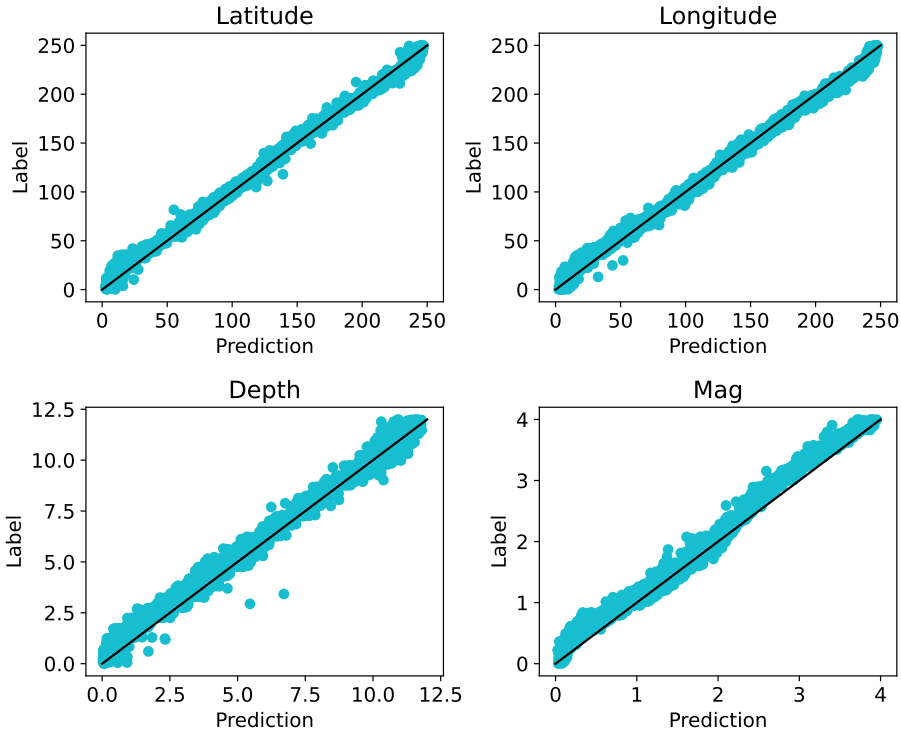


Figure 3.14 Testing performance of STGNN on synthetic data. In the scatter plot, each point represents an event, and a position on the diagonal line corresponds to perfect agreement between the predicted value (x-axis) and the true value (y-axis). Latitude and longitude values are displayed in degrees and depth values are displayed in kilometers.

3.2.5 Summary

This section has presented a graph convolutional neural network (GCNN) specifically designed for earthquake source characterization, utilizing waveform data from multiple seismic stations. This application leverages the regularization of the f term as defined in Equation (1.3). Through experimental validation in two distinct seismic environments, the Spatio-Temporal Graph Neural Network (STGNN) demonstrated superior performance over both fully-convolutional neural network (FCN) and traditional Graph Neural Network (GNN) baselines. A significant advantage of the

STGNN framework is its ability to dynamically learn feature generation and fusion processes directly from the data, thereby eliminating the need for static input types or manually predefined graph structures. This allows for an effective synthesis of waveform features and spatial data, enhancing the model's predictive accuracy and adaptability.

3.3 Unsupervised Learning of Full-Waveform Inversion: Connecting CNN and Partial Differential Equation in a Loop

In addition to designing regularization based on data structure as described in Sections 3.1 and 3.2, this section introduces a physics-informed machine learning architecture for full-waveform inversion, which is mathematically modeled by partial differential equations. By integrating governing physical equations with neural networks, we impose regularization through the f term in Equation (1.3). Furthermore, experimental results indicate that the perceptual loss Johnson et al. (2016) (represented as the R term in Equation (1.3)) significantly enhances seismic data reconstruction.

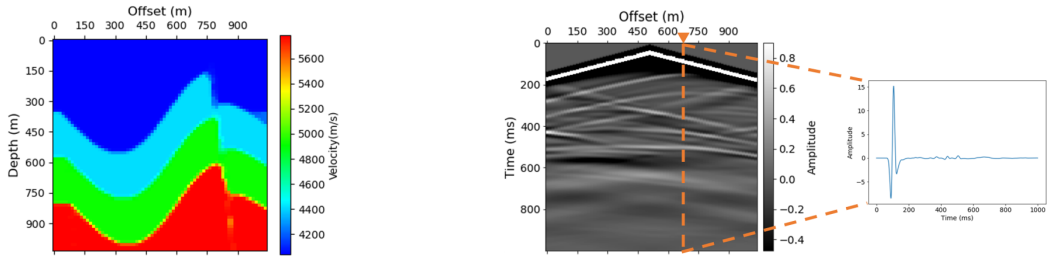
3.3.1 Introduction

Geophysical properties (such as velocity, impedance, and density) play an important role in various subsurface applications including subsurface energy exploration, carbon capture and sequestration, estimating pathways of subsurface contaminant transport, and earthquake early warning systems to provide critical alerts. These properties can be obtained via seismic surveys, i.e., receiving reflected/refracted seismic waves generated by a controlled source. This section focuses on reconstructing subsurface velocity maps from seismic measurements. Mathematically, the velocity map and seismic measurements are correlated through an acoustic-wave equation (a second-order partial differential equation) as follows:

$$\nabla^2 p(\boldsymbol{r}, t) - \frac{1}{v(\boldsymbol{r})^2} \frac{\partial^2 p(\boldsymbol{r}, t)}{\partial t^2} = s(\boldsymbol{r}, t) \ , \tag{3.14}$$

where $p(\boldsymbol{r}, t)$ denotes the pressure wavefield at spatial location $\boldsymbol{r}$ and time t, $v(\boldsymbol{r})$ represents the velocity map, and $s(\boldsymbol{r}, t)$ is the source term. Full-Waveform Inversion (FWI) is a methodology that determines high-resolution velocity maps $v(\boldsymbol{r})$ of subsurface via matching synthetic seismic waveforms to raw recorded seismic data $p(\tilde{\boldsymbol{r}}, t)$, where $\tilde{\boldsymbol{r}}$ represents the locations of seismic receivers.

A velocity map describes the wave propagation speed in the subsurface region of interest. An example in 2D scenario is shown in Figure 3.15a. Particularly, the x-axis represents the horizontal offset of a region, and the y-axis stands for the depth. The regions with the same geologic information (velocity) are called a layer in velocity maps. In a sample of seismic measurements (termed a shot gather in geophysics) as depicted in Figure 3.15b, each grid in the



(a) (b)

Figure 3.15 An example of (a) a velocity map and (b) seismic measurements (named shot gather in geophysics) and the 1D time-series signal recorded by a receiver.

x-axis represents a receiver, and the value in the y-axis is a 1D time-series signal recorded by each receiver.

Existing approaches solve FWI in two directions: *physics-driven* and *data-driven*. Physics-driven approaches rely on the forward modeling of Equation 3.14, which simulates seismic data from velocity map by finite difference. They optimize velocity map per seismic sample, by iteratively updating velocity map from an initial guess such that simulated seismic data (after forward modeling) is close to the input seismic measurements. However, these methods are slow and difficult to scale up as the iterative optimization is required per input sample. Data-driven approaches consider FWI problem as an image-to-image translation task and apply convolution neural networks (CNN) to learn the mapping from seismic data to velocity maps (Wu and Lin, 2019). The limitation of these methods is that they require paired seismic data and velocity maps to train the network. Such ground truth velocity maps are hardly accessible in real-world scenario because generating them is extremely time-consuming even for domain experts.

In this work, we leverage advantages of both directions (physics + data driven) and shift the paradigm to unsupervised learning of FWI by connecting forward modeling and CNN in a loop. Specifically, as shown in Figure 3.16, a CNN is trained to predict a velocity map from seismic data, which is followed by forward modeling to reconstruct seismic data. The loop is closed by applying reconstruction loss on seismic data to train the CNN. Due to the differentiable forward modeling, the whole loop can be trained end-to-end. Note that the CNN is trained in an unsupervised manner, as the ground truth of velocity map is *not* needed. We name our unsupervised approach as UPFWI

(Unsupervised Physics-informed Full-Waveform Inversion).

Additionally, we find that perceptual loss (Johnson et al., 2016) is crucial to improve the overall quality of predicted velocity maps due to its superior capability in preserving the coherence of the reconstructed waveforms comparing with other losses like Mean Squared Error (MSE) and Mean Absolute Error (MAE).

To encourage fair comparison on a large dataset with more complicate geological structures, we introduce a new synthetic dataset named OpenFWI, which contains 60,000 labeled data (velocity map and seismic data pairs) and 48,000 unlabeled data (seismic data alone). 30,000 of those velocity maps contain curved layers that are more challenge for inversion. We also add geological faults with various shift distances and tilting angles to all velocity maps.

We evaluate our method on this dataset. Experimental results show that for velocity maps with flat layers, our UPFWI trained with 48,000 unlabeled data achieves 1146.09 in MSE, which is 26.77% smaller than that of the supervised baseline H-PGNN+ (Sun et al., 2021), and 0.9895 in Structured Similarity (SSIM), which is 0.0021 higher; for velocity maps with curved layers, our UPFWI achieves 3639.96 in MSE, which is 28.30% smaller, and 0.9756 in SSIM, which is 0.0057 higher.

Our contribution is summarized as follows:

- We propose to solve FWI in an unsupervised manner by connecting CNN and forward modeling in a loop, enabling end-to-end learning from seismic data alone.
- We find that perceptual loss is helpful to boost the performance comparable to the supervised counterpart.
- We introduce a large-scale dataset as benchmark to encourage further research on FWI.

3.3.2 Preliminaries of Full-Waveform Inversion (FWI)

The goal of FWI in geophysics is to invert for a velocity map $\boldsymbol{v} \in \mathbb{R}^{W \times H}$ from seismic measurements $\boldsymbol{p} \in \mathbb{R}^{S \times T \times R}$, where W and H denote the horizontal and vertical dimensions of the velocity map, S is the number of sources used to generate waves during data acquisition process, T

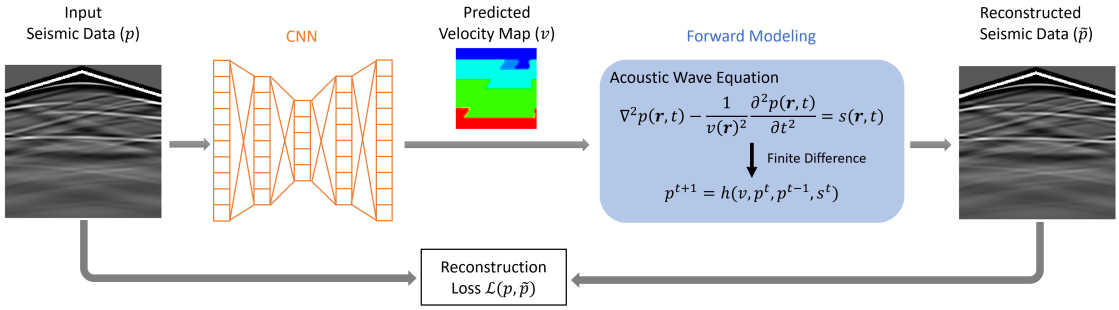


Figure 3.16 Schematic illustration of our proposed method, which comprises a CNN to learn an inverse mapping and a differentiable operator to approximate the forward modeling of PDE.

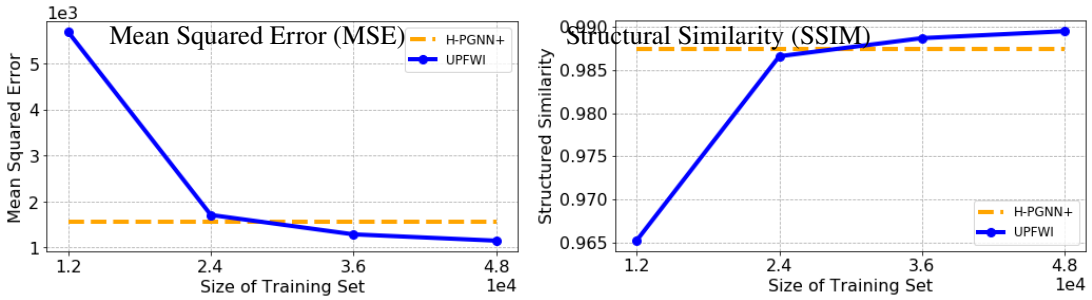


Figure 3.17 Unsupervised UPFWI (ours) vs. Supervised H-PGNN+ (Sun et al., 2021). Our method achieves better performance, e.g. lower Mean Squared Error (MSE) and higher Structural Similarity (SSIM), when involving more unlabeled data (>24k).

denotes the number of samples in the wavefields recorded by each receiver, and R represents the total number of receivers.

In conventional physics-driven methods, forward modeling is commonly referred to the process of simulating seismic data $\tilde{\boldsymbol{p}}$ from given estimated velocity maps $\hat{\boldsymbol{v}}$. For simplicity, the forward acoustic-wave operator f can be expressed as

$$\tilde{\boldsymbol{p}} = f(\hat{\boldsymbol{v}}) \ . \tag{3.15}$$

Given this forward operator f, the physics-driven FWI can be posed as a minimization problem (Virieux and Operto, 2009)

$$E(\hat{\boldsymbol{v}}) = \min_{\hat{\boldsymbol{v}}} \left\{ ||\boldsymbol{p} - f(\hat{\boldsymbol{v}})||_2^2 + \lambda R(\hat{\boldsymbol{v}}) \right\} , \tag{3.16}$$

where $||\boldsymbol{p} - f(\hat{\boldsymbol{v}})||_2^2$ is the the ℓ_2 distance between true seismic measurements $\boldsymbol{p}$ and the corresponding simulated data $f(\hat{\boldsymbol{v}})$, λ is a regularization parameter and $R(\hat{\boldsymbol{v}})$ is the regularization term which is

often the ℓ_2 or ℓ_1 norm of $\hat{\boldsymbol{v}}$. This requires optimization per sample, which is slow as the optimization involves multiple iterations from an initial guess.

Data-driven methods leverage CNNs to directly learn the inverse mapping as (Adler et al., 2021)

$$\hat{\boldsymbol{v}} = g_\theta(\boldsymbol{p}) \approx f^{-1}(\boldsymbol{p}) \ , \tag{3.17}$$

where $g_\theta(\cdot)$ is the approximated inverse operator of $f(\cdot)$ parameterized by θ. In practice, g_θ is usually implemented as a CNN (Adler et al., 2021; Wu and Lin, 2019). This requires paired seismic data and velocity maps for supervised learning. However, the acquisition of large volume of velocity maps in field applications can be extremely challenging and computationally prohibitive.

3.3.3 Related Work

Physics-driven Methods: In the past few decades, many regularization techniques have been proposed to alleviate the ill-posedness and non-linearity of FWI (Hu et al., 2009; Burstedde and Ghattas, 2009; Ramírez and Lewis, 2010; Lin and Huang, 2017, 2015b,a; Guitton, 2012; Treister and Haber, 2016). Other researchers focused on multi-scale techniques and decomposed the data into different frequency bands (Bunks et al., 1995; Boonyasiriwat et al., 2009).

Data-driven Methods: Recently, some researchers employed neural networks to solve FWI. Those methods can be further divided into supervised and unsupervised methods.

Supervised: One type of supervised methods require labeled samples to directly learn the inverse mapping, and they can be formulated as:

$$\hat{\boldsymbol{v}}(\boldsymbol{p}) = g_{\theta^*}(\boldsymbol{p}) \text{ s.t. } \theta^*(\boldsymbol{\Phi}_s) = \arg\min_\theta \sum_{\{\boldsymbol{v}_i, \boldsymbol{p}_i\} \in \boldsymbol{\Phi}_s} \mathcal{L}(g_\theta(\boldsymbol{p_i}),\ \boldsymbol{v}_i), \tag{3.18}$$

where $\boldsymbol{p}$ denotes the seismic measurements, $\boldsymbol{v}$ is the velocity map, θ represents the trainable weights in the inversion network $g_\theta(\cdot)$, $f(\cdot)$ is the forward modeling, and $\mathcal{L}(\cdot, \cdot)$ is a loss function. One example of supervised methods is the fully connected network proposed by Araya-Polo et al. (2018). Wu and Lin (2019) developed an encoder-decoder structured network to handle more complex velocity maps. Zhang and Lin (2020) adopted GAN and transfer learning to improve generalizability. Li et al. (2020b) designed SeisInvNet to solve misaligned issue when dealing sources from different locations. In Yang and Ma (2019), a U-Net architecture was proposed with

skip connections. Feng et al. (2021) proposed a multi-scale framework by considering different frequency components. Rojas-Gómez et al. (2020) developed an adaptive data augmentation method to improvegeneralizability. Sun et al. (2021) combined the data-driven and physics-based methods and proposed H-PGNN model.

Another type of supervised methods GANs to learn a distribution from velocity maps in training set as a prior (Richardson, 2018; Mosser et al., 2020). They can be formulated as:

$$\begin{aligned} \hat{\boldsymbol{v}}(\boldsymbol{z}^*) = g_{\theta^*}(\boldsymbol{z}^*) \text{ s.t. } \boldsymbol{z}^*(\boldsymbol{p}) = \arg\min_{\boldsymbol{z}} \mathcal{L}(f(g_{\theta^*}(\boldsymbol{z})), \boldsymbol{p}), \\ \theta^*(\boldsymbol{\Phi_v}) = \arg\min_{\theta} \sum_{v_i \in \Phi_v} \mathcal{L}_{\text{GAN}}(g_\theta(\boldsymbol{\alpha_i}), \boldsymbol{v_i}), \end{aligned} \tag{3.19}$$

where $\boldsymbol{\Phi_v}$ is a training dataset including numerous velocity maps. $\boldsymbol{z}$ and $\boldsymbol{\alpha_i}$ are tensors sampled from the normal distribution. The iterative optimization is then performed on $\boldsymbol{z}$ to draw a velocity map sampled from the prior distribution.

Unsupervised: The existing unsupervised methods follow the iterative optimization paradigm and perform FWI per sample. They employ neural networks to reparameterize velocity maps. The networks serve as an implicit regularization and are required to be pretrained on an expert initial guess. Those methods can be formulated as:

$$\hat{\boldsymbol{v}}(\boldsymbol{p}) = g_{\theta^*(\boldsymbol{p})}(\boldsymbol{a}) \text{ s.t. } \theta^*(\boldsymbol{p}) = \arg\min_{\theta} \mathcal{L}(f(g_\theta(\boldsymbol{a})), \boldsymbol{p}), \tag{3.20}$$

where $\boldsymbol{a}$ is a random tensor. Different network architectures have been proposed including CNN-domain FWI (Wu and McMechan, 2019) and DNN-FWI (He and Wang, 2021). Zhu et al. (2021) developed NNFWI which does not need pretraining ahead, but the initial guess is still required to be fed into the PDE with estimated velocity maps.

3.3.4 Method

In this section, we present our Unsupervised Physics-informed solution (named UPFWI), which connects CNN and forward modeling in a loop. It addresses limitations of both physics-driven and data-driven approaches, as it requires neither optimization at inference (per sample), nor velocity maps as supervision.

3.3.4.1 UPFWI: Connecting CNN and Forward Modeling

As depicted in Figure 3.16, our UPFWI connects a CNN g_θ and a differentiable forward operator f to form a loop. In particular, the CNN takes seismic measurements $\boldsymbol{p}$ as input and generates the corresponding velocity map $\hat{\boldsymbol{v}}$. We then apply forward acoustic-wave operator f (see Equation 3.15) on the estimated velocity map $\hat{\boldsymbol{v}}$ to reconstruct the seismic data $\tilde{\boldsymbol{p}}$. Typically, the forward modeling employs finite difference (FD) to discretize the wave equation (Equation 3.14). The details of forward modeling will be discussed Section 3.3.4.3. The loop is closed by the reconstruction loss between input seismic data $\boldsymbol{p}$ and reconstructed seismic data $\tilde{\boldsymbol{p}} = f(g_\theta(\boldsymbol{p}))$. Notice that the ground truth of the velocity map $\boldsymbol{v}$ is not involved, and the training process is *unsupervised*. Since the forward operator is differentiable, the reconstruction loss can be backpropagated (via gradient descent) to update the parameters θ in the CNN.

3.3.4.2 CNN Network Architecture

We use an encoder-decoder structured CNN (similar to Wu and Lin (2019) and Zhang and Lin (2020)) to model the mapping from seismic data $\boldsymbol{p} \in \mathbb{R}^{S\times T\times R}$ to velocity map $\boldsymbol{v} \in \mathbb{R}^{W\times H}$. The encoder compresses the seismic input and then transforms the latent vector to build the velocity estimation through a decoder. See the implementation details in Appendix C.1.

3.3.4.3 Differentiable Forward Modeling

We apply the standard finite difference (FD) in the space domain and time domain to discretize the original wave equation. Specifically, the second-order central finite difference in time domain ($\frac{\partial^2 p(\boldsymbol{r},t)}{\partial t^2}$ in Equation 3.14) is approximated as follows:

$$\frac{\partial^2 p(\boldsymbol{r},t)}{\partial t^2} \approx \frac{1}{(\Delta t)^2}(p_{\boldsymbol{r}}^{t+1} - 2p_{\boldsymbol{r}}^{t} + p_{\boldsymbol{r}}^{t-1}) + O[(\Delta t)^2] \,, \tag{3.21}$$

where $p_{\boldsymbol{r}}^t$ denotes the pressure wavefields at timestep t, and $p_{\boldsymbol{r}}^{t+1}$ and $p_{\boldsymbol{r}}^{t-1}$ are the wavefields at $t+\Delta t$ and $t-\Delta t$, respectively. The Laplacian of $p(\boldsymbol{r},t)$ can be estimated in the similar way on the space domain (see Appendix C.2). Therefore, the wave equation can then be written as

$$p_{\boldsymbol{r}}^{t+1} = (2 - v^2\nabla^2)p_{\boldsymbol{r}}^{t} - p_{\boldsymbol{r}}^{t-1} - v^2(\Delta t)^2 s_{\boldsymbol{r}}^{t} \,, \tag{3.22}$$

where ∇^2 here denotes the discrete Laplace operator.

The initial wavefield at the initial timestep is set to zero (i.e. $p_{\boldsymbol{r}}^0 = 0$). Thus, the gradient of loss $\mathcal{L}$ with respect to estimated velocity at spatial location $\boldsymbol{r}$ can be computed using the chain rule as

$$\frac{\partial \mathcal{L}}{\partial v(\boldsymbol{r})} = \sum_{t=0}^{T} \left[\frac{\partial \mathcal{L}}{\partial p(\boldsymbol{r}, t)} \right] \frac{\partial p(\boldsymbol{r}, t)}{\partial v(\boldsymbol{r})} , \tag{3.23}$$

where T indicates the total number of timesteps.

3.3.4.4 Loss Function

The reconstruction loss of our UPFWI includes a pixel-wise loss and a perceptual loss as follows:

$$\mathcal{L}(\boldsymbol{p}, \tilde{\boldsymbol{p}}) = \mathcal{L}_{pixel}(\boldsymbol{p}, \tilde{\boldsymbol{p}}) + \mathcal{L}_{perceptual}(\boldsymbol{p}, \tilde{\boldsymbol{p}}), \tag{3.24}$$

where $\boldsymbol{p}$ and $\tilde{\boldsymbol{p}}$ are input and reconstructed seismic data, respectively. The pixel-wise loss $\mathcal{L}_{pixel}$ combines ℓ_1 and ℓ_2 distance as:

$$\mathcal{L}_{pixel}(\boldsymbol{p}, \tilde{\boldsymbol{p}}) = \lambda_1 \ell_1(\boldsymbol{p}, \tilde{\boldsymbol{p}}) + \lambda_2 \ell_2(\boldsymbol{p}, \tilde{\boldsymbol{p}}), \tag{3.25}$$

where λ_1 and λ_2 are two hyper-parameters to control the relative importance. For the perceptual loss $\mathcal{L}_{perceptual}$, we extract features from `conv5` in a VGG-16 network (Simonyan and Zisserman, 2014) pretrained on ImageNet (Krizhevsky et al., 2012) and combine the ℓ_1 and ℓ_2 distance as:

$$\mathcal{L}_{perceptual}(\boldsymbol{p}, \tilde{\boldsymbol{p}}) = \lambda_3 \ell_1(\phi(\boldsymbol{p}), \phi(\tilde{\boldsymbol{p}})) + \lambda_4 \ell_2(\phi(\boldsymbol{p}), \phi(\tilde{\boldsymbol{p}})), \tag{3.26}$$

where $\phi(\cdot)$ represents the output of `conv5` in the VGG-16 network, and λ_3 and λ_4 are two hyper-parameters. Compared to the pixel-wise loss, the perceptual loss is better to capture the region-wise structure, which reflects the waveform coherence. This is crucial to boost the overall accuracy of velocity maps (e.g. the quantitative velocity values and the structural information).

3.3.5 OpenFWI Dataset

We introduce a new large-scale geophysics FWI dataset OpenFWI, which consists of 108K seismic data for two types of velocity maps: one with flat layers (named FlatFault) and the other one with curved layers (named CurvedFault). Each type has 54K seismic data, including 30K with paired velocity maps (labeled) and 24K unlabeled. The 30K labeled pairs are splitted as 24K/3K/3K for training, validation and testing respectively. Samples are shown in Appendix C.3.

The shape of curves in our dataset follows a sine function. Velocity maps in CurvedFault are designed to validate the effectiveness of FWI methods on curved topography. Compared to the maps with flat layers, curved velocity maps yield much more irregular geological structures, making inversion more challenging. Both FlatFault and CurvedFault contain 30,000 samples with 2 to 4 layers and their corresponding seismic data. Each velocity map has dimensions of 70×70, and the grid size is 15 meter in both directions. The layer thickness ranges from 15 grids to 35 grids, and the velocity in each layer is randomly sampled from a uniform distribution between 3,000 meter/second and 6,000 meter/second. The velocity is designed to increase with depth to be more physically realistic. We also add geological faults to every velocity map. The faults shift from 10 grids to 20 grids, and the tilting angle ranges from -123 to 123 degrees.

To synthesize seismic data, five sources are evenly placed on surface with a 255-meter spacing, and seismic traces are recorded by 70 receivers at each grid with an interval of 15 meter. The source is a Ricker wavelet with a central frequency of 25 Hz (Wang, 2015). Each receiver records time-series data for 1 second, and we use a 1 millisecond sample rate to generate 1,000 timesteps. Therefore, the dimensions of seismic data become 5×1000×70. Compared to existing datasets (Yang and Ma, 2019; Moseley et al., 2020), OpenFWI is significantly larger. It includes more complicated and physically realistic velocity maps. We hope it establishes a more challenging benchmark for the community.

3.3.6 Experiments

In this section, we present experimental results of our proposed UPFWI evaluated on the OpenFWI.

3.3.6.1 Implementation Details

Training Details: The input seismic data are normalized to range $[-1, 1]$. We employ AdamW (Loshchilov and Hutter, 2018) optimizer with momentum parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a weight decay of 1×10^{-4} to update all parameters of the network. The initial learning rate is set to 3.2×10^{-4}, and we reduce the learning rate by a factor of 10 when validation loss reaches a plateau. The minimum learning rate is set to 3.2×10^{-6}. The size of a mini-batch is set to 128. All

trade-off hyper-parameters λ in our loss function are set to 1. We implement our models in Pytorch and train them on 8 NVIDIA Tesla V100 GPUs. All models are randomly initialized.

Evaluation Metrics: We consider three metrics for evaluating the velocity maps inverted by our method: MAE, MSE and SSIM. Both MAE and MSE have been employed in existing methods (Wu and Lin, 2019; Zhang and Lin, 2020) to measure pixel-wise errors. Considering the layered-structured velocity maps contain highly structured information, degradation or distortion can be easily perceived by a human. To better align with human vision, we employ SSIM to measure perceptual similarity. Note that for MAE and MSE calculation, we denormalize velocity maps to their original scale while we keep them in normalized scale [-1, 1] for SSIM according to the algorithm.

Comparison: We compare our method with three state-of-the-art algorithms: two pure data-driven methods, i.e., InversionNet (Wu and Lin, 2019) and VelocityGAN (Zhang and Lin, 2020), and a physics-informed method H-PGNN (Sun et al., 2021). We follow the implementation described in these papers and search for the best hyper-parameters for OpenFWI dataset. Note that we improve H-PGNN by replacing the network architecture with the CNN in our UPFWI and adding perceptual loss, resulting in a significant boosted performance. We refer our implementation as H-PGNN+, which is a strong supervised baseline. Our method has two variants (UPFWI-24K and UPFWI-48K), using 24K and 48K unlabeled seismic data respectively.

3.3.6.2 Main Results

Results on FlatFault: Table 3.6 shows the results of different methods on FlatFault. Compared to InversionNet and VelocityGAN, our UPFWI-24K performs better in MSE and SSIM, but is slightly worse in MAE score. Compared to H-PGNN+, there is a gap between our UPFWI-24K and H-PGNN+ when trained with the same amount of data. However, after we double the size of unlabeled data (from 24K to 48K), a significant improvement is observed in our UPFWI-48K for all three metrics, and it outperforms all three supervised baselines in MSE and SSIM. This demonstrates the potential of our UPFWI for achieving higher performance with more unlabeled data involved.

Table 3.6 Quantitative results evaluated on OpenFWI in terms of MAE, MSE and SSIM. Our UPFWI yields comparable inversion accuracy comparing to supervised baselines. For H-PGNN+, we use our network architecture to replace the original one reported in their paper, and an additional perceptual loss between seismic data is added during training.

Supervision	Method	FlatFault			CurvedFault		
		MAE ↓	MSE ↓	SSIM ↑	MAE ↓	MSE ↓	SSIM ↑
Supervised	InversionNet	15.83	2156.00	0.9832	23.77	5285.38	0.9681
	VelocityGAN	16.15	1770.31	0.9857	25.83	5076.79	0.9699
	H-PGNN+	**12.91**	1565.02	0.9874	24.19	5139.60	0.9685
Unsupervised	**UPFWI-24K**	16.27	1705.35	0.9866	29.59	5712.25	0.9652
	UPFWI-48K	14.60	**1146.09**	**0.9895**	**23.56**	**3639.96**	**0.9756**

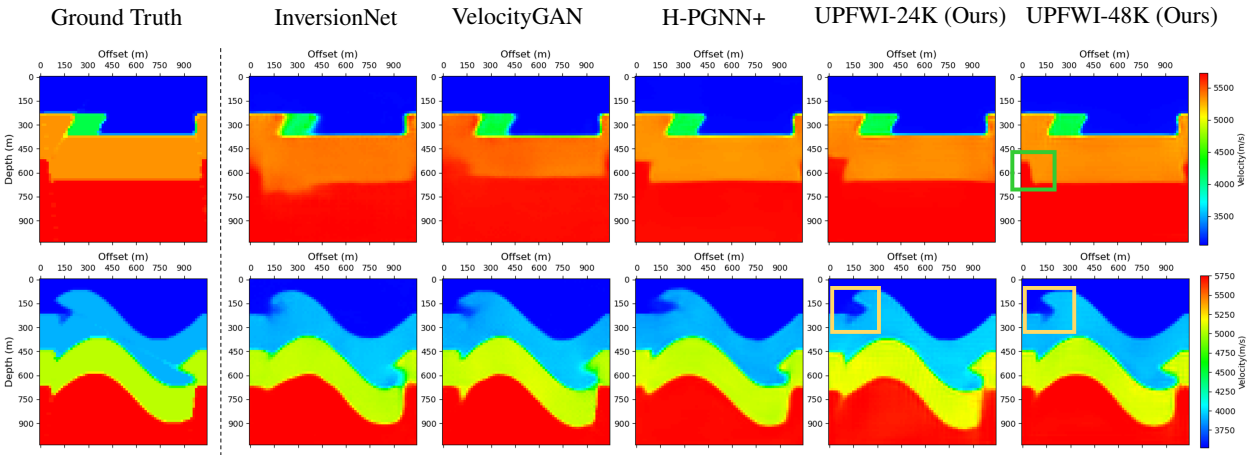
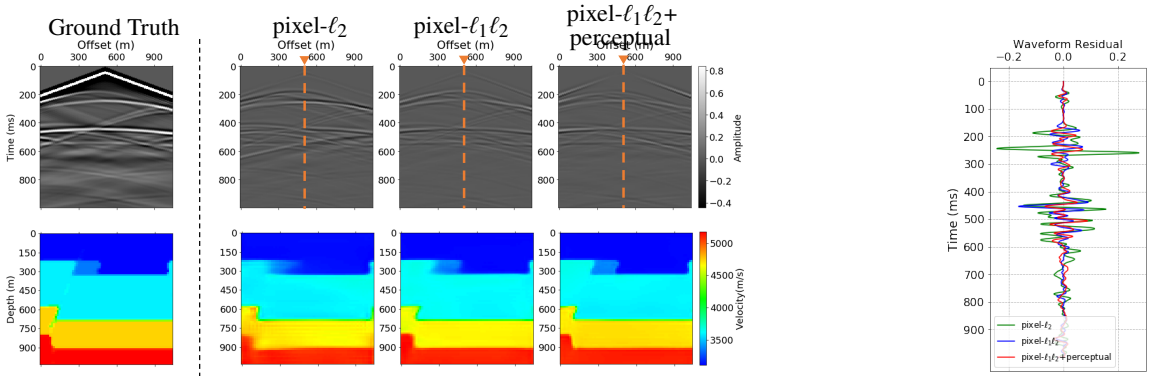


Figure 3.18 Comparison of different methods on inverted velocity maps of FlatFault (top) and CurvedFault (bottom). For FlatFault, our UPFWI-48K reveals more accurate details at layer boundaries and the slope of the fault in deep region. For CurvedFault, our UPFWI reconstructs the geological anomalies on the surface that best match the ground truth.

The velocity maps inverted by different methods are shown in the top row of Figure 3.18. Consistent with our quantitative analysis, more accurate details are observed in the velocity maps generated by UPFWI-48K. For instance, we find in the visualization results that both InversionNet and VelocityGAN generate blurry results in deep region, while H-PGNN+, UPFWI-24K and UPFWI-48K yield much clearer boundaries. We attribute this finding as the impact of seismic loss. We further observe that the slope of the fault in deep region is different from that in the shallow



(a) (b)

Figure 3.19 Comparison of UPFWI with different loss functions on (a) waveform residual and their corresponding inversion results (ground truth provided in the first column), and (b) single trace residuals recorded by the receiver at 525 m offset. Our UPFWI trained with pixel-wise loss (ℓ_1+ℓ_2 distance) and perceptual loss yields the most accurate results. Best viewed in color.

region, yet only UPFWI-48K replicates this result as highlighted by the green square.

Results on CurvedFault: Table 3.6 shows the results of CurvedFault. Performance degradation is observed for all models, due to the more complicated geological structures in CurvedFault. Although our UPFWI-24K underperforms the three supervised baselines, our UPFWI-48K significantly boosts the performance, outperforming all supervised methods in terms of all three metrics. This demonstrates the power of unsupervised learning in our UPFWI that greatly benefits from more unlabeled data when dealing with more complicated curved structure.

The bottom row of Figure 3.18 shows the visualized velocity maps in CurvedFault obtained using different methods. Similar to the observation in FlatFault, our UPFWI-48K yields more accurate details compared to the results of supervised methods. For instance, only our UPFWI-24K and UPFWI-48K precisely reconstruct the fault beneath the curve around the top-left corner as highlighted by the yellow square. Although some artifacts are observed in the results of UPFWI-24K around the layer boundary in deep region, they are eliminated in the results of UPFWI-48K. More visualization results are shown in Appendix C.3.

3.3.6.3 Ablation Study

Loss Terms: We study the contribution of each loss term in our loss function: (a) pixel-wise ℓ_2 distance (MSE), (b) pixel-wise ℓ_1 distance (MAE), and (c) perceptual loss. All experiments are

Table 3.7 Quantitative results of our UPFWI with different loss function settings.

Loss			Velocity Error			Seismic Error		
pixel-ℓ_2	pixel-ℓ_1	perceptual	MAE ↓	MSE ↓	SSIM ↑	MAE ↓	MSE ↓	SSIM ↑
✓			32.61	10014.47	0.9735	0.0167	0.0023	0.9978
✓	✓		21.71	2999.55	0.9775	0.0155	0.0025	0.9977
✓	✓	✓	**16.27**	**1705.35**	**0.9866**	**0.0140**	**0.0021**	**0.9984**

Table 3.8 Quantitative results of our UPFWI evaluated on Marmousi and Salt datasets.

Method	Marmousi			Salt		
	MAE↓	MSE↓	SSIM↑	MAE↓	MSE↓	SSIM↑
InversionNet	149.67	45936.23	0.7889	25.98	8669.98	0.9764
UPFWI	221.93	125825.75	0.7920	150.34	164595.28	0.7837

Table 3.9 Quantitative results of our UPFWI with different architectures.

Network	MAE↓	MSE↓	SSIM↑
CNN	16.27	1705.35	0.9866
ViT	41.44	11029.01	0.9461
MLP-Mixer	22.32	4177.37	0.9726

conducted on FlatFault using 24,000 unlabeled data.

Figure 3.19a shows the predicted velocity maps for using three loss combinations (pixel-ℓ_2, pixel-$\ell_1\ell_2$, pixel-$\ell_1\ell_2$+perceptual) in UPFWI. The ground truth seismic data and velocity map are shown in the left column. For each loss option, we show the difference between the reconstructed and the input seismic data (on the top) and predicted velocity (on the bottom). When using pixel-wise loss in l_2 distance alone, there are some obvious artifacts in both seismic data (around 600 millisecond) and velocity map. These artifacts are mitigated by introducing additional pixel-wise loss in l_1 distance. With perceptual loss added, more details are correctly retained (e.g. seismic data from 400 millisecond to 600 millisecond, velocity boundary between layers). Figure 3.19b compares the reconstructed seismic data (in terms of residual to the ground truth) at a slice of 525 meter offset (orange dash line in Figure 3.19a). Clearly, the combination of pixel-wise and perceptual loss has the smallest residual.

Table 3.10 Quantitative results of our UPFWI tested on seismic inputs with different noise levels.

σ (10^{-4})	FlatFault				CurvedFault			
	PSNR	MAE↓	MSE↓	SSIM↑	PSNR	MAE↓	MSE↓	SSIM↑
0.5	61.60	15.68	1343.21	0.9888	61.72	23.78	3704.00	0.9751
1.0	58.70	24.84	4010.78	0.9733	58.70	24.84	4010.78	0.9733
5.0	51.58	44.33	7592.57	0.9681	51.68	46.90	10415.38	0.9441

The quantitative results are shown in Table 3.7. They are consistent with our observation in qualitative analysis (Figure 3.19a). In particular, using pixel-wise loss in ℓ_2 distance has the worst performance. The involvement of ℓ_1 distance mitigates velocity errors but is slightly worse on MSE and SSIM of seismic error. Adding perceptual loss boosts the performance in all metrics by a clear margin. This shows perceptual loss is helpful to retain waveform coherence, which is correlated to velocity boundary, and validates our proposed loss function (combining pixel-wise and perceptual loss).

More Challenging Datasets: We further evaluate our UPFWI on two more challenging tests including Marmousi and Salt (Yang and Ma, 2019) datasets and achieve solid results. For Marmousi dataset, we follow the work of Feng et al. (2021) and employ the Marmousi velocity map as the style image to construct a low-resolution dataset. Table 3.8 shows the quantitative results on both datasets. Although our UPFWI achieves good results on Salt dataset with preserved subsurface structures, it has clearly larger errors than the supervised InversionNet. This is due to two reasons: (a) Salt dataset has a small amount of training data (120 samples), which is very challenging for unsupervised methods; (b) the variability between training and testing samples is small, providing a significantly larger favor to supervised methods than the unsupervised counterparts. The visualization of results on Marmousi dataset and Salt data are shown in Appendix C.4.

Other Network Architectures: We further conducted experiments by using Vision Transformer (ViT, Dosovitskiy et al., 2020) and MLP-Mixer (Tolstikhin et al., 2021) to replace CNN as the encoder. Table 3.9 further shows the quantitative results. Solid results are obtained for both network architectures, indicating our proposed method is model-agnostic. Visualization results are

Table 3.11 Quantitative results of our UPFWI tested on seismic inputs with missing traces.

Missing Traces	FlatFault			CurvedFault		
	MAE↓	MSE↓	SSIM↑	MAE↓	MSE↓	SSIM↑
4 (5%)	21.23	1772.05	0.9868	41.33	6914.12	0.9622
7 (10%)	33.66	3504.25	0.9814	61.72	12445.90	0.9453
17 (25%)	85.21	16731.69	0.9457	121.06	36770.77	0.8853

shown in Appendix C.4.

Robustness Evaluation: We validate the robustness of our UPFWI models by two additional tests: (1) testing data contaminated by Gaussian noise and (2) testing data with missing traces. The quantitative results are shown in Table 3.10 and Table 3.11, respectively. We observe that in both experiments our model is robust to a certain level of noise and irregular acquisition. Visualization results are shown in Appendix C.4.

3.3.7 Summary

This section presented a novel physics-informed machine learning architecture tailored for full-waveform inversion, a process effectively modeled by partial differential equations. By integrating these equations with convolutional neural networks, we achieved a sophisticated form of regularization captured by the f term in Equation (1.3). Our experimental findings underscore the benefits of incorporating perceptual loss, as denoted by the R term in Equation (1.3), which has significantly enhanced the quality of seismic data reconstruction. This method not only elevates the precision of seismic interpretations but also broadens the scope of neural networks in geophysical data analysis, setting the stage for more detailed and accurate geological assessments.

3.4 Making Invisible Visible: Data-Driven Seismic Inversion with Spatio-temporally Constrained Data Augmentation

Deep learning methods have unlocked new potential for leveraging the vast amounts of data available today. Particularly noteworthy are the advancements in data-driven full-waveform inversion techniques, as seen in recent developments Zhang and Lin (2020); Wu and Lin (2019); Araya-Polo et al. (2018). These methods, however, typically rely on large-scale datasets that are often not available for full-waveform inversion tasks, thereby underscoring the critical role of data augmentation. Traditional data augmentation techniques used in computer vision, such as shifting, rotating, scaling, and cropping, may interfere with the physical integrity of seismic data. This section introduces a data augmentation method using generative models (variational autoencoder) that adheres to the $\mathcal{S}$ term in Equation (1.3), specifically designed to maintain the physical properties inherent in the data.

3.4.1 Introduction

Most of the current data-driven seismic FWI techniques are built-on end-to-end neural network structures. In order to improve the inversion accuracy and the model generalization, data-driven techniques are usually trained on a large volume of dataset, which in turn significantly increases the complexity of the networks. In Wu and Lin (2019), an encoder-decoder structure is developed to learn the regression correspondence from raw seismic data to velocity maps. In Araya-Polo et al. (2018), a fully-connected network structure is designed for the inversion. In Zhang and Lin (2020), a generative model is utilized and trained for learning the inversion operator. To give an idea of the size of the training data, in Wu and Lin (2019), a ten-layer encoder-decoder results in more than 40 million learnable model parameters. To train this deep neural network, more than 60,000 pairs of labeled simulations need to be available as reported in Wu and Lin (2019). However, obtaining such a large amount of data is challenging (or even infeasible) for some subsurface applications due to the practical obstacles in data acquisition and simulation. Particularly, seismic FWI for monitoring is notoriously known as a "small-data regime". A limited number of seismic sensors are distributed over a large area, and very few time-lapse observations can be affordably

acquired Lumley (2001). Training a data-driven seismic FWI using limited data will result in weak generalizability and misfitting. To fully unleash the power of deep learning for a better, faster, and cheaper subsurface seismic FWI approach, we develop a new data augmentation technique to bridge the gap by addressing the critical issues of generalizability and the capability of generating high quality and a large volume of training data.

Data augmentation, the process of creating new samples by manipulating the original data, addresses the data shortage at the root of the problem Shorten and Khoshgoftaar (2019). However, the most popular data augmentation methods are not appropriate for seismic imaging due to their inability to incorporate generic physics properties. Furthermore, seismic data usually yields both spatial and temporal characteristics. To address those issues, we develop data augmentation techniques to account for both spatio-temporal characteristics and the critical seismic physics to generate high-quality simulations. Our models are built on variation autoencoder (VAE) to take advantage of its direct tie to the latent representations Kingma and Welling (2014). The design of our techniques considers different representations of physics including the governing equations, the observable perception, and the physics phenomena. To validate the performance of our developed techniques, we test our models using an existing CO_2 leakage synthetic dataset, Kimberlina dataset, generated and operated by the U.S. Department of Energy (DOE) Jordan and Wagoner (2017). Our interest is to employ our data-driven FWI to image and detect small CO_2 leaks. Via various numerical tests, we demonstrate that our data augmentation techniques significantly improve the data representativeness of the training set, which in turn enhances the seismic imaging accuracy. Specifically, CO_2 plumes related to small leaks can now be much better imaged than those obtained without using augmentation.

In the following sections, we first briefly provide the related work in Section 3.4.2. We develop and discuss our data augmentation techniques in Section 3.4.4. We further provide all the numerical tests and results in Section 3.4.5. Finally, further discussion, future work, and concluding remarks will be presented.

3.4.2 Related Work

3.4.2.1 Deep Generative Models

Generative models are known as a type of unsupervised learning approaches that explicitly or implicitly model the distribution of true data so as to generate new samples with some variations Bishop (2006). Current state-of-the-art generative models are built on deep neural networks (i.e., deep generative models (DGMs)). Examples of recent DGMs include variational autoencoders (VAE) Kingma and Welling (2014) and generative adversarial networks (GAN) Goodfellow et al. (2014).

As a variation in autoencoder, VAE belongs to the DGMs that learn the data distribution explicitly. It solves a variational inference problem to maximize the marginalized data likelihood by using a generative network (decoder) and a recognition network (encoder). Once fully trained, the encoder learns a distribution over latent variable given observation, and the decoder learns a distribution over observation given latent variable. VAE and its variants have shown great potential in generating data for augmentation in different applications Luo et al. (2020); Hsu et al. (2017); Nishizaki (2017); Liu et al. (2018). In particular, Luo et al. (2020) employ a vanilla VAE to generate synthetic EEG time series for recognizing emotions. Nishizaki (2017) also employ VAE to generate waveform data for automatic speech recognition. In Liu et al. (2018), VAE is used to extracted useful features in the latent space from image data. Linear interpolation on the latent space is conducted to obtain new synthetic images. Hsu et al. (2017) develop a VAE-based data augmentation technique to address the distribution mismatch in source and target domains for improving the performance of a domain adaptation method in speech recognition. Besides VAE, other DGMs (such as GAN) have also been applied for the task of data augmentation Zhang et al. (2019); Antoniou et al. (2017); Shrivastava et al. (2017); Sixt et al. (2018). In comparison, VAE provides a natural connection to the data distribution by collapsing most dimensions in the latent representations. Another noticeable benefit of VAE-based DGMs is the relatively easier effort to train with less technical complexity for hyper-parameter selection. Provided with these aforementioned encouraging results of DGMs, a direct application of DGMs to our problems may face two major challenges. Firstly, DGMs are

in general highly data-demanding. Secondly, they are purely driven by data without considering physics.

3.4.2.2 Physics-Informed Deep Learning

Physics-informed (i.e., domain-aware) learning is a critical task to scientific machine learning (SciML) community Bas (2019). Particularly, how to incorporate physics information becomes one of the most challenging and important research topics across different scientific domains Sun et al. (2020); Gomez et al. (2020); Wang et al. (2020); Raissi et al. (2019); Zhu et al. (2019b). A thorough survey on this topic is published by Karniadakis et al. (2021); Willard et al. (2020). As pointed out in Karniadakis et al. (2021), there are three ways to make a learning algorithm physics-informed, "observation bias", "inductive bias", and "learning bias". The observation bias approaches introduce physics to the model directly through data that embody the underlying knowledge. The inductive bias approaches focus on designing neural network architectures that implicitly enforce physics knowledge associated with a given predictive task. The learning bias approaches incorporate the physics knowledge in a soft manner by appropriately penalizing the loss function of conventional neural networks. Our approach developed in this work belong to two categories of the above: observation bias and learning bias.

There are many benefits considering physics knowledge when designing a neural network models. Regardless of the application domains, one of the major benefits is to improve the robustness of the prediction model and to produce physically meaningful (and more accurate) results. Particularly, Lagaris et al. (1998) propose an artificial neural network method to solve partial differential equations (PDEs) for flow simulations. Raissi et al. (2019) develop a deep-learning-based nonlinear PDE solver. Zhu et al. (2019b) develop a numerical PDE solver using a convolutional encoder-decoder and a flow-based generative model with physics constraints. More accurate results have been shown in their work. Sun et al. (2020) develop another PDE solver using the physics-informed deep learning method. Their method leverages both the full-physics simulations and additional physics-based constraints. Wang et al. (2020) develop a spatiotemporal deep learning model to account for both data characteristics and underline physics to synthesize high-quality turbulent

imagery. All these aforementioned works provide us with great inspiration about leveraging useful physics information while developing deep learning models for our seismic imaging problems.

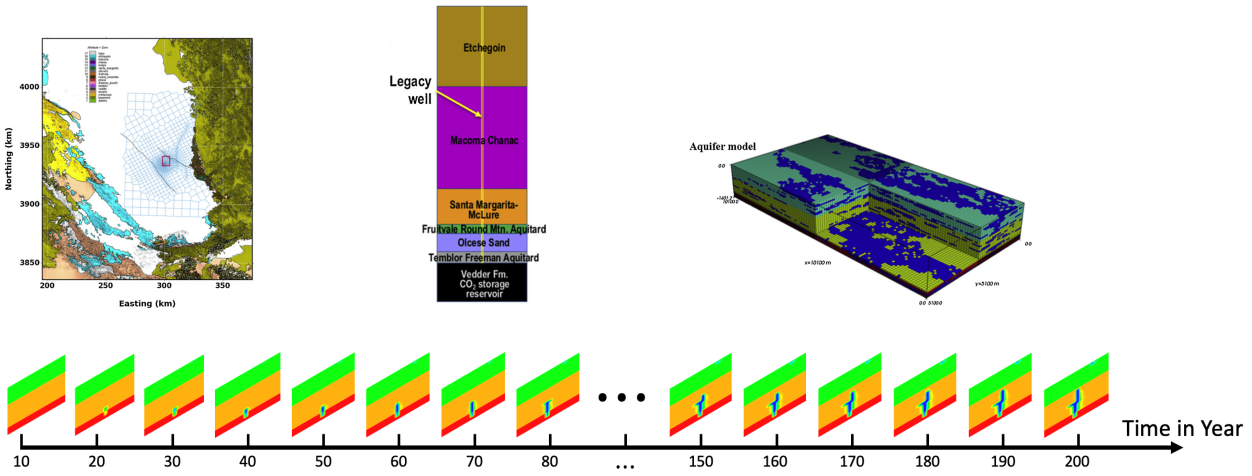


Figure 3.20 Illustration of the Kimberlina dataset and three modeling modules used to generate the simulated velocity maps. (a) CO_2 storage reservoir model, (b) wellbore leakage model, (c) multi-phase flow and reactive transport models of CO_2 migration in aquifers Buscheck et al. (2019); Yang et al. (2019), and (d) Illustration of a set of simulation with 20 velocity maps over a duration of 200 years. A CO_2 leakage will result in a decrease of the velocity value in the location where the leak happens. (© 2022 IEEE)

3.4.2.3 Data Augmentation in Seismic Exploration

In the seismic exploration community, there has been surprisingly little work addressing this dilemma of lack of data for data-driven seismic FWI. The existing approaches can be roughly categorized into two groups, those based on velocity building Liu et al. (2021); Ren et al. (2021); Wu et al. (2020a) and those based on pure machine learning approaches Feng et al. (2020); Gomez et al. (2020); Ovcharenko et al. (2019). Specifically, in Liu et al. (2021) and Ren et al. (2021), a large volume of subsurface velocity maps are generated to include different geologic structures. The geometry of those pre-generated geologic structures is assumed to follow a certain distribution. Wu et al. (2020a) design a workflow to automatically build a subsurface structure with folding and faulting features. Their method relies on the initial layer-like structure, therefore, producing unsatisfactory results when applying to different sites. In Gomez et al. (2020), an adaptive data augmentation technique is developed to augment the training by using unlabeled seismic data. Feng et al. (2020) develop a style transfer technique to generate synthetic velocity maps from natural

images. Ovcharenko et al. (2019) develop a set of subsurface structure maps using customized subsurface random model generators. Their method strongly relies on domain knowledge to generate the content images, which in turn significantly limits the variability of the training set.

3.4.3 Small CO_2 Leak Detection and Kimberlina Dataset

In geologic carbon sequestration (GCS), also known as carbon capture and storage (CCS), developing effective monitoring methods is urgently needed to detect and respond to CO_2 leakage. This is particularly important for early detection, which would provide timely warning and intervention before the potential damages to the environment (such as acidification of groundwater and killing of plant life, contamination of the atmosphere, etc) Ha-Duong and Keith (2003). On the other hand, detecting small CO_2 leaks is also technically challenging since it requires high detectability and sufficient spatial resolution of geophysical methods to capture the subtle geologic feature perturbation induced by the leaks. Considering this pressing need, our goal in this work is to assess and further improve the early CO_2 leak-detection capabilities of the seismic FWI method.

To our best knowledge, we are unaware of any available field seismic data that fits the scope of our problem of interest. Meanwhile, this lack of data is recognized by the U.S. Department of Energy (DOE), and to alleviate this problem, given the importance of this application, the DOE, through the National Risk Assessment Partnership (NRAP) project, has generated a set of high fidelity simulations, the Kimberlina dataset, with the aim of providing a standard baseline dataset to understand and assess the effectiveness of various geophysical monitoring techniques for detecting CO_2 leakage Jordan and Wagoner (2017). The Kimberlina dataset is generated from a hypothetical numerical model built on the geologic structure of a commercial-scale geologic carbon sequestration reservoir at the Kimberlina site in the southern San Joaquin Basin, 30 km northwest of Bakersfield, CA, USA. The simulation procedure consists of four modules: a CO_2 storage reservoir model (Fig. 3.20(a)), a wellbore leakage model (Fig. 3.20(b)), a multi-phase flow and reactive transport models of CO_2 migration in aquifers (Fig. 3.20(c)), and a geophysical model. In particular, the P-wave velocity maps used in this work belong to the geophysical model, which is created based on the realistic geologic-layer properties from the GCS site as shown in Fig. 3.20(b) Buscheck et al.

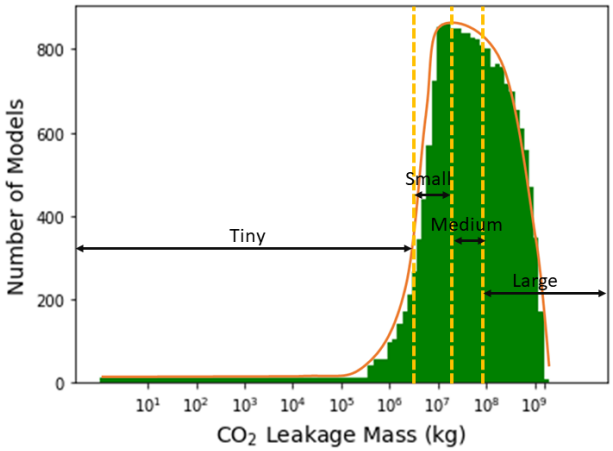


Figure 3.21 Distribution of leakage mass of Kimberlina Dataset. Each of the splittings covers 20%, 20%, 20%, and 40% of the data samples, respectively. (© 2022 IEEE)

(2019); Yang et al. (2019).

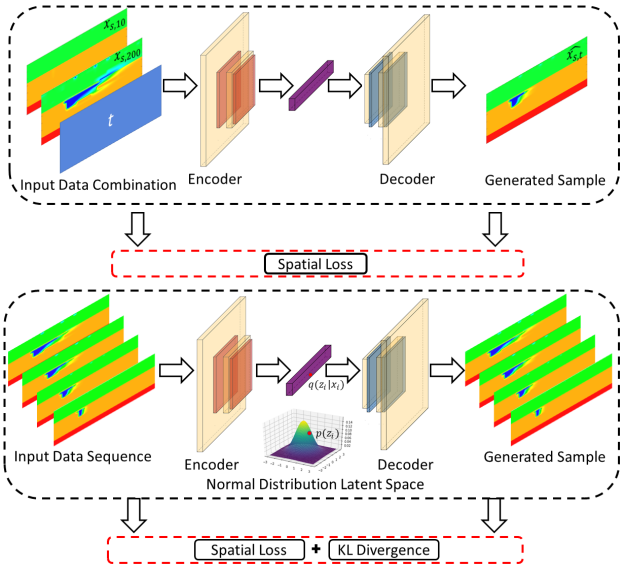


Figure 3.22 Schematic illustration of our (a) autoencoder and (b) VAE generative models. (© 2022 IEEE)

The Kimberlina dataset contains 991 CO_2 leakage scenarios, each simulated over a duration of 200 years, with 20 leakage velocity maps provided (i.e., at every 10 years) for each scenario. An illustration of one specific leakage simulation associated with the leakage mass over 200 years are shown in Fig. 3.20(d). We also provide the overall distribution of the whole dataset in Fig. 3.21. For a balanced dataset, we would expect the data label (leakage mass) should be uniformly distributed, which however is not the case for Kimberlina dataset. Particularly, the whole dataset can be split

into four parts by its leak mass as

$$\begin{cases} \textbf{\textit{Tiny}} & \text{if } \textbf{mass} < 9.10 \times 10^6 \text{ kg}, \\ \textbf{\textit{Small}} & \text{if } 9.10 \times 10^6 \text{ kg} < \textbf{mass} < 2.67 \times 10^7 \text{ kg}, \\ \textbf{\textit{Medium}} & \text{if } 2.67 \times 10^7 \text{ kg} < \textbf{mass} < 8.05 \times 10^7 \text{ kg}, \\ \textbf{\textit{Large}} & \text{if } 8.05 \times 10^7 \text{ kg} < \textbf{mass}. \end{cases} \tag{3.27}$$

Each of the splittings covers 20%, 20%, 20%, and 40% of the data samples, respectively. Although we have 20% of tiny leakage samples, these samples are distributed from 0 to 9.1×10^6 that covers nearly 70% of CO_2 leakage scenarios, as shown in Fig. 3.21. In other words, the density of tiny samples is much lower than that of the other three classes. This sparsity and in-balanced sample density create the major challenge when imaging tiny leakage samples.

The Kimberlina project focuses on the shallow CO_2 leakage. That leads to 3-layer synthetic velocity models (baseline and monitor), which reflect the shallow geologic structure from the field study. The Kimberlina model and simulations have been the basis for a variety of extensive research efforts in characterizing and detecting for CO_2 using different geophysical approaches Appriou et al. (2020); Chen and Huang (2020); Zhou et al. (2019); Buscheck et al. (2019); Yang et al. (2019). Our interest is on the early-leak detection, which requires to image those small leaks. The unbalancing of the dataset becomes a major challenge for our data-driven seismic inversion technique since it will mislead our InversionNet model towards medium or large leaks. On the other hand, due to the limitation of physical simulations, we will not be able to further generate more synthetic for the small leaks. Hence, those practical obstacles make our problem reside in a low-data regime scenario. Next, we will describe our techniques to augment the Kimberlina dataset while preserving the physics information as much as we can to improve the prediction accuracy of our InversionNet model.

3.4.4 Methodology

3.4.4.1 Physics of the Problem

Our data augmentation techniques will leverage existing physics knowledge of the problem. It is worth understanding what specific physics information are referred to in this context for the designing and training our neural networks.

1. **Governing Equations.** One of the most prominent physics knowledge in our problem is that the governing equations are used to generate original physical simulations (as shown in Fig. 3.20). Those equations describe specific physical relationships between time and spatial derivatives explicitly using temporally dynamic formulas. In order to generate physically meaningful synthesized data, it would be important to embed that physics information in the generative models.

2. **Observable Perception.** As described in Section 3.4.3, the data that we are interested in synthesizing are two-dimensional (2D), which means it yields a distribution that would be represented in certain visual perception. We expect our generative model would be able to capture the underline true data distribution, which in turn would require the synthesized 2D data would physically "look like" those in the training data.

3. **Physics Phenomena.** Any physical simulation should respect the realistic physics phenomena. As one example, in our problem of interest, the super-critical CO_2 will migrate over time, meaning that we will observe the spatial spreading of CO_2 should gradually increase over time. How to best design our generative model without violating this phenomenon would potentially help to improve the performance of our generative model.

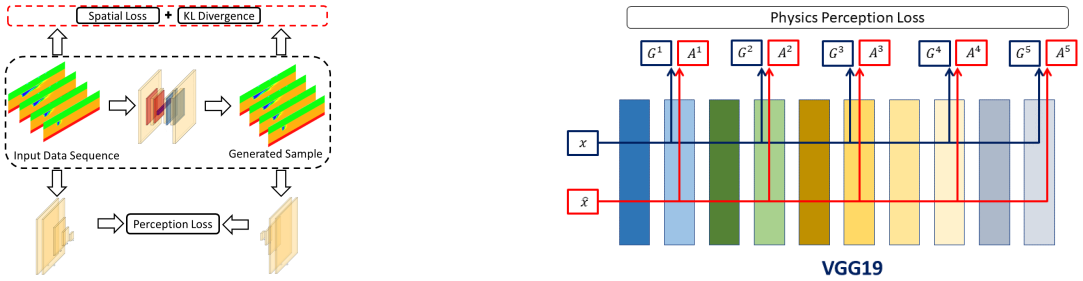
We will consider all of the above during the development of our generative models. Another point that would be important to consider is that all of the above physics information is consistent throughout all temporal duration.

3.4.4.2 Data-driven Generative Models

To compensate for the imbalance data as shown in Fig. 3.21, we would like to generate more data in the small-leak region. Luckily, the original Kimberlina dataset provides full-physics simulation in the medium- and large-leak regions. Those data are generated by the governing physics equations, which means those physics knowledge are represented by those data implicitly. Our first two generative models are built on autoencoder and VAE to leverage those existing simulations while taking into account the temporal variation.

3.4.4.3 Autoencoder

Our first model is to build a "regression" model that would provide interpolated data for those temporally missing points (as shown in Fig. 3.22(a)). The hypothesis behind this idea is that considering the consistency of the physics, we would expect that once fully trained, our generative model will capture the intrinsic dynamics of the physics from the existing simulations so that it will provide physically realistic prediction at any given time, particularly, those at the early stage of the leakage.



(a) (b)

Figure 3.23 Schematic illustration of (a) our new variational autoencoder with perception loss, and (b) the perception loss using the pre-trained VGG-19 network Simonyan and Zisserman (2014). (© 2022 IEEE)

Technically, our model will be based on an autoencoder structure, which consists of a convolutional encoder $\mathcal{F}_\theta$, with a set of trainable parameters θ, and a convolutional decoder $\mathcal{G}_\phi$, with a set of trainable parameters ϕ. To incorporate the spatial information, we set two input channels of our encoder as the first and the last velocity maps from one simulation. To incorporate the temporal information, we further create a temporal matrix by replicating the single time value over

all matrix entries. The temporal matrix will then be used as one of the three input channels of the autoencoder together with two other two. When training the autoencoder, all three input channels will be convolved together, leading to the incorporation of both spatial and temporal information. Once fully trained, our encoder will learn to reduce the dimensionality of this mixture of inputs to a latent variable which is a high-level latent representation containing both spatio-temporal information. Our decoder will estimate the target velocity map using the latent variable, which can be considered as nonlinear high-dimensional regression.

The structure of our generative model is shown as Fig. 3.22(a), and mathematically our autoencoder can be represented as

$$\begin{aligned} &\text{Encoder}: \ z = \mathcal{F}_\theta(x_{s,10},\, x_{s,200},\, t), \\ &\text{Decoder}: \ \hat{x}_{s,t} = \mathcal{G}_\phi(z), \end{aligned} \tag{3.28}$$

where $x_{s,10}$, $x_{s,200}$ and t are the inputs of the encoder. t is the time of the velocity map that needs to be predicted and it is created as a temporal matrix by replicating the single time value over all matrix entries. $x_{s,10}$ and $x_{s,200}$ are the first data and the last data from the same simulation s, where 10 and 200 are the time index of the data. z is the latent variable output by the encoder $\mathcal{F}$, and the decoder $\mathcal{G}$ produces $\hat{x}_{s,t}$, the estimated velocity map of simulation s at time t. We use Mean Squared Error (MSE) as our optimality criterion to compute the reconstruction loss between the ground truth and the generated velocity maps and to update trainable parameters through backpropagation

$$\begin{aligned} \mathcal{L}(\theta, \phi) = \mathcal{L}_{recon} &= \frac{1}{|S||T|} \sum_{s \in S, t \in T} (x_{s,t} - \hat{x}_{s,t})^2, \\ &= \frac{1}{|S||T|} \sum_{s \in S, t \in T} (x_{s,t} - \mathcal{G}_\phi(\mathcal{F}_\theta(x_{s,10}, x_{s,200}, t)))^2. \end{aligned} \tag{3.29}$$

This model generates synthetic samples in the data domain. As discussed in Oring et al. (2020), generating samples in the latent space might increase the variability within the data distribution. We, therefore, study VAE and its capability in generating samples.

3.4.4.4 Variational Autoencoder

The variational autoencoder (VAE) is a probabilistic generative model to create a latent representation of the input data. That would allow us to generate new samples with high diversity by

manipulating the latent representations. Unlike the autoencoder model, which incorporates temporal information as part of the input (Fig. 3.22(a)), our VAE generative model produces new temporal interpolation separately in two steps. In the first step, we train the VAE by taking only velocity maps as input without explicit temporal information. Once fully trained, the VAE will be able to generate latent variables representing the velocity maps. In the second step, we provide a linear interpolation scheme on the normal distributed latent space to produce new latent variables for further synthesizing new velocity samples. The idea behind the VAE generative model is somewhat similar to that of the autoencoder. We expect the physics knowledge, i.e., the governing physics relationship can be captured by training the VAE using simulations. The consistency of the physics information will be leveraged when generating new samples at different times.

We provide the illustration of our VAE generative model in Fig. 3.22(b). The encoder, $\mathcal{F}$, and decoder, $\mathcal{G}$, structures of VAE are similar to those of the autoencoder. However, the encoder in VAE is to learn the posterior distribution $q_\theta(z|x)$, which is the distribution parameter of latent variable z given input x. The decoder in VAE is to learn the conditional distribution $p_\phi(x|z)$, which is the distribution of reconstructed data given latent distribution. There is a prior distribution $p(z)$ over the latent space, which we set as a standard normal distribution. The output of encoder $q_\theta(z|x)$ has two parts of mean and log-variance of the posterior distribution. One of the known problems associated with VAE is that its gradients cannot flow through the bottleneck of mean and log-variance. So, we perform a re-parameterize trick to make the gradient able to flow through the bottleneck Kingma and Welling (2014),

$$z = \mu + \sigma \odot \epsilon, \tag{3.30}$$

where $z \in \mathbb{R}^{64}$ is the latent sample, $\mu \in \mathbb{R}^{64}$ and $\sigma \in \mathbb{R}^{64}$ are the mean and log-variance of the posterior distribution $q_\theta(z|x)$, and $\epsilon \in \mathbb{R}^{64}$ is a random variable sampled from normal distribution

and independent from μ and σ. We employ the standard VAE loss function as below

$$\begin{aligned}\mathcal{L}(\theta,\phi) &= \mathcal{L}_{recon} + \mathcal{L}_{kld},\\ &= \sum_i \mathbb{E}_{q_\theta(z_i|x_i)}\left[\log \frac{p_\phi(x_i,z_i)}{q_\theta(z_i|x_i)}\right],\\ &= \sum_i \mathbb{E}_{q_\theta(z_i|x_i)}(\log p_\phi(x_i|z_i)\\ &+ \log p(z_i) - \log q_\theta(z_i|x_i)),\\ &= \sum_i (x_i-\hat{x}_i)^2 + \sum_i D_{KL}(q_\theta(z_i|x_i)\|p(z_i)),\end{aligned} \tag{3.31}$$

where $\mathcal{L}_{kld} = \sum_i D_{KL}(q_\theta(z|x_i)\|p(z)) = \sum_i \mathbb{E}_{q_\theta(z_i|x_i)}(\log p(z_i) - \log q_\theta(z_i|x_i))$ is to measure the KL-divergence between the posterior distribution $q_\theta(z|x)$ and the prior distribution $p(z)$. $\mathcal{L}_{recon} = \sum_i \mathbb{E}_{q_\theta(z|x_i)}(\log p_\phi(x_i|z_i) = \sum_i (x_i - \hat{x}_i)^2$ is the reconstruction loss between ground truth velocity maps, x_i and generated velocity maps, $\hat{x}_i$.

When generating new velocity maps to augment our dataset, a directly random sampling on the prior distribution may lead to velocity maps associated with different leaks, whereas our interest is to obtain more small-leakage velocity maps. So we come up with an interpolation strategy. Particularly, we obtain the latent variables of two adjacent velocity maps from the same simulation, namely, z_1 and z_2 through the encoder, $\mathcal{F}$. Multiple new latent variables can be interpolated between these z_1 and z_2 before passing them through the decoder, $\mathcal{G}$, to further generate additional velocity maps Berthelot et al. (2018). The procedure can be posed as follows

$$\begin{aligned}\hat{x}_\alpha &= \mathcal{G}_\phi(\alpha z_1 + (1-\alpha)z_2),\\ &= \mathcal{G}_\phi(\alpha \mathcal{F}_\theta(x_{s,10}) + (1-\alpha)\mathcal{F}_\theta(x_{s,200})),\end{aligned} \tag{3.32}$$

where $\alpha \in [0,1]$ is the coefficient of the interpolation. Different from the autoencoder model, the temporal information is not used as the input in this VAE generative model.

It is worth mentioning that for either the autoencoder or VAE as shown in Fig. 3.22, we expect that the governing physics will be able to be learned through training the models using full-physics simulations. However, other physics knowledge (such as observable perception or physics phenomena) will not be able to be captured by the generative models. Hence, we come

up with two different strategies to further constrain our generative models with additional physics knowledge.

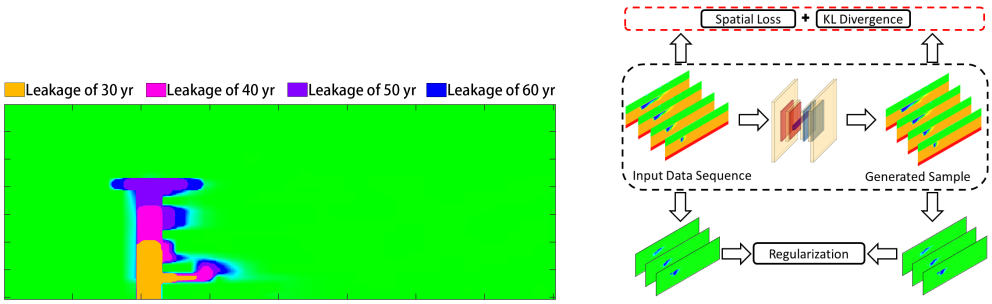


Figure 3.24 Schematic illustration of (a) the spatio-temporal dynamics of velocity maps at four consecutive times, and (b) our new variational autoencoder with regularization. In (a), the CO_2 plume is observed migrating towards a specific spatial direction over the monitoring period. (© 2022 IEEE)

3.4.4.5 Spatio-temporal Constrained Generative Models

3.4.4.6 Variational Autoencoder with Perception Loss

Perception of the generated velocity map is an important criterion to evaluate the quality of the synthesized image data. For both loss functions in Eqs. (3.29) and (3.31), we employ $\mathcal{L}_2$ norm to quantify the error. However, as pointed out by Zhang et al. (2018), classic per-pixel measures would be insufficient for assessing structured data such as images. It is obvious that the perception of the generated velocity map and that of the true velocity map should be consistent throughout the whole dataset. Inspired by recent work on style transfer Gatys et al. (2015), we can quantify perception using features extracted from pre-trained VGG-19 classification network Simonyan and Zisserman (2014), and further calculate the perception error between the true and the generated velocity maps. Thus, reducing the perception error can make our generated velocity maps more physically realistic.

Our new VAE generative model is shown in Fig. 3.23(a), where one additional loss function (i.e., "Perception Loss") is added on top of the spatial loss and KL divergence. In Fig. 3.23(b), we illustrate how we extract spatial features and use them to calculate the perception loss. Particularly, we generate representation using the Gram matrix, $G^l \in \mathbb{R}^{N_l \times N_l}$, on feature maps from several

intermediate layers of VGG-19 net

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}, \tag{3.33}$$

where G^l_{ij} is the inner product between the vectorized feature i, j at layer l and F^l_{ik} is the position k of the vectorized feature map of the i^{th} filter at layer l of VGG-19 net. There are N_l feature maps at layer l of VGG-19 net with the size of M_l which is the height times the width of the feature map. With the Gram matrix of ground truth velocity map, x (denoted as "G^l") and that of the generated velocity map, $\hat{x}_\alpha$ (denoted as "A^l") obtained at layer l, we will have the perception loss function as

$$L_{phys} = \sum_l \lambda_l \sum_{ij} (G^l_{ij} - A^l_{ij})^2, \tag{3.34}$$

where $\lambda_l = \frac{1}{4N_l^2 M_l^2}$ is the coefficient of the perception loss at layer l. Hence, our new VAE with perception loss function becomes

$$\begin{aligned} \mathcal{L}(\theta, \phi) &= \mathcal{L}_{recon} + \mathcal{L}_{kld} + \mathcal{L}_{phys}, \\ &= \sum_i (x_i - \hat{x}_i)^2 + D_{KL}(q_\theta(z_i|x_i) \| p(z_i)) \\ &+ \sum_l \lambda_l \sum_{ij} (G^l_{ij} - A^l_{ij})^2. \end{aligned} \tag{3.35}$$

An important hyper-parameter that needs to be carefully tuned is the selection of layers from VGG-19 net that will be used for calculating the perception loss. We will provide more details later in the numerical test.

3.4.4.7 Variational Autoencoder with Regularization

As we discussed previously, prominent phenomena also play a critical role in designing our generative model. For the CO_2 storage problem, it has been well understood that starting from the injection well, CO_2 will enter the formation at high flow rates and migrate relatively and vigorously into the most permeable regions under strong pressure gradients while displacing native fluids (e.g., brine) Birkholzer et al. (2015). Our full-physics simulations as shown in Fig. 3.20 accurately illustrate this process, which results in a prominent spatially and temporally varying pattern of the CO_2 plume (shown in Fig. 3.24(a)). We expect our generative model would respect (at least not violate) this particular dynamics. To enforce this constraint, our idea is to design a regularization

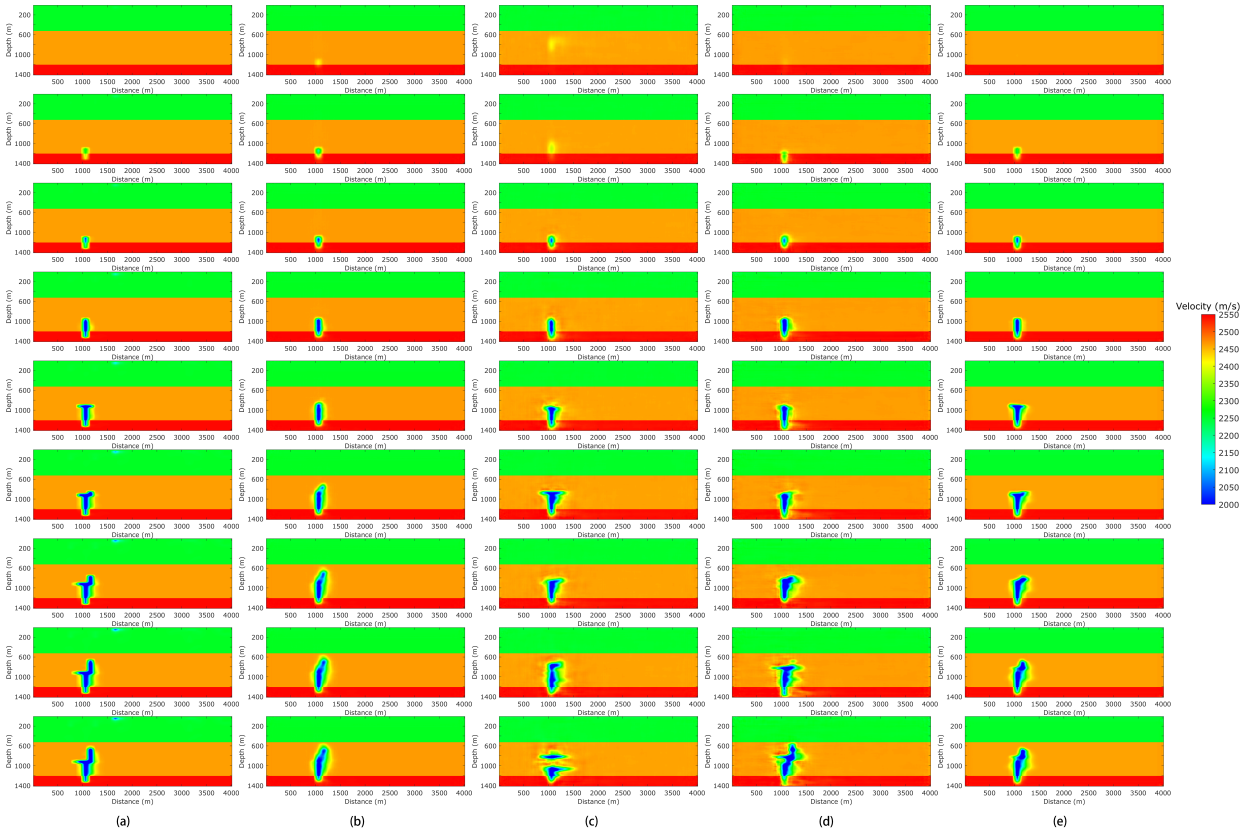


Figure 3.25 (a) Ground truth velocity maps, and generated velocity maps using (b) autoencoder, (C) VAE, (d) VAE with perception loss, and (e) VAE with regularization. (© 2022 IEEE)

term informed by the leakage process with the hope to produce new samples being consistent with the underline spatio-temporal dynamics.

As shown in Fig. 3.24(a), we observe that the CO_2 plume would span towards a certain spatial direction over the time during the migration, which indicates a clear spatio-temporal dynamical pattern. We, therefore, impose regularization on top of the difference between velocity maps at two consecutive times and ensure the dynamics of the ground truth would be preserved to its best when generating new synthetics. To achieve this, we employ a $\mathcal{L}_1$-norm based regularization given by

$$\mathcal{L}_{reg} = \left\| (x_{s,t1} - x_{s,t2}) - (\hat{x}_{s,t1} - \hat{x}_{s,t2}) \right\|_1 , \tag{3.36}$$

where x_{t1} and x_{t2} are ground truth velocity maps at two consecutive times, respectively. Accordingly, $\hat{x}_{t1}$ and $\hat{x}_{t2}$ are generated velocity maps at the same times, respectively. The times, $t1$ and $t2$, are adjacent to each other with $t1 > t2$. The reason that we use $\mathcal{L}_1$ norm instead of $\mathcal{L}_2$ norm is that the value of the subtraction of the differences of two consecutive velocity maps can sometimes be

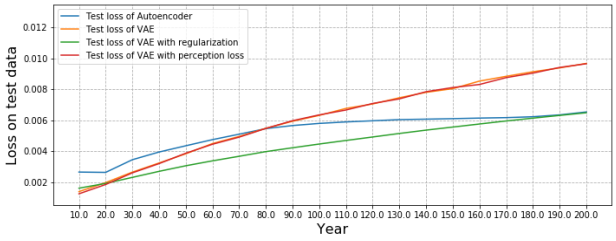


Figure 3.26 Reconstruction loss on test dataset using different models w.r.t. each year in the dataset. (© 2022 IEEE)

very close to zero, which would make the $\mathcal{L}_2$-norm value too small and may lead to the gradient vanishing issue. In Fig. 3.24(b), we provide the network structure of our generative model using a new VAE loss function with regularization

$$\begin{aligned}\mathcal{L}(\theta,\phi) &= \mathcal{L}_{recon} + \mathcal{L}_{kld} + \gamma\,\mathcal{L}_{reg},\\ &= \sum_i (x_i - \hat{x}_i)^2 + D_{KL}(q_\theta(z_i|x_i)\|p_\phi(z_i))\\ &+ \gamma\,\left\|(x_{s,t1} - x_{s,t2}) - (\hat{x}_{s,t1} - \hat{x}_{s,t2})\right\|_1,\end{aligned} \tag{3.37}$$

where the first and the second terms are the reconstruction loss and KL-divergence of the VAE, respectively. The third term is regularization. γ is the regularization parameter. The regularization parameter in Eq. (3.37) is important to the accuracy of data generation. We will explore its impact and how we select it in our numerical test. Another technical details may be worthwhile mentioning is the selection of the norms (i.e., $\mathcal{L}_2$ norm versus $\mathcal{L}_1$ norm) in our loss functions and the regularization terms. The $\mathcal{L}_2$ loss (i.e., Mean Squared Error (MSE)) and $\mathcal{L}_1$ loss (i.e., Mean Absolute Error (MAE)) are two of the most popularly used functions. Since MAE is minimized by conditional median which may lead to bias during optimization, we therefore choose MSE, which is minimized by conditional mean. However, we select the $\mathcal{L}_1$ regularization term to promote the sparsity of the coefficients when constraining the differences between two (or thee) adjacent velocity maps.

3.4.5 Experiments

With all 4 different generative models being designed in the previous section, we will validate their performances in a couple of scenarios including a general assessment of the synthesized

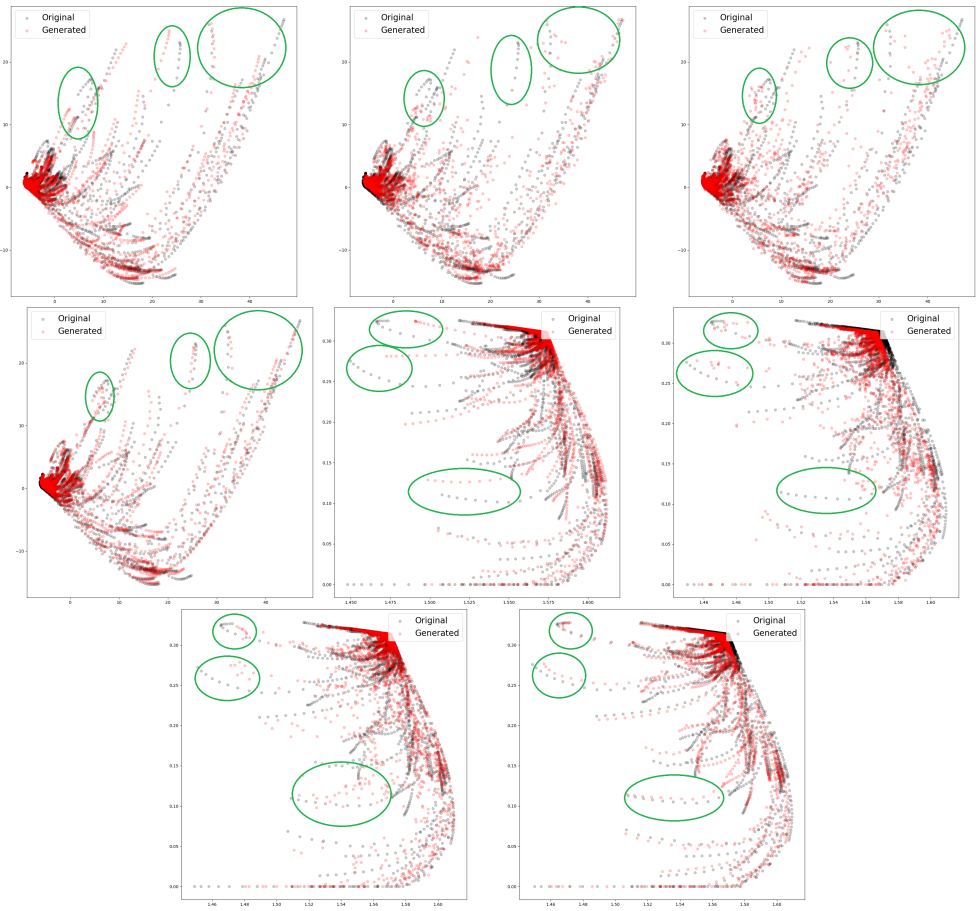


Figure 3.27 Clustering results of the generated versus true samples. PCA-based clustering for (a) autoencoder, (b) VAE, (c) VAE with perception loss, and (d) VAE with regularization; NMF-based clustering for (e) autoencoder, (f) VAE, (g) VAE with perception loss, and (h) VAE with regularization. (© 2022 IEEE)

data (Test 1) and the performance in imaging small CO_2 leakage (Test 2). We will also provide numerical tests to illustrate what would be a reasonable augmented data size (Test 3) and how we pick some of the critical hyper-parameters (Test 4).

3.4.5.1 Experiment Setup

We use 800 leakage scenarios in our dataset (16,000 samples in total) as a training dataset and the rest of the simulations (3,763 samples in total) as a test dataset. For training of our proposed data augmentation models, we use a batch size of 32, and train models for 100 epochs using ADAM optimizer with a learning rate of 0.0001. The initialization of model weights is based on He initialization He et al. (2015). For the training of InversionNet, we use a batch size of 24, and train InversionNet for 80 epochs using ADAM optimizer with an initial learning rate of 0.01, and with a

Table 3.12 Computational costs of different generative models. Row 1 is the size of the each model. Row 2 is the memory cost. Row 3 is the time per epoch in training each model. Row 4 is the total time in training each model. Row 5 is the time in generating a single sample. Row 6 is time in generating 3,000 velocity maps set in parallel. (© 2022 IEEE)

	Autoencoder	VAE	VAE_percep	VAE_reg
Parameter #	3,382,290	6,432,818	6,432,818	6,432,818
GPU Memory Cost	10.3GB	14.6GB	14.6GB	14.6GB
Time (Training/epoch)	48s	36s	258s	48s
Time (Training/total)	80m	60m	430m	80m
Time (Generation/sample)	1.36s	4.38s	4.38s	4.38s
Time (Generation/set)	3.13m	3.41m	3.41m	3.41m

weight decay coefficient of 0.0001.

Table 3.13 Computational costs of training InversionNet without and with augmented dataset with 3,000 more velocity maps. (© 2022 IEEE)

	InversionNet without augment	InversionNet with augment
Training Time (total)	1h23m	1h31m

3.4.5.2 Test 1: Velocity Map Generation

We provide the synthesized velocity maps generated using our four different generative models in Fig. 3.25. For VAE, we use the first column, the ground truth velocity maps as the input velocity maps. For autoencoder, we use two velocity maps of 10-year and 200-year as the input velocity maps. Overall, we observe that all four generative models produce reasonable results. VAE (Fig. 3.25(c)) yields images with the highest variability among all four models. This is particularly true when comparing to autoencoder results (Fig. 3.25(b)). However, due to the variability, some unrealistic features can be also observed in the VAE results. An example would be the one at the last row, where the whole leakage plume is unphysical split into two. The VAE with perception loss (Fig. 3.25(d)) produces better results in preserving the perception of the velocity map. The unphysical data in the later stage is improved. Meanwhile, the images at the early stage also match better to the true images comparing to the VAE results. However, we also notice some artifacts being generated in the VAE with the perception loss model. The best results produced by all four models is the VAE with

regularization (Fig. 3.25(e)). It produces not only cleaner images but also highly accurate images in the early stage. To further quantitatively compare different generative models, we run our models on the test dataset and calculate the test loss w.r.t. each year of the data. The result is shown in Fig. 3.26. Consistent with what we observe in Fig. 3.25, VAE with regularization (in red), yields the best performance among all four models. Autoencoder yields a lower reconstruction loss for velocity maps after 80 years compared to those of vanilla VAE, VAE_percep, and VAE_reg models. However, this only implies that autoencoder produces a better performance on reconstructing velocity maps after 80 years, and it does not mean that autoencoder can generate more realistic and diversified velocity maps from the underlying distribution, which however is essential for InversionNet to capture the underlying data distribution and help with the generalization ability.

Another means to justify the quality of our synthesized data is to visualize the distribution of the generated data versus that of the true data. To achieve this, we employ two commonly used methods: Principal Components Analysis (PCA) Smith (2002) and Non-Negative Matrix Factorization (NMF) Lee and Seung (1999). The visualization results are provided in Fig. 3.27. Regardless of the clustering approaches, the VAE with regularization (Figs. 3.27(d) and (h)) produces the distribution matching most closely to that of the ground truth. VAE and VAE with perception loss models yield comparable results. The autoencoder-based model performs the worst out of all four models.

Computation cost is also an important factor in evaluating the performance of a generative model. To that perspective, we provide in Table 3.12 more details of the cost by comparing the model complexity (number of parameters), memory consumption, training time per epoch, total training time, sample-generation time, and total generation time required. Particularly, we compare all four models listed in our manuscript including autoencoder, VAE, VAE_percep and VAE_reg. We observe that autoencoder yields the smallest number of network parameters among all four models and all the other three VAE-based models yield comparable network complexity and memory requirement. As for the training time cost, VAE_percep is the most time-consuming whereas the remaining three models are comparable in training cost. The excessive time of training the

VAE_percep model is due to accessing multiple layers of VGG-19 for extracting relevant spatial features that would be needed in computing the Gram matrix and perception loss. Once the models are fully trained, generating the augmented dataset (3,000 samples) only spends around 4 minutes, which is not very time-consuming. With new samples being generated, we provide the training time comparison of InversionNet with/without augmented dataset in Table 3.13. We notice that training InversionNet with augmentation dataset will only increase 8 minutes in training.

To summarize, in this test we demonstrate the capability of our four generative models in synthesizing high-quality velocity maps, which would provide additional data to train data-driven seismic imaging methods. Particularly, our VAE with regularization model generates the synthesized data with the most appealing visual quality and matches best to the true data distribution.

3.4.5.3 Test 2: Performance on Edge Cases

The main purpose of this test is to evaluate the performance of our developed data augmentation techniques in improving the data-driven seismic imaging method in characterizing small leakage. We firstly generate 4 groups of synthesized data using our proposed generative models. There are 3,000 velocity maps in each group. As a baseline, we will train InversionNet Wu and Lin (2019) with an initial training dataset, which consists of 800 simulations, amounting to a total of around 15,000 samples. We further design two different test categories of one with all sizes of leakage (named as "General Leakage") and the other one with only tiny and small leakage (named as "Small Leakage"). For the definitions of different leakage (tiny, small, medium, and large), please refer to Eq. (3.27) and Fig. 3.21. The test samples of different leakage sample in General Leakage and Small Leakage are provided in Table 3.14.

Table 3.14 Two different test sets for evaluating the performance of our generative models. For the definitions of different leakage (tiny, small, medium, and large), please refer to Eq. (3.27) and Fig. 3.21. (© 2022 IEEE)

	Tiny	Small	Medium	Large
General leakage	717	770	675	1494
Small leakage	153	38	0	0

Table 3.15 Test loss of InversionNet on General Leakage and Small Leakage tests without augmentation (Col 2), and with augmentation data generated using autoencoder (Col 3), VAE (Col 4), VAE with perception loss (Col 5) and VAE with regularization (Col 6). The results using VAE with regularization are the best comparing to all others. (© 2022 IEEE)

	Baseline	AE	VAE	VAE_percep	VAE_reg
General leakage	0.001294	0.001229	0.001522	0.001331	**0.001093**
Small leakage	0.000780	0.000813	0.001157	0.000924	**0.000646**

For all tests, we train InversionNet for 80 epochs to assure its convergence. We report in Table 3.15 the test loss of InversionNet on both testing categories with/without augmented training data sets. Particularly, our VAE with regularization yields the smallest loss value for both "General Leakage" and "Small Leakage". On the other hand, VAE model (Col 4) produces the worst results among all four methods. We suspect that high variability and some unphysical synthesized samples "confuses" the InversionNet in learning the data distribution leading to degraded performance. This can be confirmed by noticing that with some additional constraints being imposed to the VAE model, an immediate performance improvement can be observed in the results using VAE with perception loss (Col 5) and VAE with regularization (Col 6). To better understand the error distribution using different generative models, we also provide in Fig. 3.28 a box-plot on the small leakage test. Out of the three methods, it is clear that VAE_reg yields the best performance with the smallest median value and interquartile ranges. Comparing VAE_percep and AE models, although they both produce similar median values, the VAE_percep is much less dispersed than the AE model. However, we notice more outliers are existing in the VAE_percep box-plot than those in the other two box-plots, which explains degradation of the overall test loss of VAE_percep as reported in Table 3.15.

To better visualize the performance in imaging small leakage, we provide the reconstructed imaging results of InversionNet in Figs. 3.29(b) to (d). The differences of the reconstructions (by subtraction results from ground truth) are further provided in Figs. 3.30(a) to (c). The ground truth of the testing samples is shown in Fig. 3.29(a). To quantify the errors of the imaging results, we use two metrics, the mean-absolute errors (MAE) and structural similarity indexes (SSIM) Wang et al. (2004). The leakage mass and errors of the imaging results are provided in Table 3.16. InversionNet

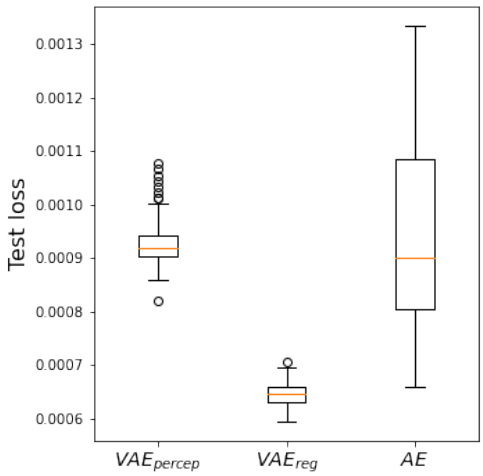


Figure 3.28 Illustration of the error distribution (median, range, and outliers) in a box-plot on the small leakage test. (© 2022 IEEE)

trained without any augmentation data yields the worst imaging results. The CO_2 plume is either very hard to visualize or distorted severely, which results in the highest MAE and the lowest SSIM value comparing to others. InversionNet trained on augmented data sets produce much-improved imaging results. Particularly, the one using VAE with regularization yields the best imaging results with the smallest MAE and highest SSIM values.

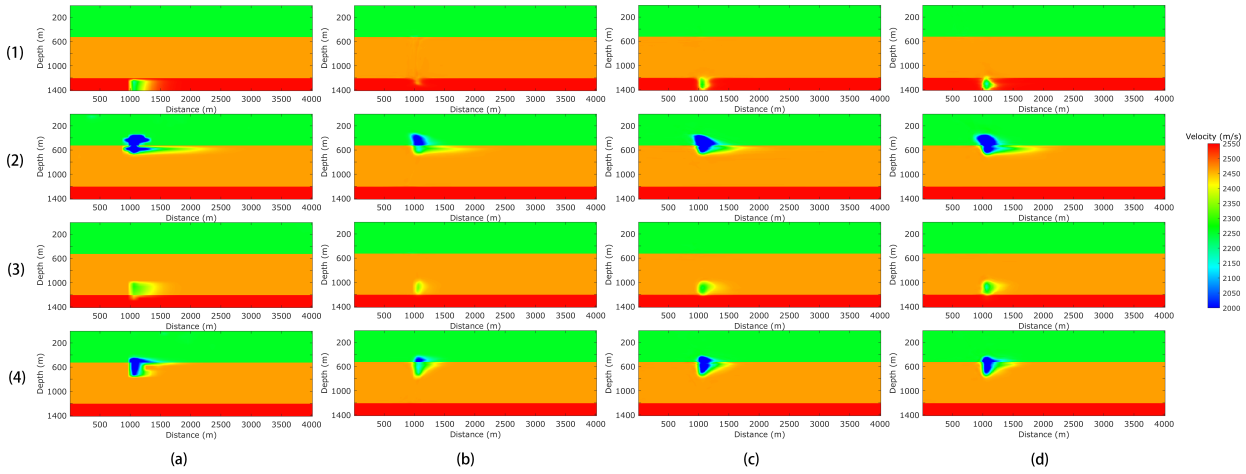


Figure 3.29 Four groups of InversionNet imaging results (b, c, d) on small leakage test data. (a) Ground truth, InversionNet imaging results (b) without augmentation, with augmented data set generated using (c) VAE with perception loss, and (d) VAE with regularization. (© 2022 IEEE)

Besides visualization of the resulting images, quantifying the spatial resolution will provide a different perspective to evaluate the quality of the results. Here, we employ the commonly used

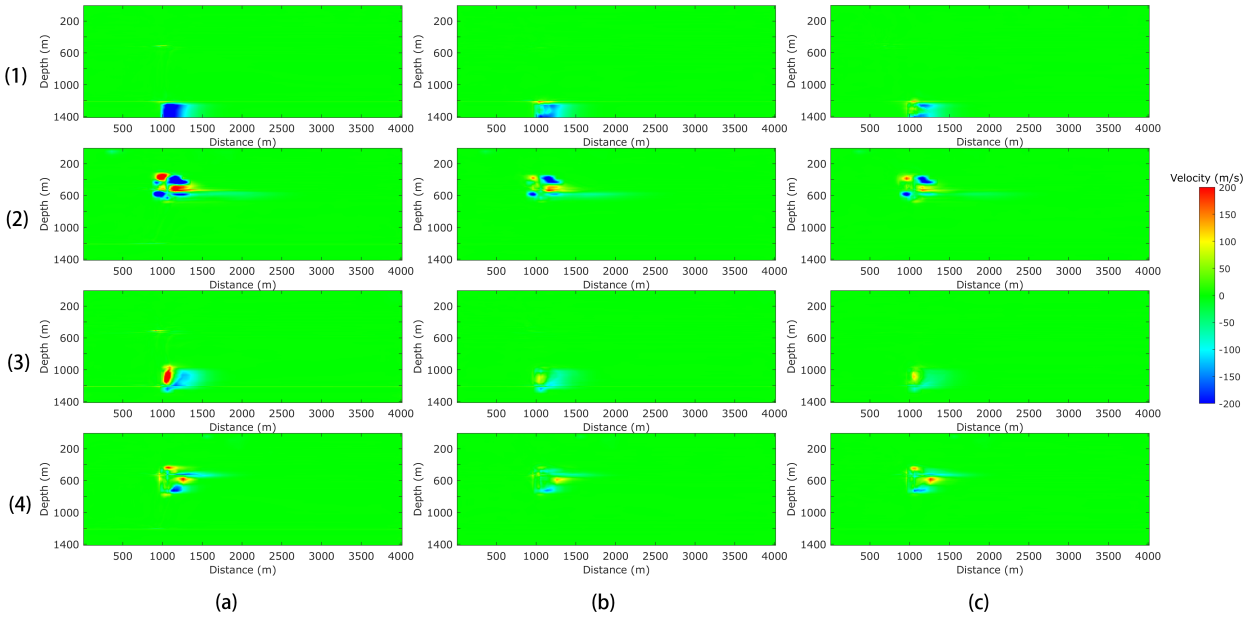


Figure 3.30 Four groups of differences of InversionNet imaging results to ground truth (e, f, g) on small leakage test data. (a) Difference of InversionNet imaging results without augmentation (Fig. 3.29 (b)) to ground truth (Fig. 3.29 (a)), (b) difference of results with augmented data set generated using VAE with perception loss (Fig. 3.29 (c)) to ground truth, (c) difference of results with augmented data set generated using VAE with regularization (Fig. 3.29 (d)) to ground truth. (© 2022 IEEE)

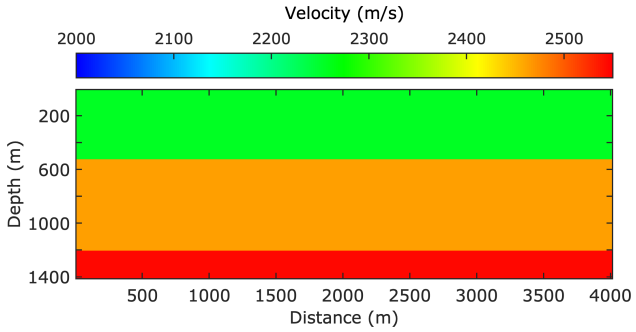


Figure 3.31 Illustration of the baseline velocity map. (© 2022 IEEE)

wavenumber analysis on our imaging results to help with justifying the quality of the resulting image resolution Our focus is on the velocity perturbation induced by the CO_2 leaks as shown in Fig. 3.29. The perturbation can be obtained by subtracting the time-lapsed images from the baseline image (shown in Fig. 3.31), where the baseline image refers to the one without any leaks. Once the velocity perturbation is obtained after the subtraction, we employ the spatial Fourier transform to obtain the wavenumber (i.e., the Kz spectrum), and we provide the plots (in Figs. 3.32) using all four imaging results shown in Fig. 3.29. We observe that the Kz spectra of our results (in blue) are much closer to

Table 3.16 Leakage mass (Col 2) of CO_2 of four groups of velocity maps showed in Fig. 3.29. MAE and SSIM errors of InversionNet imaging results using baseline without augmentation (Col 4), VAE with perception loss (Col 5) and VAE with regularization (Col 6). (© 2022 IEEE)

Group	Leakage Mass	Metric	Baseline	VAE_percep	VAE_reg
1	4.08×10^6 kg	MAE	0.00277	0.00237	**0.000889**
		SSIM	0.9907	0.9929	**0.9936**
2	4.29×10^3 kg	MAE	0.00437	0.00346	**0.00290**
		SSIM	0.9833	0.9842	**0.9871**
3	8.89×10^5 kg	MAE	0.00102	**0.000742**	0.000766
		SSIM	0.9985	**0.9987**	0.9986
4	8.05×10^5 kg	MAE	0.00255	**0.00138**	0.00168
		SSIM	0.9926	0.9928	**0.9949**

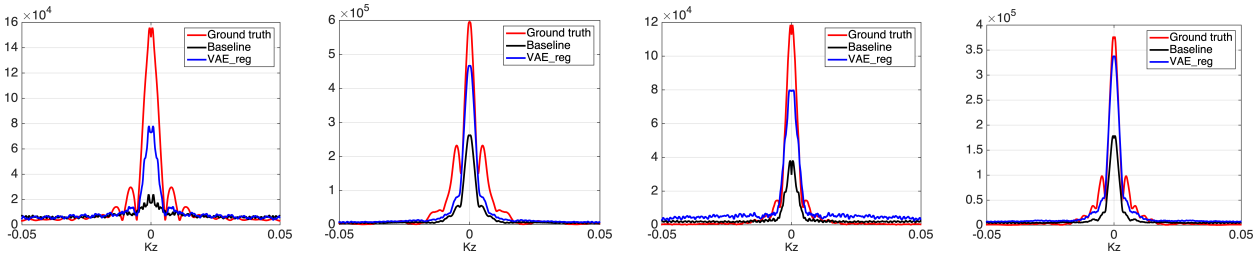


Figure 3.32 Resolution analysis of the four imaging results as shown in Fig. 3.29. The Kz spectra of our results (in blue) are much closer to those of the ground truth (in red) by comparing the baseline method (in black). That indicates that our imaging method yields higher spatial resolution than the baseline method. (© 2022 IEEE)

those of the ground truth (in red) by comparing the baseline method (in black) for all imaging results. That indicates that our imaging method yields higher spatial resolution than the baseline method.

In this test, we study the performance of InversionNet using augmented data sets generated by our models on various leakage scenarios. Due to the lack of consideration of underlying physics knowledge, both autoencoder and vanilla VAE models do not help to improve the overall performance of InversionNet. On the other hand, our proposed model, VAE_reg, is capable of generating physically realistic synthetic samples, which in turn will further improve in imaging all leakage cases. In particular, the imaging resolution on tiny leakage is significantly enhanced with the CO_2 plume much better resolved. It is worth mentioning that although the other proposed model, VAE_percep, is capable of generating comparable synthetic samples to VAE_percep, it may not

lead to an improved overall imaging quality in this dataset. We still include this new model and its results since it may perform better in a different application and dataset.

Through the numerical tests and comparison, we conclude that the performance of InversionNet can be much improved in imaging all leakage cases. In particular, the imaging resolution on tiny leakage is significantly enhanced with the CO_2 plume much better resolved.

3.4.5.4 Test 3: Determination of Augmented Data Size

The size of the augmentation data is critical to the resulting performance. Without sufficient augmented data, the imaging results of InversionNet will be sub-optimal. In contrast, augmenting the original training set with too much data may lead to "augmentation leak", meaning that synthesized data dominate the training set and distorts the true data distribution Zhao et al. (2020). In this test, we aim to find out the best range of the amount of augmentation data for optimizing the performance of InversionNet.

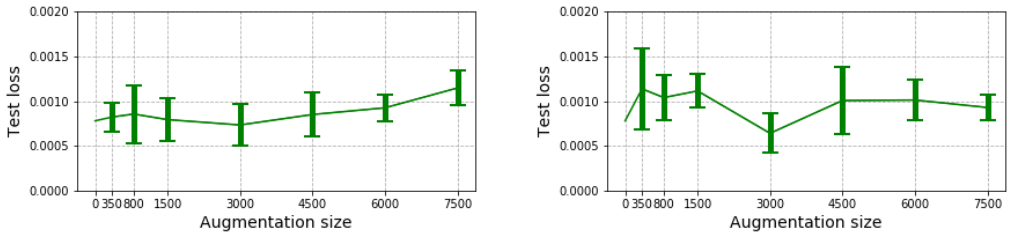


Figure 3.33 Mean and standard deviation of test loss for different augmentation sizes. Test loss of InversionNet using augmented data set generated with (a) VAE with perception loss, (b) VAE with regularization. For all the tests, the same Small Leak data set (see Table 3.14) are utilized as the test data. (© 2022 IEEE)

Through Tests 1 and 2, we learn that VAE with perception loss and VAE with regularization usually yield better results. Hence, we focus on the impact of the varying augmentation data size over those two models. We vary the size of the augmented data set among 350, 800, 1500, 3000, 4500, 6000, and 7500. For each case, we generate 5 different groups of augmentation data using our generative models. With all those groups of synthesized data being available, we train InversionNet with augmented data and test on the same Small Leak data as used in Test 2 (see Table 3.14). We report the corresponding loss values in terms of mean and standard deviation in Fig. 3.33. We observe a general pattern "decrease first $\longrightarrow$ bottom $\longrightarrow$ increase later" from both results in

Fig. 3.33. This type of pattern has recently been discovered and analyzed in other data augmentation literature Zhao et al. (2020); Karras et al. (2020). It is mainly caused by an augmentation leak, which should be used as an indication to decide a reasonable augmentation data size. Interestingly, for both VAE with physics perception loss and VAE with regularization, the smallest test loss values are achieved when 3,000 synthetic velocity maps are generated. Hence, throughout all the tests, we use 3,000 as the size of the augmented data set.

3.4.5.5 Test 4: Hyper-parameter Selection

Hyper-parameters play an important role in our generative models. In this test, we will study two critical hyper-parameters: the selection of the layers from the pre-trained VGG-19 network in Eq. (3.35) and the regularization parameter, γ, in Eq. (3.37).

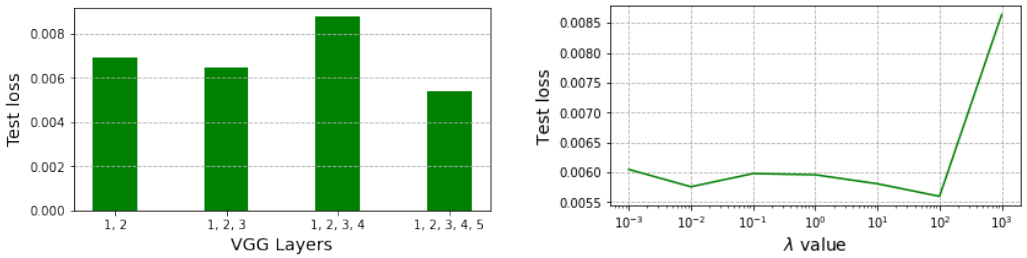


Figure 3.34 Visualization of the hyper-parameter versus loss values. (a) different combinations of layers selected from VGG-19 used in our VAE with perception loss. (b) Various values of the regularization parameter used in our VAE with regularization. (© 2022 IEEE)

To select the optimal VGG-19 layers for computing perception loss, there are 5 convolutional blocks in VGG-19 as shown in Fig. 3.23(b). We follow a similar idea in Gatys et al. (2015) to select effective layers. Particularly, we choose from following four combinations of **(A)** [conv1_1, conv2_1]; **(B)** [conv1_1, conv2_1, conv3_1]; **(C)** [conv1_1, conv2_1, conv3_1, conv4_1]; and **(D)** [conv1_1, conv2_1, conv3_1, conv4_1, conv5_1]. We train VAE with perception loss using different combinations of layers, and compute test loss of each case on the same test dataset (as shown in Fig. 3.34(a)). We can observe that VAE with perception loss reaches smallest value when using combination **(D)**. That gives us the indication to use first layer of 5 convolutional blocks of VGG-19 to compute perception loss.

Similarly, in order to select optimal λ, we choose 7 values evenly distributed on a log scale from

10^{-3} to 10^{3} (i.e. 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2} and 10^{3}). For each λ value, we train a VAE with regularization, and test it on a common test dataset. The resulting test loss is shown in Fig. 3.34(b). We observe that test loss is relative stable when $\lambda < 10^{3}$, and it reaches lowest value when $\lambda = 10^{2}$. So, $\lambda = 10^{2}$ becomes the optimal value for our problem.

3.4.6 Summary

In this work, we have developed several spatio-temporal data augmentation strategies using convolutional neural networks to enhance data-driven seismic imaging, particularly in reconstructing "rare events." Our data augmentation techniques not only incorporate various physics information—such as governing equations, observable perception, and physics phenomena—but also integrate this information through perceptual loss and advanced regularization techniques. We evaluated the performance of our generative models by imaging very small CO_2 leaks from the subsurface with InversionNet. Through detailed comparison and analysis, we demonstrate that incorporating physics information is crucial for generating realistic and physically consistent synthetic data. This enhanced data quality significantly improves the representativeness of the training set. These advancements are seamlessly combined with regularization from the $\mathcal{S}$ term in Equation (1.3).

CHAPTER 4

CONCLUSION

This dissertation has explored the landscape of regularization techniques in deep learning, dividing the discussion into theory-driven and data-driven approaches as outlined in Chapters 3 and 2, respectively.

The theory-driven section underscored the importance of understanding and implementing theory such as the PAC-Bayes bounds discussed in Section 2.2. By minimizing the upper bound of the generalization error, this dissertation illustrated how theoretical insights into regularization can lead to more effective training algorithms and improved model performance across various tasks. Conversely, in the data-driven section, this dissertation introduced architectures such as MagNet and STGNN (Sections 3.1 and 3.2) to encode specific patterns directly from the data. Furthermore, the application of physics-informed frameworks in Section 3.3 and data augmentation strategies in Section 3.4 demonstrated how the integration of physical laws and generative models could significantly enhance the representational capability and physical consistency of training datasets. The contributions of this dissertation lie in the integration of practical machine learning architectures and generalization theory to address both applied and fundamental challenges in the field.

The methodologies and frameworks developed in this dissertation open several promising avenues for future research. Below, we outline potential areas for extending and improving the work introduced in various sections of this thesis:

1. **PAC-Bayes Training** (Zhang et al., 2023): This work has primarily utilized Gaussian priors and posteriors. Exploring alternative distributions could uncover unique advantages. The complexity added by managing additional parameters such as λ and σ in PAC-Bayes training necessitates more efficient parameterization of the prior and posterior. Additionally, the optimization challenges introduced by these parameters underscore the need for a comprehensive convergence analysis to ensure the robustness and efficacy of the training algorithms.

2. **MagNet** (Zhang et al., 2021b): While MagNet extends naturally to weighted, directed graphs where all edges are directed, its application to weighted mixed graphs (containing both directed and undirected edges) remains unexplored. Moreover, the current implementation lacks an attention mechanism and does not scale well to large graphs. Future research could explore architectural enhancements to address these limitations, potentially incorporating scalable graph learning techniques and attention mechanisms to improve performance on large datasets.

3. **STGNN** (Zhang et al., 2022): Transferability remains a major concern in earthquake source characterization. Although STGNN does not rely on specific graph structures or seismic station distributions, its performance in regions that differ from the training area requires improvement. Addressing this challenge might involve leveraging recent advancements in foundation models Bommasani et al. (2021), which could provide new ways to enhance model robustness and adaptability across different geophysical environments.

4. **UPFWI** (Jin et al., 2021): The UPFWI model faces challenges with velocity maps where adjacent layers have very close velocity values, which could be improved by updating architectures such as Jagtap et al. (2020). Additionally, the computational demands for forward modeling, in terms of speed and memory due to the necessity of storing gradients for backpropagation, are substantial. Future iterations of this model could experiment with alternative loss functions, such as adversarial loss, and explore computational strategies to balance resource use and accuracy. Expanding the application of CNN-PDE integration to other inverse problems like medical imaging and flow estimation also holds significant potential.

5. **Seismic Data Augmentation** (Yang et al., 2022): Evaluating the quality of synthesized data remains a significant challenge. Unlike disciplines such as computer vision, where metrics like the Fréchet Inception Distance (FID) are commonly used, seismic data requires domain-specific metrics for effective evaluation. Additionally, choosing the domain for data augmentation

(velocity or seismic) impacts the utility of the synthetic data. While augmentation in the velocity domain leverages prominent spatiotemporal dynamics, augmentation in the seismic domain aligns more closely with real-world scenarios, presenting a trade-off that warrants further exploration.

These directions not only build on the work presented in this dissertation but also promise to advance the state of the art in machine learning. By bridging theoretical insights with practical applications, this work sets the stage for the development of more robust, interpretable, and effective deep learning models and training algorithms.

BIBLIOGRAPHY

(2019). Basic research needs for scientific machine learning. Technical report, U.S. the Department of Energy Advanced Scientific Computing Research.

Adler, A., Araya-Polo, M., and Poggio, T. (2021). Deep learning for seismic inverse problems: toward the acceleration of geophysical analysis workflows. IEEE Signal Processing Magazine, 38(2):89–119.

Alquier, P. and Guedj, B. (2018). Simpler pac-bayesian bounds for hostile data. Machine Learning, 107(5):887–902.

Andriushchenko, M., Croce, F., Müller, M., Hein, M., and Flammarion, N. (2023). A modern look at the relationship between sharpness and generalization. arXiv preprint arXiv:2302.07011.

Antoniou, A., Storkey, A., and Edwards, H. (2017). Data augmentation generative adversarial networks. arXiv preprint arXiv:1711.04340.

Appriou, D., Bonneville, A., Zhou, Q., and Gasperikova, E. (2020). Time-lapse gravity monitoring of CO_2 migration based on numerical modeling of a faulted storage complex. International Journal Greenhouse Gas Control, 95:102956.

Araya-Polo, M., Jennings, J., Adler, A., and Dahlke, T. (2018). Deep-learning tomography. The Leading Edge, 37(1):58–66.

Atwood, J. and Towsley, D. (2016). Diffusion-convolutional neural networks. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, Advances in Neural Information Processing Systems, volume 29, pages 1993–2001. Curran Associates, Inc.

Audibert, J.-Y. and Catoni, O. (2011). Robust linear least squares regression. The Annals of Statistics, 39(5):2766 – 2794.

Barrett, D. G. and Dherin, B. (2020). Implicit gradient regularization. arXiv preprint arXiv:2009.11162.

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation, 15(6):1373–1396.

Benson, A. R., Gleich, D. F., and Leskovec, J. (2016). Higher-order organization of complex networks. Science, 353(6295):163–166.

Bergen, K. J., Johnson, P. A., Maarten, V., and Beroza, G. C. (2019). Machine learning for data-driven discovery in solid earth geoscience. Science, 363(6433).

Berthelot, D., Raffel, C., Roy, A., and Goodfellow, I. (2018). Understanding and improving

interpolation in autoencoders via an adversarial regularizer. arXiv preprint arXiv:1807.07543.

Beskardes, G. D., Hole, J. A., Wang, K., Michaelides, M., and Wu, Q. (2018). A comparison of earthquake back-projection imaging methods for dense local arrays. Geophysical Journal International, 212(3):1986–2002.

Beyreuther, M., Barsch, R., Krischer, L., Megies, T., Behr, Y., and Wassermann, J. (2010). Obspy: A python toolbox for seismology. Seismological Research Letters, 81(3):530–533.

Biggs, F. and Guedj, B. (2021). Differentiable pac–bayes objectives with partially aggregated neural networks. Entropy, 23(10):1280.

Birkholzer, J., Oldenburg, C., and Zhou, Q. (2015). CO_2 migration and pressure evolution in deep saline aquifers. International Journal of Greenhouse Gas Control, 40:203–220.

Bishop, C. (2006). Pattern Recognition and Machine Learning. Springer Science & Business Media.

Bishop, C. M. (1995). Training with noise is equivalent to tikhonov regularization. Neural computation, 7(1):108–116.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In International conference on machine learning, pages 1613–1622. PMLR.

Bojchevski, A. and Günnemann, S. (2017). Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. arXiv preprint arXiv:1707.03815.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258.

Boonyasiriwat, C., Valasek, P., Routh, P., Cao, W., Schuster, G. T., and Macy, B. (2009). An efficient multiscale method for time-domain waveform tomography. Geophysics, 74(6):WCC59–WCC68.

Bovet, A. and Grindrod, P. (2020). The activity of the far right on telegram v2.11.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2014). Spectral networks and deep locally connected networks on graphs. In International Conference on Learning Representations (ICLR).

Bunks, C., Saleck, F., Zaleski, S., and Chavent, G. (1995). Multiscale seismic waveform inversion. Geophysics, 60(5):1457–1473.

Burstedde, C. and Ghattas, O. (2009). Algorithmic strategies for full waveform inversion: 1D experiments. Geophysics, 74(6):37–46.

Buscheck, T., Mansoor, K., Yang, X., Wainwright, H., and Carroll, S. (2019). Downhole pressure and chemical monitoring for CO_2 and brine leak detection in aquifers above a CO_2 storage reservoir. International Journal Greenhouse Gas Control, 91:102812.

Casado, I., Ortega, L. A., Masegosa, A. R., and Pérez, A. (2024). Pac-bayes-chernoff bounds for unbounded losses. arXiv preprint arXiv:2401.01148.

Cattaneo, M. D., Klusowski, J. M., and Shigida, B. (2023). On the implicit bias of adam. arXiv preprint arXiv:2309.00079.

Chen, T. and Huang, L. (2020). Optimal design of microseismic monitoring network: Synthetic study for the Kimberlina CO_2 storage demonstration site. International Journal Greenhouse Gas Control, 95:102981.

Chung, F. (2005). Laplacians and the Cheeger inequality for directed graphs. Annals of Combinatorics, 9(1):1–19.

Chung, F. and Kempton, M. (2013). A local clustering algorithm for connection graphs. In International Workshop on Algorithms and Models for the Web-Graph, pages 26–43. Springer.

Chung, F. R. and Graham, F. C. (1997). Spectral graph theory. Number 92. American Mathematical Soc.

Cloninger, A. (2017). A note on markov normalized magnetic eigenmaps. Applied and Computational Harmonic Analysis, 43(2):370 – 380.

Cohen, J., Kaur, S., Li, Y., Kolter, J. Z., and Talwalkar, A. (2020). Gradient descent on neural networks typically occurs at the edge of stability. In International Conference on Learning Representations.

Cohen, J. M., Kaur, S., Li, Y., Kolter, J. Z., and Talwalkar, A. (2021). Gradient descent on neural networks typically occurs at the edge of stability. arXiv preprint arXiv:2103.00065.

Coifman, R. R. and Lafon, S. (2006). Diffusion maps. Applied and computational harmonic analysis, 21(1):5–30.

Collino, F. and Tsogka, C. (2001). Application of the perfectly matched absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media. Geophysics, 66(1):294–307.

Cucuringu, M., Li, H., Sun, H., and Zanetti, L. (2020). Hermitian matrices for clustering directed graphs: insights and applications. In International Conference on Artificial Intelligence and Statistics, pages 983–992. PMLR.

Damian, A., Ma, T., and Lee, J. D. (2021). Label noise sgd provably prefers flat global minimizers. Advances in Neural Information Processing Systems, 34:27449–27461.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In Advances in Neural Information Processing Systems 29, pages 3844–3852.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. CoRR, abs/1810.04805.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, Advances in Neural Information Processing Systems, volume 28, pages 2224–2232. Curran Associates, Inc.

Dziugaite, G. K., Hsu, K., Gharbieh, W., Arpino, G., and Roy, D. (2021). On the role of data in pac-bayes bounds. In International Conference on Artificial Intelligence and Statistics, pages 604–612. PMLR.

Dziugaite, G. K. and Roy, D. M. (2017). Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In Proceedings of the 33rd Annual Conference on Uncertainty in Artificial Intelligence (UAI).

Dziugaite, G. K. and Roy, D. M. (2018). Data-dependent pac-bayes priors via differential privacy. Advances in neural information processing systems, 31.

F. de Resende, B. M. and F. Costa, L. d. (2020). Characterization and comparison of large directed networks through the spectra of the magnetic laplacian. Chaos: An Interdisciplinary Journal of Nonlinear Science, 30(7):073141.

Fanuel, M., Alaíz, C. M., Ángela Fernández, and Suykens, J. A. (2018). Magnetic eigenmaps for the visualization of directed networks. Applied and Computational Harmonic Analysis, 44:189–199.

Fanuel, M., Alaiz, C. M., and Suykens, J. A. (2017). Magnetic eigenmaps for community detection in directed networks. Physical Review E, 95(2):022302.

Feng, S., Lin, Y., and Wohlberg, B. (2020). Physically realistic training data construction for data-driven full-waveform inversion and traveltime tomography. In SEG Technical Program Expanded Abstracts, pages 3472–3476.

Feng, S., Lin, Y., and Wohlberg, B. (2021). Multiscale data-driven seismic full-waveform inversion with field data study. IEEE Transactions on Geoscience and Remote Sensing, pages 1–14.

Fey, M., Lenssen, J. E., Weichert, F., and Leskovec, J. (2021). GNNAutoScale: Scalable and

expressive graph neural networks via historical embeddings. In International Conference on Machine Learning (ICML).

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. (2020). Sharpness-aware minimization for efficiently improving generalization. In International Conference on Learning Representations.

Furutani, S., Shibahara, T., Akiyama, M., Hato, K., and Aida, M. (2020). Graph signal processing for directed graphs based on the hermitian laplacian. In Machine Learning and Knowledge Discovery in Databases, pages 447–463.

Gajewski, D., Anikiev, D., Kashtan, B., and Tessmer, E. (2007). Localization of seismic events by diffraction stacking. In SEG Technical Program Expanded Abstracts 2007, pages 1287–1291.

Gasteiger, J., Bojchevski, A., and Günnemann, S. (2018). Predict then propagate: Graph neural networks meet personalized pagerank. arXiv preprint arXiv:1810.05997.

Gastpar, M., Nachum, I., Shafer, J., and Weinberger, T. (2023). Fantastic generalization measures are nowhere to be found.

Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576.

Geiping, J., Goldblum, M., Pope, P. E., Moeller, M., and Goldstein, T. (2021). Stochastic training is not necessary for generalization. arXiv preprint arXiv:2109.14119.

Germain, P., Bach, F., Lacoste, A., and Lacoste-Julien, S. (2016). Pac-bayesian theory meets bayesian inference. Advances in Neural Information Processing Systems, 29.

Ghosh, A., Lyu, H., Zhang, X., and Wang, R. (2022). Implicit regularization in heavy-ball momentum accelerated stochastic gradient descent. In The Eleventh International Conference on Learning Representations.

Gomez, R., Yang, J., Lin, Y., Theiler, J., and Wohlberg, B. (2020). Physics-consistent data-driven waveform inversion with adaptive data augmentation. arXiv preprint (also accepted in IEEE Geoscience and Remote Sensing Letters).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680.

Guitton, A. (2012). Blocky regularization schemes for full waveform inversion. Geophysical Prospecting, 60:870–884.

Guo, K. and Mohar, B. (2017). Hermitian adjacency matrix of digraphs and mixed graphs. Journal of Graph Theory, 85(1):217–248.

Ha-Duong, M. and Keith, D. (2003). Carbon storage: the economic efficiency of storing CO_2 in leaky reservoirs. Technological Choices for Sustainability, 5:181–189.

Haddouche, M., Guedj, B., Rivasplata, O., and Shawe-Taylor, J. (2021). Pac-bayes unleashed: Generalisation bounds with unbounded losses. Entropy, 23(10):1330.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. Advances in neural information processing systems, 30.

Hammond, D. K., Vandergheynst, P., and Gribonval, R. (2011). Wavelets on graphs via spectral graph theory. Applied and Computational Harmonic Analysis, 30(2):129–150.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision, pages 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778.

He, Q. and Wang, Y. (2021). Reparameterized full-waveform inversion using deep neural networks. Geophysics, 86(1):V1–V13.

He, Y., Reinert, G., and Cucuringu, M. (2021). Digrac: Digraph clustering with flow imbalance. arXiv preprint arXiv:2106.05194.

Herbrich, R. and Graepel, T. (2000). A pac-bayesian margin bound for linear classifiers: Why svms work. Advances in neural information processing systems, 13.

Hernández-García, A. and König, P. (2018). Data augmentation instead of explicit regularization. arXiv preprint arXiv:1806.03852.

Holland, M. (2019). Pac-bayes under potentially heavy tails. Advances in Neural Information Processing Systems, 32.

Hsu, W.-N., Zhang, Y., and Glass, J. (2017). Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation. In 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pages 16–23. IEEE.

Hu, W., Abubakar, A., and Habashy, T. (2009). Simultaneous multifrequency inversion of full-waveform seismic data. Geophysics, 74(2):1–14.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4700–4708.

Hutton, K., Woessner, J., and Hauksson, E. (2010). Earthquake monitoring in southern california for seventy-seven years (1932–2008). Bulletin of the Seismological Society of America, 100(2):423–446.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning, pages 448–456. PMLR.

Jagtap, A. D., Kawaguchi, K., and Karniadakis, G. E. (2020). Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. Journal of Computational Physics, 404:109136.

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., and Bengio, S. (2019). Fantastic generalization measures and where to find them. In International Conference on Learning Representations.

Jin, P., Zhang, X., Chen, Y., Huang, S. X., Liu, Z., and Lin, Y. (2021). Unsupervised learning of full-waveform inversion: Connecting cnn and partial differential equation in a loop. arXiv preprint arXiv:2110.07584.

Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In European Conference on Computer Vision, pages 694–711. Springer.

Jordan, P. and Wagoner, J. (2017). Characterizing construction of existing wells to a CO_2 storage target: The Kimberlina site, California. Technical report, U.S. Department of Energy - Office of Fossil Energy.

Karniadakis, G., Kevrekidis, I., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). Physics-informed machine learning. Nature Reviews Physics, 3:422–440.

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., and Aila, T. (2020). Training generative adversarial networks with limited data. arXiv preprint arXiv:2006.06676v2.

Kaufman, S., Rosset, S., Perlich, C., and Stitelman, O. (2012). Leakage in data mining: Formulation, detection, and avoidance. ACM Transactions on Knowledge Discovery from Data (TKDD), 6(4):1–21.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. arXiv preprint arXiv:1609.04836.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional

networks. arXiv preprint arXiv:1609.02907.

Klicpera, J., Bojchevski, A., and Günnemann, S. (2019a). Predict then propagate: Graph neural networks meet personalized pagerank. In ICLR.

Klicpera, J., Groß, J., and Günnemann, S. (2019b). Directional message passing for molecular graphs. In International Conference on Learning Representations.

Kobak, D., Lomond, J., and Sanchez, B. (2020). The optimal ridge penalty for real-world high-dimensional data can be zero or negative due to the implicit ridge regularization. J. Mach. Learn. Res., 21:169–1.

Kong, Q., Trugman, D. T., Ross, Z. E., Bianco, M. J., Meade, B. J., and Gerstoft, P. (2019). Machine learning in seismology: Turning data into insights. Seismological Research Letters, 90(1):3–14.

Kriegerowski, M., Petersen, G. M., Vasyura-Bathke, H., and Ohrnberger, M. (2019). A deep convolutional neural network for localization of clustered earthquakes based on multistation full waveforms. Seismological Research Letters, 90:510 – 516.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25:1097–1105.

Kukačka, J., Golkov, V., and Cremers, D. (2017). Regularization for deep learning: A taxonomy. arXiv preprint arXiv:1710.10686.

Kuzborskij, I. and Szepesvári, C. (2019). Efron-stein pac-bayesian inequalities. arXiv preprint arXiv:1909.01931.

Lagaris, I. E., Likas, A., and Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. IEEE transactions on neural networks, 9(5):987–1000.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. Nature, 401(6755):788–791.

Lee, S. and Jang, C. (2022). A new characterization of the edge of stability based on a sharpness measure aware of batch gradient distribution. In The Eleventh International Conference on Learning Representations.

Letarte, G., Germain, P., Guedj, B., and Laviolette, F. (2019). Dichotomize and generalize: Pac-bayesian binary activated deep neural networks. Advances in Neural Information Processing Systems, 32.

Levie, R., Huang, W., Bucci, L., Bronstein, M. M., and Kutyniok, G. (2019). Transferability of spectral graph convolutional neural networks. arXiv preprint arXiv:1907.12972.

Lewkowycz, A., Bahri, Y., Dyer, E., Sohl-Dickstein, J., and Gur-Ari, G. (2020). The large learning rate phase of deep learning: the catapult mechanism. arXiv preprint arXiv:2003.02218.

Li, L., Tan, J., Schwarz, B., Stanek, F., Poiata, N., Shi, P., Diekmann, L., Eisner, L., and Gajewski, D. (2020a). Recent advances and challenges of waveform-based seismic location methods at multiple scales. Reviews of Geophysics, page e2019RG000667.

Li, S., Liu, B., Ren, Y., Chen, Y., Yang, S., Wang, Y., and Jiang, P. (2020b). Deep-learning inversion of seismic data. IEEE Transactions on Geoscience and Remote Sensing, 58(3):2135–2149.

Li, Z., Meier, M.-A., Hauksson, E., Zhan, Z., and Andrews, J. (2018). Machine learning seismic wave discrimination: Application to earthquake early warning. Geophysical Research Letters, 45(10):4773–4779.

Li, Z. and van der Baan, M. (2016). Microseismic event localization by acoustic time reversal extrapolation. Geophysics, 81(3):KS123–KS134.

Lieb, E. H. and Loss, M. (1993). Fluxes, Laplacians, and Kasteleyn's theorem. In Statistical Mechanics, pages 457–483. Springer.

Lin, Y. and Huang, L. (2015a). Acoustic- and elastic-waveform inversion using a modified Total-Variation regularization scheme. Geophysical Journal International, 200(1):489–502.

Lin, Y. and Huang, L. (2015b). Quantifying subsurface geophysical properties changes using double-difference seismic-waveform inversion with a modified Total-Variation regularization scheme. Geophysical Journal International, 203(3):2125–2149.

Lin, Y. and Huang, L. (2017). Building subsurface velocity models with sharp interfaces using interface-guided seismic full-waveform inversion. Pure and Applied Geophysics, 174(11):4035–4055.

Lin, Y., Syracuse, E. M., Maceira, M., Zhang, H., and Larmat, C. (2015). Double-difference traveltime tomography with edge-preserving regularization and a priori interfaces. Geophysical Journal International, 201(2):574–594.

Liu, B., Yang, S., Ren, Y., Xu, X., Jiang, P., and Chen, Y. (2021). Deep-learning seismic full-waveform inversion for realistic structural models. Geophysics, 86(1):R31 – R44.

Liu, X., Zou, Y., Kong, L., Diao, Z., Yan, J., Wang, J., Li, S., Jia, P., and You, J. (2018). Data augmentation via latent space interpolation for image classification. In 2018 24th International Conference on Pattern Recognition (ICPR), pages 728–733. IEEE.

Livni, R. and Moran, S. (2020). A limitation of the pac-bayes framework. Advances in Neural Information Processing Systems, 33:20543–20553.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

Loshchilov, I. and Hutter, F. (2018). Decoupled weight decay regularization. In International Conference on Learning Representations.

Lumley, D. (2001). Time-lapse seismic reservoir monitoring. Geophysics, 66:50–53.

Luo, P., Wang, X., Shao, W., and Peng, Z. (2018). Towards understanding regularization in batch normalization. arXiv preprint arXiv:1809.00846.

Luo, Y., Zhu, L., Wan, Z., and Lu, B. (2020). Data augmentation for enhancing EEG-based emotion recognition with deep generative models. Journal of Neural Engineering, 17(5):056021.

Ma, Y., Hao, J., Yang, Y., Li, H., Jin, J., and Chen, G. (2019). Spectral-based graph convolutional network for directed graphs. arXiv:1907.08990.

Marques, A. G., Segarra, S., and Mateos, G. (2020). Signal processing on directed graphs: The role of edge directionality when processing and learning from network data. IEEE Signal Processing Magazine, 37(6):99–116.

Maurer, A. (2004). A note on the pac bayesian theorem. arXiv preprint cs/0411099.

McAllester, D. A. (1998). Some pac-bayesian theorems. In Proceedings of the eleventh annual conference on Computational learning theory, pages 230–234.

McAllester, D. A. (1999). Pac-bayesian model averaging. In Proceedings of the twelfth annual conference on Computational learning theory, pages 164–170.

McBrearty, I. W. and Beroza, G. C. (2022). Earthquake location and magnitude estimation with graph neural networks. arXiv preprint arXiv:2203.05144.

Mernyei, P. and Cangea, C. (2020). Wiki-cs: A wikipedia-based benchmark for graph neural networks. arXiv preprint arXiv:2007.02901.

Milletari, F., Navab, N., and Ahmadi, S.-A. (2016). V-net: Fully convolutional neural networks for volumetric medical image segmentation. In 2016 fourth international conference on 3D vision (3DV), pages 565–571. Ieee.

Mohar, B. (2020). A new kind of hermitian matrices for digraphs. Linear Algebra and its Applications, 584:343–352.

Monti, F., Otness, K., and Bronstein, M. M. (2018). Motifnet: A motif-based graph convolutional network for directed graphs. In 2018 IEEE Data Science Workshop, pages 225–228.

Moseley, B., Nissen-Meyer, T., and Markham, A. (2020). Deep learning for fast simulation of seismic waves in complex media. Solid Earth, 11(4):1527–1549.

Mosser, L., Dubrule, O., and Blunt, M. J. (2020). Stochastic seismic waveform inversion using generative adversarial networks as a geological prior. Mathematical Geosciences, 52(1):53–79.

Mousavi, S. M. and Beroza, G. C. (2020a). Bayesian-deep-learning estimation of earthquake location from single-station observations. IEEE Transactions on Geoscience and Remote Sensing, pages 1 – 14.

Mousavi, S. M. and Beroza, G. C. (2020b). A machine-learning approach for earthquake magnitude estimation. Geophysical Research Letters, 47(1):e2019GL085976.

Münchmeyer, J., Bindi, D., Leser, U., and Tilmann, F. (2020). The transformer earthquake alerting model: a new versatile approach to earthquake early warning. Geophysical Journal International, 225(1):646–656.

Münchmeyer, J., Bindi, D., Leser, U., and Tilmann, F. (2021). Earthquake magnitude and location estimation from real time seismic waveforms with a transformer network. Geophysical Journal International, 226(2):1086–1104.

Nagarajan, V. and Kolter, J. Z. (2019). Uniform convergence may be unable to explain generalization in deep learning. Advances in Neural Information Processing Systems, 32.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), pages 807–814.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. (2021). Deep double descent: Where bigger models and more data hurt. Journal of Statistical Mechanics: Theory and Experiment, 2021(12):124003.

Nanometrics Seismological Instruments (2013). Nanometrics research network.

Neelakantan, A., Vilnis, L., Le, Q. V., Sutskever, I., Kaiser, L., Kurach, K., and Martens, J. (2015). Adding gradient noise improves learning for very deep networks. arXiv preprint arXiv:1511.06807.

Neyshabur, B., Tomioka, R., and Srebro, N. (2014). In search of the real inductive bias: On the role of implicit regularization in deep learning. arXiv preprint arXiv:1412.6614.

Nishizaki, H. (2017). Data augmentation and feature extraction using variational autoencoder for acoustic modeling. In 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), pages 1222–1227. IEEE.

Oring, A., Yakhini, Z., and Hel-Or, Y. (2020). Autoencoder image interpolation by shaping the latent space. arXiv preprint arXiv:2008.01487v2.

Ortega, A., Frossard, P., Kovačević, J., Moura, J. M., and Vandergheynst, P. (2018). Graph signal processing: Overview, challenges, and applications. Proceedings of the IEEE, 106(5):808–828.

Orvieto, A., Kersting, H., Proske, F., Bach, F., and Lucchi, A. (2022). Anticorrelated noise injection for improved generalization. In International Conference on Machine Learning, pages 17094–17116. PMLR.

Ovcharenko, O., Kazei, V., Peter, D., and Alkhalifah, T. (2019). Style transfer for generation of realistically textured subsurface models. In SEG Technical Program Expanded Abstracts 2019, pages 2393–2397. Society of Exploration Geophysicists.

Palmer, W. R. and Zheng, T. (2021). Spectral clustering for directed networks. Studies in Computational Intelligence, 943.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. (2020). Geom-gcn: Geometric graph convolutional networks. arXiv preprint arXiv:2002.05287.

Perez-Ortiz, M., Rivasplata, O., Guedj, B., Gleeson, M., Zhang, J., Shawe-Taylor, J., Bober, M., and Kittler, J. (2021). Learning pac-bayes priors for probabilistic neural networks. arXiv preprint arXiv:2109.10304.

Pérez-Ortiz, M., Rivasplata, O., Shawe-Taylor, J., and Szepesvári, C. (2021). Tighter risk certificates for neural networks. The Journal of Machine Learning Research, 22(1):10326–10365.

Perol, T., Gharbi, M., and Denolle, M. (2018). Convolutional neural network for earthquake detection and location. Science Advances, 4:e1700578.

Pesicek, J. D., Child, D., Artman, B., and Cieslik, K. (2014). Picking versus stacking in a modern microearthquake location: Comparison of results from a surface passive seismic monitoring array in Oklahoma. Geophysics, 79(6):KS61–KS68.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378:686–707.

Ramírez, A. and Lewis, W. (2010). Regularization and full-waveform inversion: A two-step approach. In 80th Annual International Meeting, SEG, Expanded Abstracts, pages 2773–2778.

Ren, Y., Nie, L., Yang, S., Jiang, P., and Chen, Y. (2021). Building complex seismic velocity models for deep learning inversion. IEEE Access, 4(1):R31 – R44.

Richardson, A. (2018). Generative adversarial networks for model order reduction in seismic

full-waveform inversion. arXiv preprint arXiv:1806.00828.

Rivasplata, O., Kuzborskij, I., Szepesvári, C., and Shawe-Taylor, J. (2020). Pac-bayes analysis beyond the usual bounds. Advances in Neural Information Processing Systems, 33:16833–16845.

Rivasplata, O., Tankasali, V. M., and Szepesvári, C. (2019). Pac-bayes with backprop. arXiv preprint arXiv:1908.07380.

Rodríguez-Gálvez, B., Thobaben, R., and Skoglund, M. (2023). More pac-bayes bounds: From bounded losses, to losses with general tail behaviors, to anytime-validity. arXiv preprint arXiv:2306.12214.

Rojas-Gómez, R., Yang, J., Lin, Y., Theiler, J., and Wohlberg, B. (2020). Physics-consistent data-driven waveform inversion with adaptive data augmentation. IEEE Geoscience and Remote Sensing Letters.

Ross, Z. E., Yue, Y., Meier, M.-A., Hauksson, E., and Heaton, T. H. (2019). Phaselink: A deep learning approach to seismic phase association. Journal of Geophysical Research: Solid Earth, 124(1):856–869.

Rozemberczki, B., Allen, C., and Sarkar, R. (2019). Multi-scale attributed node embedding. arXiv preprint arXiv:1909.13021.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How does batch normalization help optimization? Advances in neural information processing systems, 31.

Shawe-Taylor, J. and Williamson, R. C. (1997). A pac analysis of a bayesian estimator. In Proceedings of the tenth annual conference on Computational learning theory, pages 2–9.

Shen, H. and Shen, Y. (2021). Array-based convolutional neural networks for automatic detection and 4d localization of earthquakes in hawai 'i. Seismological Society of America, 92(5):2961–2971.

Shi, J. and Malik, J. (1997). Normalized cuts and image segmentation. In Proceedings of IEEE computer society conference on computer vision and pattern recognition, pages 731–737. IEEE.

Shorten, C. and Khoshgoftaar, T. (2019). A survey on image data augmentation for deep learning. Journal of Big Data, 6(1):1–48.

Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., and Webb, R. (2017). Learning from simulated and unsupervised images through adversarial training. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2107–2116.

Sim, A., Wiatrak, M., Brayne, A., Creed, P., and Paliwal, S. (2021). Directed graph embeddings in pseudo-riemannian manifolds. In Meila, M. and Zhang, T., editors, Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event,

volume 139 of Proceedings of Machine Learning Research, pages 9681–9690. PMLR.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Sixt, L., Wild, B., and Landgraf, T. (2018). Rendergan: Generating realistic labeled data. Frontiers in Robotics and AI, 5:66.

Smith, L. I. (2002). A tutorial on principal components analysis.

Smith, S. L., Dherin, B., Barrett, D. G., and De, S. (2021). On the origin of implicit regularization in stochastic gradient descent. arXiv preprint arXiv:2101.12176.

Spielman, D. A. and Teng, S.-H. (2004). Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, pages 81–90.

Sun, J., Innanen, K. A., and Huang, C. (2021). Physics-guided deep learning for seismic inversion with hybrid training and uncertainty analysis. Geophysics, 86(3):R303–R317.

Sun, L., Gao, H., Pan, S., and Wang, J.-X. (2020). Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. Computer Methods in Applied Mechanics and Engineering, 361:112732.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2818–2826.

Thiemann, N., Igel, C., Wintenberger, O., and Seldin, Y. (2017). A strongly quasiconvex pac-bayesian bound. In International Conference on Algorithmic Learning Theory, pages 466–492. PMLR.

Tiira, T. (1999). Detecting teleseismic events using artificial neural networks. Comput. Geosci., 25:929 – 938.

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A. P., Keysers, D., Uszkoreit, J., et al. (2021). Mlp-mixer: An all-mlp architecture for vision. In Thirty-Fifth Conference on Neural Information Processing Systems.

Tong, Z., Liang, Y., Sun, C., Li, X., Rosenblum, D., and Lim, A. (2020a). Digraph inception convolutional networks. In NeurIPS.

Tong, Z., Liang, Y., Sun, C., Rosenblum, D. S., and Lim, A. (2020b). Directed graph convolutional network. arXiv:2004.13970.

Treister, E. and Haber, E. (2016). Full waveform inversion guided by travel time tomography. SIAM

Journal on Scientific Computing, 39:S587–S609.

United States Geological Survey and California Geological Survey (2022). Quaternary fault and fold database for the united state.

van den Ende, M. P. and Ampuero, J.-P. (2020). Automated seismic source characterisation using deep graph neural networks. Geophysical Research Letters, page e2020GL088690.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph Attention Networks. International Conference on Learning Representations.

Virieux, J. and Operto, S. (2009). An overview of full-waveform inversion in exploration geophysics. Geophysics, 74(6):WCC1–WCC26.

Wang, J., Perez, L., et al. (2017). The effectiveness of data augmentation in image classification using deep learning. Convolutional Neural Networks Vis. Recognit, 11(2017):1–8.

Wang, J. and Teng, T. (1995). Artificial neural network-based seismic detector. Bull. Seismol. Soc. Am., 85:308 – 319.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R. (2020). Towards physics-informed deep learning for turbulent flow prediction. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 1457–1466.

Wang, Y. (2015). Frequencies of the Ricker wavelet. Geophysics, 80(2):A31–A37.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019). Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog), 38(5):1–12.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4):600–612.

Wei, C., Kakade, S., and Ma, T. (2020). The implicit and explicit regularization effects of dropout. In International conference on machine learning, pages 10181–10192. PMLR.

Willard, J., Jia, X., Xu, S., Steinbach, M., and Kumar, V. (2020). Integrating physics-based modeling with machinelearning: A survey. arXiv preprint arXiv:2003.04919v4.

Wu, X., Geng, Z., Shi, Y., Pham, N., Fomel, S., and Caumon, G. (2020a). Building realistic structure models to train convolutional neural networks for seismic structural interpretation. Geophysics, 85(4):WA27–WA39.

Wu, Y. and Lin, Y. (2019). InversionNet: An efficient and accurate data-driven full waveform inversion. IEEE Transactions on Computational Imaging, 6(1):419–433.

Wu, Y. and McMechan, G. A. (2019). Parametric convolutional neural network-domain full-waveform inversion. Geophysics, 84(6):R881–R896.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2020b). A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1):4–24.

Xi, Z., Li, J., Chen, M., and Wei, S. (2021). Pyfk: A fast mpi and cuda accelerated python package for calculating synthetic seismograms based on the frequencywavenumber method. In AGU Fall Meeting Abstracts, volume 2021, pages S15E–0288.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018). How powerful are graph neural networks? arXiv preprint arXiv:1810.00826.

Yang, F. and Ma, J. (2019). Deep-learning inversion: A next-generation seismic velocity model building method. Geophysics, 84(4):R583–R599.

Yang, X., Buscheck, T., Mansoor, K., Wang, Z., Gao, K., Huang, L., Wainwright, H., and Carroll, S. (2019). Assessment of geophysical monitoring methods for detection of brine and CO_2 leakage in drinking water aquifers. International Journal Greenhouse Gas Control, 90:102803.

Yang, Y., Zhang, X., Guan, Q., and Lin, Y. (2022). Making invisible visible: Data-driven seismic inversion with spatio-temporally constrained data augmentation. IEEE Transactions on Geoscience and Remote Sensing, 60:1–16, copyright © 2022 IEEE.

Yano, K., Shiina, T., Kurata, S., Kato, A., Komaki, F., Sakai, S., and Hirata, N. (2021). Graph-partitioning based convolutional neural network for earthquake detection using a seismic array. Journal of Geophysical Research: Solid Earth, 126(5):e2020JB020269.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2021a). Understanding deep learning (still) requires rethinking generalization. Communications of the ACM, 64(3):107–115.

Zhang, H. and Thurber, C. H. (2003). Double-difference tomography: The method and its application to the Hayward Fault, California. Bulletin of the Seismological Society of America, 93(5):1875–1889.

Zhang, R., Isola, P., Efros, A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. arXiv preprint arXiv:1801.03924v2.

Zhang, X., Ghosh, A., Liu, G., and Wang, R. (2023). Unleashing the power of pac-bayes training for unbounded loss.

Zhang, X., He, Y., Brugnone, N., Perlmutter, M., and Hirn, M. (2021b). Magnet: A neural network for directed graphs. Advances in neural information processing systems, 34:27003–27015.

Zhang, X., Reichard-Flynn, W., Zhang, M., Hirn, M., and Lin, Y. (2022). Spatiotemporal graph convolutional networks for earthquake source characterization. Journal of Geophysical Research: Solid Earth, 127(11):e2022JB024401.

Zhang, X., Wang, Z., Liu, D., and Ling, Q. (2019). Dada: Deep adversarial data augmentation for extremely low data regime classification. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2807–2811. IEEE.

Zhang, X., Zhang, J., Yuan, C., Liu, S., Chen, Z., and Li, W. (2020). Locating induced earthquakes with a network of seismic station in Oklahoma via a deep learning method. Scientific Report, 10.

Zhang, X., Zhang, M., and Tian, X. (2021c). Real-time earthquake early warning with deep learning: Application to the 2016 m 6.0 central apennines, italy earthquake. Geophysical Research Letters, 48(5):2020GL089394.

Zhang, Z. and Lin, Y. (2020). Data-driven seismic waveform inversion: A study on the robustness and generalization. IEEE Transactions on Geoscience and Remote sensing, 58(10):6900–6913.

Zhang, Z., Rector, J. W., and Nava, M. J. (2017). Simultaneous inversion of multiple microseismic data for event locations and velocity model with bayesian inference. Geophysics, 82(3):KS27–KS39.

Zhao, Z., Zhang, Z., Chen, T., Singh, S., and Zhang, H. (2020). Image augmentations for GAN training. arXiv preprint arXiv:2006.02595v1.

Zhebel, O. and Eisner, L. (2015). Simultaneous microseismic event localization and source mechanism determination. Geophysics, 80(1):KS1–KS9.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2018a). Graph neural networks: A review of methods and applications. arXiv preprint arXiv:1812.08434.

Zhou, W., Veitch, V., Austern, M., Adams, R. P., and Orbanz, P. (2018b). Non-vacuous generalization bounds at the imagenet scale: a pac-bayesian compression approach. arXiv preprint arXiv:1804.05862.

Zhou, Z., Lin, Y., Zhang, Z., Wu, Y., Wang, Z., Dilmore, R., and Guthrie, G. (2019). A data-driven CO_2 leakage detection using seismic data and spatial-temporal densely connected convolutional neural networks. International Journal of Greenhouse Gas Control, 90:102790.

Zhu, W. and Beroza, G. C. (2019). Phasenet: a deep-neural-network-based seismic arrival-time picking method. Geophysical Journal International, 216(1):261–273.

Zhu, W., Mousavi, S. M., and Beroza, G. C. (2019a). Seismic signal denoising and decomposition using deep neural networks. IEEE Transactions on Geoscience and Remote Sensing, 57(11):9476–9488.

Zhu, W., Xu, K., Darve, E., Biondi, B., and Beroza, G. C. (2021). Integrating deep neural networks with full-waveform inversion: Reparametrization, regularization, and uncertainty quantification. Geophysics, 87(1):1–103.

Zhu, Y., Zabaras, N., Koutsourelakis, P.-S., and Perdikaris, P. (2019b). Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. Journal of Computational Physics, 394:56–81.

APPENDIX A

UNLOCKING TUNING-FREE GENERALIZATION: MINIMIZING THE PAC-BAYES BOUND WITH TRAINABLE PRIORS

A.1 Proofs

A.1.1 Proofs of Theorem 2.2.6

Theorem A.1.1. *Given a prior $\mathcal{P}_{\boldsymbol{\lambda}}$ parametrized by $\boldsymbol{\lambda} \in \Lambda$ over the hypothesis set $\mathcal{H}$. Fix $\boldsymbol{\lambda} \in \Lambda$, $\delta \in (0,1)$ and $\gamma \in [\gamma_1, \gamma_2]$. For any choice of i.i.d m-sized training dataset $\mathcal{S}$ according to $\mathcal{D}$, and all posterior distributions Q over $\mathcal{H}$, we have*

$$\mathbb{E}_{\mathbf{h}\sim Q}\ell(\mathbf{h};\mathcal{D}) \leq \mathbb{E}_{\mathbf{h}\sim Q}\ell(\mathbf{h};\mathcal{S}) + \frac{1}{\gamma m}(\log\frac{1}{\delta} + \mathrm{KL}(Q||\mathcal{P}_{\boldsymbol{\lambda}})) + \gamma K(\boldsymbol{\lambda}) \tag{A.1}$$

holds with probability at least $1-\delta$ when $\ell(\mathbf{h},\cdot)$ satisfies Definition 2.2.5 with bound $K(\boldsymbol{\lambda})$.

Proof. Firstly, in the bounded interval $\gamma \in [\gamma_1, \gamma_2]$, we bound the difference of the expected loss over the posterior distribution evaluated on the training dataset $\mathcal{S}$ and $\mathcal{D}$ with the KL divergence between the posterior distribution Q and prior distribution $\mathcal{P}_{\boldsymbol{\lambda}}$ evaluated over a hypothesis space $\mathcal{H}$.

For $\gamma \in [\gamma_1, \gamma_2]$,

$$\begin{aligned}
&\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[\exp\left(\gamma m(\mathbb{E}_{\mathbf{h}\sim Q}\ell(\mathbf{h};\mathcal{D}) - \mathbb{E}_{\mathbf{h}\sim Q}\ell(\mathbf{h};\mathcal{S})) - \mathrm{KL}(Q||\mathcal{P}_{\boldsymbol{\lambda}})\right)] \\
=&\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[\exp\left(\gamma m(\mathbb{E}_{\mathbf{h}\sim Q}\ell(\mathbf{h};\mathcal{D}) - \mathbb{E}_{\mathbf{h}\sim Q}\ell(\mathbf{h};\mathcal{S})) - \mathbb{E}_{\mathbf{h}\sim Q}\log\frac{\mathrm{d}Q}{\mathrm{d}\mathcal{P}_{\boldsymbol{\lambda}}}(\mathbf{h})\right)] &\text{(A.2)}\\
\leq&\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}\mathbb{E}_{\mathbf{h}\sim Q}[\exp\left(\gamma m(\ell(\mathbf{h};\mathcal{D}) - \ell(\mathbf{h};\mathcal{S})) - \log\frac{\mathrm{d}Q}{\mathrm{d}\mathcal{P}_{\boldsymbol{\lambda}}}(\mathbf{h})\right)] &\text{(A.3)}\\
=&\mathbb{E}_{\mathbf{h}\sim\mathcal{P}_{\boldsymbol{\lambda}}}\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[\exp(\gamma m(\ell(\mathbf{h};\mathcal{D}) - \ell(\mathbf{h};\mathcal{S})))], &\text{(A.4)}
\end{aligned}$$

where $\mathrm{d}Q/\mathrm{d}\mathcal{P}$ denotes the Radon-Nikodym derivative.

In (A.2), we use $\mathrm{KL}(Q||\mathcal{P}_\lambda) = \mathbb{E}_{\mathbf{h}\sim Q}\log\frac{\mathrm{d}Q}{\mathrm{d}\mathcal{P}_\lambda}(\mathbf{h})$. From (A.2) to (A.3), Jensen's inequality is used over the convex exponential function. Since this argument holds for any Q, we have

$$\begin{aligned}
&\sup_{Q\in\mathbf{Q}}\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[\exp\left(\gamma m(\mathbb{E}_{\mathbf{h}\sim Q}\ell(\mathbf{h};\mathcal{D}) - \mathbb{E}_{\mathbf{h}\sim Q}\ell(\mathbf{h};\mathcal{S})) - \mathrm{KL}(Q||\mathcal{P}_{\boldsymbol{\lambda}})\right)] \leq &\text{(A.5)}\\
&\quad\mathbb{E}_{\mathbf{h}\sim\mathcal{P}_{\boldsymbol{\lambda}}}\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[\exp(\gamma m(\ell(\mathbf{h};\mathcal{D}) - \ell(\mathbf{h};\mathcal{S})))] &\text{(A.6)}
\end{aligned}$$

Let $X = \ell(\mathbf{h}; \mathcal{D}) - \ell(\mathbf{h}; \mathcal{S})$, then X is centered with $\mathbb{E}[X] = 0$. Then, by Definition 2.2.5,

$$\exists K(\boldsymbol{\lambda}),\ \mathbb{E}_{\mathbf{h}\sim\mathcal{P}_\lambda}\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[\exp(\gamma m X)] \leq \exp(m\gamma^2 K(\boldsymbol{\lambda})). \tag{A.7}$$

Using Markov's inequality, (A.8) holds with probability at least $1-\delta$.

$$\exp(\gamma m X) \leq \frac{\exp(m\gamma^2 K(\boldsymbol{\lambda}))}{\delta}. \tag{A.8}$$

Combining (A.5) and (A.8), the following inequality holds with probability at least $1-\delta$.

$$\begin{aligned}
&\sup_{Q\in\mathbb{Q}} \exp(\gamma m(\mathbb{E}_{\mathbf{h}\sim Q}\ell(\mathbf{h};\mathcal{D}) - \mathbb{E}_{\mathbf{h}\sim Q}\ell(\mathbf{h};\mathcal{S})) - \mathrm{KL}(Q||\mathcal{P}_{\boldsymbol{\lambda}})) \leq \frac{\exp(m\gamma^2 K(\boldsymbol{\lambda}))}{\delta} \\
\Rightarrow &\gamma m(\mathbb{E}_{\mathbf{h}\sim Q}\ell(\mathbf{h};\mathcal{D}) - \mathbb{E}_{\mathbf{h}\sim Q}\ell(\mathbf{h};\mathcal{S})) - \mathrm{KL}(Q||\mathcal{P}_{\boldsymbol{\lambda}}) \leq \log\frac{1}{\delta} + m\gamma^2 K(\boldsymbol{\lambda}), \forall Q \\
\Rightarrow &\mathbb{E}_{\mathbf{h}\sim Q}\ell(\mathbf{h};\mathcal{D}) \leq \mathbb{E}_{\mathbf{h}\sim Q}\ell(\mathbf{h};\mathcal{S}) + \frac{1}{\gamma m}(\log\frac{1}{\delta} + \mathrm{KL}(Q||\mathcal{P}_{\boldsymbol{\lambda}})) + \gamma K(\boldsymbol{\lambda}),\ \ \forall Q.
\end{aligned} \tag{A.9}$$

The bound A.9 is exactly the statement of the Theorem.

□

A.1.2 Proof of Theorem 2.2.10

Theorem A.1.2. *Let $n(\varepsilon) := \mathcal{N}(\Lambda, \|\cdot\|, \varepsilon)$ be the covering number of the set of the prior parameters. Under Assumption 2.2.8 and Assumption 2.2.9, the following inequality holds for the minimizer $(\hat{\mathbf{h}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}})$ of upper bound in* (A.1) *with probability as least $1-\epsilon$:*

$$\begin{aligned}
\mathbb{E}_{\mathbf{h}\sim Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})}\ell(\mathbf{h};\mathcal{D}) &\leq \mathbb{E}_{\mathbf{h}\sim Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})}\ell(\mathbf{h};\mathcal{S}) + \frac{1}{\hat{\gamma}m}\left[\log\frac{n(\varepsilon)+\frac{\gamma_2-\gamma_1}{\varepsilon}}{\epsilon} + \mathrm{KL}(Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\hat{\boldsymbol{\lambda}}})\right] + \hat{\gamma}K(\hat{\boldsymbol{\lambda}}) + \eta \\
&= L_{PAC}(\hat{\mathbf{h}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}}) + \eta
\end{aligned} \tag{A.10}$$

holds for any $\epsilon, \varepsilon > 0$, where $\eta = B\varepsilon + C(\eta_1(\varepsilon) + \eta_2(\varepsilon)) + \frac{\log(n(\varepsilon)+\frac{\gamma_2-\gamma_1}{\varepsilon})}{\gamma_1 m}$, with $C = \frac{1}{\gamma_1 m} + \gamma_2$, and $B := \sup_{\boldsymbol{\lambda}\in\Lambda}\frac{1}{m\gamma_1^2}(\mathrm{KL}(Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\boldsymbol{\lambda}}) + \log\frac{1}{\delta}) + K(\boldsymbol{\lambda})$.

Proof: In this proof, we extend our PAC-Bayes bound with data-independent priors to data-dependent ones that accommodate the error when the prior distribution is parameterized and optimized over a finite set of parameters $\mathfrak{P} = \{P_{\boldsymbol{\lambda}}, \boldsymbol{\lambda} \in \Lambda \subseteq \mathbb{R}^k\}$ with a much smaller dimension

than the model itself. Let $\mathbb{T}(\Lambda, \|\cdot\|, \varepsilon)$ be an ε-cover of the set Λ, which states that for any $\boldsymbol{\lambda} \in \Lambda$, there exists a $\tilde{\boldsymbol{\lambda}} \in \mathbb{T}(\Lambda, \|\cdot\|, \varepsilon)$, such that $||\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}}|| \leq \varepsilon$.

Now we select the posterior distribution as $Q_{\boldsymbol{\sigma}}(\mathbf{h})$, parameterized by $\mathbf{h}$ and $\boldsymbol{\sigma} \in \mathbb{R}^d$, where $\mathbf{h}$ represents the mean of the posterior, and $\boldsymbol{\sigma}$ accounts for the variations in each model parameter from this mean. Assuming the prior $\mathcal{P}$ is parameterized by $\boldsymbol{\lambda} \in \mathbb{R}^k$ ($k \ll d$).

Then the PAC-Bayes bound A.1 holds already for any $(\hat{\mathbf{h}}, \gamma, \hat{\boldsymbol{\sigma}}, \boldsymbol{\lambda})$, with fixed $\boldsymbol{\lambda} \in \Lambda$ and $\gamma \in [\gamma_1, \gamma_2]$, i.e.,

$$\mathbb{E}_{\tilde{\mathbf{h}} \sim Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})} \ell(\tilde{\mathbf{h}}; \mathcal{D}) \leq \mathbb{E}_{\tilde{\mathbf{h}} \sim Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})} \ell(\tilde{\mathbf{h}}; \mathcal{S}) + \frac{1}{\gamma m}(\log \frac{1}{\delta} + \mathrm{KL}(Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}}) || \mathcal{P}_{\boldsymbol{\lambda}})) + \gamma K(\boldsymbol{\lambda}) \tag{A.11}$$

with probability over $1 - \delta$.

Now, for the collection of $\boldsymbol{\lambda}$s in the ε-net $\mathbb{T}(\Lambda, \|\cdot\|, \varepsilon)$, by the union bound, the PAC-Bayes bound uniformly holds on the ε-net with probability at least $1 - |\mathbb{T}|\delta = 1 - n(\varepsilon)\delta$. For an arbitrary $\boldsymbol{\lambda} \in \Lambda$, its distance to the ε-net is at most ε. Then under Assumption 2.2.8 and Assumption 2.2.9, we have:

$$\min_{\tilde{\boldsymbol{\lambda}} \in \mathbb{T}} |\mathrm{KL}(Q||\mathcal{P}_{\boldsymbol{\lambda}}) - \mathrm{KL}(Q||\mathcal{P}_{\tilde{\boldsymbol{\lambda}}})| \leq \eta_1(\|\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}}\|) \leq \eta_1(\varepsilon),$$

and

$$\min_{\tilde{\boldsymbol{\lambda}} \in \mathbb{T}} |K(\boldsymbol{\lambda}) - K(\tilde{\boldsymbol{\lambda}})| \leq \eta_2(\|\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}}\|) \leq \eta_2(\varepsilon).$$

Similarly, for γ, a ε-net on its range $\gamma_1 \leq \gamma \leq \gamma_2$ is the uniform grid with a grid separation ε, so the net contains $\frac{\gamma_2 - \gamma_1}{\varepsilon}$ points. By the union bound, requiring the PAC-Bayes bound to uniformly hold for all the γ within this ε-net induces an extra probability of failure of $\frac{\gamma_2 - \gamma_1}{\varepsilon}\delta$. So, the total probability of failure is $n(\varepsilon)\delta + \frac{\gamma_2 - \gamma_1}{\varepsilon}\delta$.

For an arbitrary $\gamma \in \Gamma$, and $\Gamma := \{\gamma \in [\gamma_1, \gamma_2]\}$, its distance to the ε-net $\mathbb{T}'$ is at most ε, we have:

$$\begin{aligned}
\min_{\tilde{\gamma} \in \mathbb{T}'} |L_{PAC}(\hat{\mathbf{h}}, \gamma, \hat{\boldsymbol{\sigma}}, \boldsymbol{\lambda}) - L_{PAC}(\hat{\mathbf{h}}, \tilde{\gamma}, \hat{\boldsymbol{\sigma}}, \boldsymbol{\lambda})| &= \frac{1}{m}\left(\mathrm{KL}(Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\boldsymbol{\lambda}}) + \log\frac{1}{\delta}\right)\left|\frac{1}{\gamma} - \frac{1}{\tilde{\gamma}}\right| + |\gamma - \tilde{\gamma}|K(\boldsymbol{\lambda}) \\
&= \left(\frac{1}{m\gamma\tilde{\gamma}}(\mathrm{KL}(Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\boldsymbol{\lambda}}) + \log\frac{1}{\delta}) + K(\boldsymbol{\lambda})\right)|\gamma - \tilde{\gamma}| \\
&\leq \left(\frac{1}{m\gamma_1^2}(\mathrm{KL}(Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\boldsymbol{\lambda}}) + \log\frac{1}{\delta}) + K(\boldsymbol{\lambda})\right)\varepsilon \\
&\leq B\varepsilon,
\end{aligned}$$

where $B := \sup_{\boldsymbol{\lambda}\in\Lambda} \frac{1}{m\gamma_1^2}(\mathrm{KL}(Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\boldsymbol{\lambda}}) + \log\frac{1}{\delta}) + K(\boldsymbol{\lambda})$, clearly, B is a constant depending on the range of the parameters.

With the three inequalities above, we can control the PAC-Bayes loss at the given $\boldsymbol{\lambda}$ and γ as follows:

$$
\begin{aligned}
&\min_{\tilde{\boldsymbol{\lambda}}\in\mathbb{T},\tilde{\gamma}\in\mathbb{T}'} |L_{PAC}(\hat{\mathbf{h}},\gamma,\hat{\boldsymbol{\sigma}},\boldsymbol{\lambda}) - L_{PAC}(\hat{\mathbf{h}},\tilde{\gamma},\hat{\boldsymbol{\sigma}},\tilde{\boldsymbol{\lambda}})| \\
&\leq \min_{\tilde{\gamma}\in\mathbb{T}'} |L_{PAC}(\hat{\mathbf{h}},\gamma,\hat{\boldsymbol{\sigma}},\boldsymbol{\lambda}) - L_{PAC}(\hat{\mathbf{h}},\tilde{\gamma},\hat{\boldsymbol{\sigma}},\boldsymbol{\lambda})| + \min_{\tilde{\boldsymbol{\lambda}}\in\mathbb{T}} |L_{PAC}(\hat{\mathbf{h}},\tilde{\gamma},\hat{\boldsymbol{\sigma}},\boldsymbol{\lambda}) - L_{PAC}(\hat{\mathbf{h}},\tilde{\gamma},\hat{\boldsymbol{\sigma}},\tilde{\boldsymbol{\lambda}})| \\
&\leq B\varepsilon + \frac{1}{\tilde{\gamma}m}\eta_1(\varepsilon) + \tilde{\gamma}\eta_2(\varepsilon) \\
&\leq B\varepsilon + \frac{1}{\gamma_1 m}\eta_1(\varepsilon) + \gamma_2\eta_2(\varepsilon) \\
&\leq B\varepsilon + C(\eta_1(\varepsilon) + \eta_2(\varepsilon))
\end{aligned}
$$

where $C = \frac{1}{\gamma_1} + \gamma_2$ and $\gamma_1 \leq \gamma \leq \gamma_2$. Since this inequality holds for any $\boldsymbol{\lambda} \in \Lambda$ and $\gamma \in \Gamma$, it certainly holds for the optima $\hat{\boldsymbol{\lambda}}$ and $\hat{\gamma}$. Combining this with (A.11), we have

$$
\mathbb{E}_{\mathbf{h}\sim Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})}\ell(\mathbf{h};\mathcal{D}) \leq L_{PAC}(\hat{\mathbf{h}},\hat{\gamma},\hat{\boldsymbol{\sigma}},\hat{\boldsymbol{\lambda}}) + B\varepsilon + C(\eta_1(\varepsilon) + \eta_2(\varepsilon)),
$$

where $B := \sup_{\boldsymbol{\lambda}\in\Lambda} \frac{1}{m\gamma_1^2}(\mathrm{KL}(Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\boldsymbol{\lambda}}) + \log\frac{1}{\delta}) + K(\boldsymbol{\lambda})$.

Now taking $\epsilon := (n(\varepsilon) + \frac{\gamma_2-\gamma_1}{\varepsilon})\delta$ to be the previously calculated probability of failure, we get, with probability $1-\epsilon$, it holds that

$$
\begin{aligned}
\mathbb{E}_{\mathbf{h}\sim Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})}\ell(\mathbf{h};\mathcal{D}) &\leq \mathbb{E}_{\mathbf{h}\sim Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})}\ell(\mathbf{h};\mathcal{S}) + \frac{1}{\hat{\gamma}m}\left[\log\frac{n(\varepsilon)+\frac{\gamma_2-\gamma_1}{\varepsilon}}{\epsilon} + \mathrm{KL}(Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\hat{\boldsymbol{\lambda}}})\right] \qquad \text{(A.12)} \\
&\quad + \hat{\gamma}K(\hat{\boldsymbol{\lambda}}) + B\varepsilon + C(\eta_1(\varepsilon) + \eta_2(\varepsilon)) \\
&\leq L_{PAC}(\hat{\mathbf{h}},\hat{\gamma},\hat{\boldsymbol{\sigma}},\hat{\boldsymbol{\lambda}}) + \eta \qquad \text{(A.13)}
\end{aligned}
$$

and the proof is completed. □

A.1.3 KL divergence of the Gaussian prior and posterior

For a k-layer network, the prior is written as $\mathcal{P}_{\boldsymbol{\lambda}}(\boldsymbol{\theta}_0)$, where $\boldsymbol{\theta}_0$ is the random initialized model parameterized by $\boldsymbol{\theta}_0$ and $\boldsymbol{\lambda} \in \mathbb{R}_+^k$ is the vector containing the variance for each layer. The set of

all such priors is denoted by $\mathfrak{P} := \{\mathcal{P}_{\boldsymbol{\lambda}}(\boldsymbol{\theta}_0), \boldsymbol{\lambda} \in \Lambda \subseteq \mathbb{R}^k, \boldsymbol{\theta}_0 \in \Theta\}$. In the PAC-Bayes training, we select the posterior distribution to be centered around the trained model parameterized by $\boldsymbol{\theta}$, with independent anisotropic variance. Specifically, for a network with d trainable parameters, the posterior is $Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta}) := \mathcal{N}(\boldsymbol{\theta}, \mathrm{diag}(\boldsymbol{\sigma}))$, where $\boldsymbol{\theta}$ (the current model) is the mean and $\boldsymbol{\sigma} \in \mathbb{R}_+^d$ is the vector containing the variance for each trainable parameter. The set of all posteriors is $\mathfrak{Q} := \{Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta}), \boldsymbol{\sigma} \in \Sigma, \boldsymbol{\theta} \in \Theta\}$, and the KL divergence between all such prior and posterior in $\mathfrak{P}$ and $\mathfrak{Q}$ is:

$$\mathrm{KL}(Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P}_{\boldsymbol{\lambda}}(\boldsymbol{\theta}_0)) = \frac{1}{2}\sum_{i=1}^{k}\left[-\mathbf{1}_{d_i}^\top \log(\boldsymbol{\sigma}_i) + d_i(\log(\boldsymbol{\lambda}_i) - 1) + \frac{\|\boldsymbol{\sigma}_i\|_1 + \|(\boldsymbol{\theta} - \boldsymbol{\theta}_0)_i\|_2^2)}{\boldsymbol{\lambda}_i}\right], \quad \text{(A.14)}$$

where $\boldsymbol{\sigma}_i$, $(\boldsymbol{\theta} - \boldsymbol{\theta}_0)_i$ are vectors denoting the variances and weights for the i-th layer, respectively, and λ_i is the scalar variance for the i-th layer. $d_i = \dim(\boldsymbol{\sigma}_i)$, and $\mathbf{1}_{d_i}$ denotes an all-ones vector of length d_i[1].

Scalar prior is a special case of the layerwise prior by setting all entries of $\boldsymbol{\lambda}$ to be equal, for which the KL divergence reduces to

$$\mathrm{KL}(Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P}_{\lambda}(\boldsymbol{\theta}_0)) = \frac{1}{2}\left[-\mathbf{1}_{d}^\top \log(\boldsymbol{\sigma}) + d(\log(\lambda) - 1) + \frac{1}{\lambda}(\|\boldsymbol{\sigma}\|_1 + \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2^2)\right]. \quad \text{(A.15)}$$

A.1.4 Proof of Corollary 2.2.11

Recall for the training, we proposed to optimize over all four variables: $\boldsymbol{\theta}$, γ, $\boldsymbol{\sigma}$, and $\boldsymbol{\lambda}$.

$$(\hat{\boldsymbol{\theta}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}}) = \arg\min_{\substack{\boldsymbol{\theta},\boldsymbol{\lambda},\boldsymbol{\sigma},\\ \gamma\in[\gamma_1,\gamma_2]}} \underbrace{\mathbb{E}_{\tilde{\boldsymbol{\theta}}\sim Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})}\ell(f_{\tilde{\boldsymbol{\theta}}}; \mathcal{S}) + \frac{1}{\gamma m}(\log\frac{1}{\delta} + \mathrm{KL}(Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P}_{\boldsymbol{\lambda}})) + \gamma K(\boldsymbol{\lambda})}_{\equiv L_{PAC}(\boldsymbol{\theta},\gamma,\boldsymbol{\sigma},\boldsymbol{\lambda})}. \quad \text{(A.16)}$$

Corollary A.1.3. *Assume all parameters for the prior and posterior are bounded, i.e., we restrict the model parameter $\boldsymbol{\theta}$, the posterior variance $\boldsymbol{\sigma}$ and the prior variance $\boldsymbol{\lambda}$, and the exponential moment $K(\boldsymbol{\lambda})$ all to be searched over bounded sets, $\Theta := \{\boldsymbol{\theta} \in \mathbb{R}^d : \|\boldsymbol{\theta}\|_2 \leq \sqrt{d}M\}$, $\Sigma := \{\boldsymbol{\sigma} \in \mathbb{R}_+^d : \|\boldsymbol{\sigma}\|_1 \leq dT\}$, $\Lambda =: \{\boldsymbol{\lambda} \in [e^{-a}, e^b]^k\}$, $\Gamma := \{\gamma \in [\gamma_1, \gamma_2]\}$, respectively, with fixed $M, T, a, b > 0$. Then,*

- *Assumption 2.2.8 holds with $\eta_1(x) = L_1 x$, where $L_1 = \frac{1}{2}\max\{d, e^a(2\sqrt{d}M + dT)\}$*

[1]Note that with a little ambiguity, the $\boldsymbol{\lambda}_i$ here has a different meaning from that in (A.20) and Algorithm A.1, here $\boldsymbol{\lambda}_i$ means the ith element in $\boldsymbol{\lambda}$, whereas in (A.20) and Algorithm A.1, $\boldsymbol{\lambda}_i$ means the ith element in the discrete set.

- *Assumption 2.2.9 holds with* $\eta_2(x) = L_2 x$, *where* $L_2 = \frac{1}{\gamma_1^2}\left(2dM^2e^{2a} + \frac{d(a+b)}{2}\right)$

- *With high probability, the PAC-Bayes bound for the minimizer of* (P) *has the form*

$$\mathbb{E}_{\boldsymbol{\theta}\sim Q_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})}\ell(f_{\boldsymbol{\theta}};\mathcal{D}) \leq L_{PAC}(\hat{\boldsymbol{\theta}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}}) + \eta,$$

where $\eta = \frac{k}{\gamma_1 m}\left(1 + \log\frac{2(CL+B)\Delta\gamma_1 m}{k}\right)$, $L = L_1 + L_2$, $\Delta := \max\{b + a, 2(\gamma_2 - \gamma_1)\}$, $B = \sup_{\boldsymbol{\lambda}\in\Lambda}\frac{1}{m\gamma_1^2}(\mathrm{KL}(Q_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})||\mathcal{P}_{\boldsymbol{\lambda}}) + \log\frac{1}{\delta}) + K(\boldsymbol{\lambda})$, *and* $C = \frac{1}{\gamma_1 m} + \gamma_2$.

Proof: We first prove the two assumptions are satisfied by the Gaussian family with bounded parameter spaces. To prove Assumption 2.2.8 is satisfied, let $v_i = \log 1/\lambda_i$, $i = 1, ..., k$ and perform a change of variable from λ_i to v_i. The weight of prior for the ith layer now becomes $\mathcal{N}(\mathbf{0}, e^{-v_i}\mathbf{I}_{d_i}))$, where d_i is the number of trainable parameters in the ith layer. It is straightforward to compute

$$\frac{\partial \mathrm{KL}(Q_{\boldsymbol{\sigma}}||\tilde{\mathcal{P}}_{\mathbf{v}})}{\partial v_i} = \frac{1}{2}[-d_i + e^{v_i}(\|\boldsymbol{\sigma}_i\|_1 + \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_{0,i}\|_2^2)],$$

where $\boldsymbol{\sigma}_i$, $\boldsymbol{\theta}_i$, $\boldsymbol{\theta}_{0,i}$ are the blocks of $\boldsymbol{\sigma}$, $\boldsymbol{\theta}$, $\boldsymbol{\theta}_0$, containing the parameters associated with the ith layer, respectively. Now, given the assumptions on the boundedness of the parameters, we have:

$$\|\nabla_{\mathbf{v}}\mathrm{KL}(Q_{\boldsymbol{\sigma}}||\tilde{\mathcal{P}}_{\mathbf{v}})\|_2 \leq \|\nabla_{\mathbf{v}}\mathrm{KL}(Q_{\boldsymbol{\sigma}}||\tilde{\mathcal{P}}_{\mathbf{v}})\|_1 \leq \frac{1}{2}\max\{d, e^a(2\sqrt{d}M + dT)\} \equiv L_1(d, M, T, a), \tag{A.17}$$

where we used the assumption $\|\boldsymbol{\sigma}\|_1 \leq dT$ and $\|\boldsymbol{\theta}_0\|_2, \|\boldsymbol{\theta}\|_2 \leq \sqrt{d}M$.

Equation A.17 says $L_1(d, M, T, a)$ is a valid Lipschitz bound on the KL divergence and therefore Assumption 2.2.8 is satisfied by setting $\eta_1(x) = L_1(d, M, T, a)x$.

Next, we prove Assumption 2.2.9 is satisfied. We use $K_{\min}(\boldsymbol{\lambda})$ defined in Definition 2.2.5 as the

$K(\boldsymbol{\lambda})$ in the PAC-Bayes training, and verify that it makes Assumption 2.2.9 hold.

$$
\begin{aligned}
&|K_{\min}(\boldsymbol{\lambda}_1) - K_{\min}(\boldsymbol{\lambda}_2)| \\
&= \left| \sup_{\gamma\in[\gamma_1,\gamma_2]} \frac{1}{\gamma^2} \log(\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}_{\boldsymbol{\lambda}_1}} \mathbb{E}_{z\sim\mathcal{D}}[\exp(\gamma\ell(f_{\boldsymbol{\theta}}; z))]) - \sup_{\gamma\in[\gamma_1,\gamma_2]} \frac{1}{\gamma^2} \log(\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}_{\boldsymbol{\lambda}_2}} \mathbb{E}_{z\sim\mathcal{D}}[\exp(\gamma\ell(f_{\boldsymbol{\theta}}; z))]) \right| \\
&\leq \sup_{\gamma\in[\gamma_1,\gamma_2]} \frac{1}{\gamma^2} \left| \log(\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}_{\boldsymbol{\lambda}_1}} \mathbb{E}_{z\sim\mathcal{D}}[\exp(\gamma\ell(f_{\boldsymbol{\theta}}; z))]) - \log(\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}_{\boldsymbol{\lambda}_2}} \mathbb{E}_{z\sim\mathcal{D}}[\exp(\gamma\ell(f_{\boldsymbol{\theta}}; z))]) \right| \\
&= \sup_{\gamma\in[\gamma_1,\gamma_2]} \frac{1}{\gamma^2} \left| \log(\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}_{\boldsymbol{\lambda}_2}} \mathbb{E}_{z\sim\mathcal{D}}[\exp(\gamma\ell(f_{\boldsymbol{\theta}}; z))] \frac{p_{\boldsymbol{\lambda}_1}(\boldsymbol{\theta})}{p_{\boldsymbol{\lambda}_2}(\boldsymbol{\theta})}) - \log(\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}_{\boldsymbol{\lambda}_2}} \mathbb{E}_{z\sim\mathcal{D}}[\exp(\gamma\ell(f_{\boldsymbol{\theta}}; z))]) \right| \\
&\leq \sup_{\gamma\in[\gamma_1,\gamma_2]} \frac{1}{\gamma^2} \sup_{\boldsymbol{\theta}\in\Theta} \left| \log \frac{p_{\boldsymbol{\lambda}_1}(\boldsymbol{\theta})}{p_{\boldsymbol{\lambda}_2}(\boldsymbol{\theta})} \right| \\
&\leq \frac{1}{\gamma_1^2} \sup_{\mathbf{h}\in\mathcal{H}} \left| \log \frac{p_{\boldsymbol{\lambda}_1}(\boldsymbol{\theta})}{p_{\boldsymbol{\lambda}_2}(\boldsymbol{\theta})} \right| \\
&\leq \frac{1}{\gamma_1^2} \left(2dM^2 e^{2a} + \frac{d(a+b)}{2} \right) \|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\|_2,
\end{aligned}
$$

where the first inequality used the property of the supremum, the $p_{\lambda_1}(\boldsymbol{\theta}), p_{\lambda_2}(\boldsymbol{\theta})$ in the fourth line denote the probability density function of Gaussian with mean $\boldsymbol{\theta}$ and variance parametrized by $\boldsymbol{\lambda}_1$, $\boldsymbol{\lambda}_2$ (i.e., $\boldsymbol{\lambda}_{1,i}$, $\boldsymbol{\lambda}_{2,i}$ are the variances for the ith layer), the second inequality use the fact that if $X(\mathbf{h})$ is a non-negative function of $\mathbf{h}$ and $Y(\mathbf{h})$ is a bounded function of $\mathbf{h}$, then

$$
|\mathbb{E}_{\mathbf{h}}(X(\mathbf{h})Y(\mathbf{h}))| \leq (\sup_{\mathbf{h}\in\mathcal{H}} |Y(\mathbf{h})|) \cdot \mathbb{E}_{\mathbf{h}} X(\mathbf{h}).
$$

The last inequality used the formula of the Gaussian density

$$
p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu) \right)
$$

and the boundedness of the parameters. Therefore, Assumption 2.2.9 is satisfied by setting $\eta_2(x) = L_2(d, M, \gamma_1, a)x$, where $L_2(d, M, \gamma_1, a) = \frac{1}{\gamma_1^2}\left(2dM^2 e^{2a} + \frac{d(a+b)}{2}\right)$.

Let $L(d, M, T, \gamma_1, a) = L_1(d, M, T, a) + L_2(d, M, \gamma_1, a)$. Then we can apply Theorem 2.2.10, to get with probability $1 - \epsilon$,

$$
\begin{aligned}
&\mathbb{E}_{\boldsymbol{\theta}\sim Q_{\hat{\sigma}}(\hat{\boldsymbol{\theta}})} \ell(f_{\boldsymbol{\theta}}; \mathcal{D}) \\
&\leq \mathbb{E}_{\boldsymbol{\theta}\sim Q_{\hat{\sigma}}(\hat{\boldsymbol{\theta}})} \ell(f_{\boldsymbol{\theta}}; \mathcal{S}) + \frac{1}{\hat{\gamma} m} \left[\log \frac{n(\varepsilon) + \frac{\gamma_2 - \gamma_1}{2\varepsilon}}{\epsilon} + \mathrm{KL}(Q_{\hat{\sigma}}(\hat{\boldsymbol{\theta}}) || \mathcal{P}_{\boldsymbol{\lambda}}) \right] + \hat{\gamma} K_{\min}(\hat{\boldsymbol{\lambda}}) + \qquad \text{(A.18)} \\
&\quad (CL(d, M, T, \gamma_1, a)) + B)\varepsilon.
\end{aligned}
$$

Here, we used $\eta_1(x) = L_1 x$ and $\eta_2(x) = L_2 x$. Note that for the set $[-b, a]^k$, the covering number $n(\varepsilon) = \mathcal{N}([-b, a]^k, |\cdot|, \varepsilon)$ is $\left(\frac{b+a}{2\varepsilon}\right)^k$, and the covering number $\frac{\gamma_2-\gamma_1}{2\varepsilon}$ for $\gamma \in [\gamma_1, \gamma_2]$.

We introduce a new variable $\rho > 0$, letting $\varepsilon = \frac{\rho}{2(CL(d,M,T,\gamma_1,a)+B)}$ and inserting it into equation (A.18), we obtain with probability $1 - \epsilon$:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\theta}\sim Q_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})}\ell(f_{\boldsymbol{\theta}}; \mathcal{D}) \\
&\leq \mathbb{E}_{\boldsymbol{\theta}\sim Q_{\hat{\sigma}}(\hat{\boldsymbol{\theta}})}\ell(f_{\boldsymbol{\theta}}; \mathcal{S}) + \frac{1}{\hat{\gamma}m}\left[\log\frac{1}{\epsilon} + \mathrm{KL}(Q_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})||\mathcal{P}_{\boldsymbol{\lambda}})\right] \\
&+ \hat{\gamma}K_{\min}(\hat{\boldsymbol{\lambda}}) + \rho + \frac{k}{\gamma_1 m}\log\frac{2(CL(d, M, T, \gamma_1, a) + B)\Delta}{\rho}.
\end{aligned}$$

where $\Delta := \max\{b + a, 2(\gamma_2 - \gamma_1)\}$.

Optimizing over ρ, we obtain:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\theta}\sim Q_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})}\ell(f_{\boldsymbol{\theta}}; \mathcal{D}) \\
&\leq \mathbb{E}_{\boldsymbol{\theta}\sim Q_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})}\ell(f_{\boldsymbol{\theta}}; \mathcal{S}) + \frac{1}{\hat{\gamma}m}\left[\log\frac{1}{\epsilon} + \mathrm{KL}(Q_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})||\mathcal{P}_{\boldsymbol{\lambda}})\right] \\
&+ \hat{\gamma}K_{\min}(\hat{\boldsymbol{\lambda}}) + \frac{k}{\gamma_1 m}\left(1 + \log\frac{2(CL(d, M, T, \gamma_1, a) + B)\Delta\gamma_1 m}{k}\right) \\
&= L_{PAC}(\hat{\boldsymbol{\theta}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}}) + \frac{k}{\gamma_1 m}\left(1 + \log\frac{2(CL(d, M, T, \gamma_1, a) + B)\Delta\gamma_1 m}{k}\right).
\end{aligned}$$

Hence we have

$$\mathbb{E}_{\boldsymbol{\theta}\sim Q_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})}\ell(f_{\boldsymbol{\theta}}; \mathcal{D}) \leq L_{PAC}(\hat{\boldsymbol{\theta}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}}) + \eta,$$

where

$$\begin{aligned}
\eta = \max(&\frac{1}{\gamma_1 m}(1 + \log(2(CL(d, M, T, \gamma_1, a) + B)(\gamma_2 - \gamma_1)\gamma_1 m)), \\
&\frac{k}{\gamma_1 m}\left(1 + \log\frac{2(CL(d, M, T, \gamma_1, a) + B)\Delta\gamma_1 m}{k}\right)).
\end{aligned}$$

□

Remark A.1.4. In defining the boundedness of the domain Θ of $\boldsymbol{\theta}$ in Corollary 2.2.11, we used $\sqrt{d}M$ as the bound. Here, the factor $\sqrt{d}$ (where d denotes the dimension of $\mathbf{h}$) is used to encapsulate the idea that if on average, the components of the weight are bounded by M, then the ℓ_2 norm would naturally be bounded by $\sqrt{d}M$. The same idea applies to the definition of Σ.

Remark A.1.5. Due to the above remark, M, T, a, b can be treated as dimension-independent constants that do not grow with the network size d. As a result, the constants L_1, L_2, L in Corollary 2.2.11, are dominated by d, and $L_1, L_2, L = O(d)$. This then implies the logarithm term in η scales as $O(\log d)$, which grows very mildly with the size. Therefore, Corollary 2.2.11 can be used as the generalization guarantee for large neural networks.

A.2 Algorithm Details

A.2.1 Algorithms to estimate $K(\boldsymbol{\lambda})$

In this section, we explain the algorithm to compute $K(\lambda)$. In previous literature, the moment bound K or its analog term in the PAC-Bayes bounds was often assumed to be a constant. One of our contributions is to allow K to vary with the variance λ of the prior, so if a small prior variance is found by PAC-Bayes training, then the corresponding K would also be small. We perform linear interpolation to approximate the function $K_{\min}(\lambda)$ defined in (2) of the main text. When λ is 1D, We first compute $K_{\min}(\lambda)$ on a finite grid of the domain of λ, by solving (A.19) below. With the computed function values on the grid $\{K_{\min}(\lambda_i)\}_i$, we can construct a piecewise linear function as the approximation of $K_{\min}(\lambda)$.

$$
\begin{aligned}
K_{\min}(\lambda_i) = \arg\min_{K>0} & K \\
& \text{s.t.} \exp\left(\gamma^2 K\right) \geq \frac{1}{nm} \sum_{l=1}^{n} \sum_{j=1}^{m} \exp(\gamma(\ell(f_{\boldsymbol{\theta}_l}; \mathcal{S}) - \ell(f_{\boldsymbol{\theta}_l}; z_j))), \qquad \text{(A.19)} \\
& \forall\, \gamma \in [\gamma_1, \gamma_2], \quad \boldsymbol{\theta}_l \sim \mathcal{N}(\boldsymbol{\theta}_0, \lambda_i), \quad \lambda_{\min} \leq \lambda_i \leq \lambda_{\max}
\end{aligned}
$$

where $\boldsymbol{\theta}_l \sim \mathcal{P}_{\lambda_i}(\boldsymbol{\theta}_0), l = 1, ..., n$, are samples from the prior distribution and are fixed when solving (A.19) for $K_{\min}(\boldsymbol{\lambda}_i)$. (A.19) is the discrete version of the formula (2) in the main text. This optimization problem is 1-dimensional, and the function in the constraint is monotonic in K, so it can be solved efficiently by the bisection method.

When extending this procedure to high dimension, where $\boldsymbol{\lambda}$ is a k-dimension vector, we need to set up a grid for the domain of $\boldsymbol{\lambda}$ in k-dimensional space and estimate $K_{\min}$ on each grid point, which is time-consuming when k is large. To address this issue, we propose to use the following

approximation:

$$\hat{K}(\max(\boldsymbol{\lambda}_i)) = \arg\min_{K>0} K$$
$$\text{s.t.} \exp\left(\gamma^2 K\right) \geq \frac{1}{nm}\sum_{l=1}^{n}\sum_{j=1}^{m}\exp(\gamma(\ell(f_{\boldsymbol{\theta}_l};\mathcal{S}) - \ell(f_{\boldsymbol{\theta}_l};z_j))),$$
$$\forall\ \gamma \in [\gamma_1,\gamma_2], \quad \boldsymbol{\theta}_l \sim \mathcal{N}(\boldsymbol{\theta}_0, \max(\boldsymbol{\lambda}_i)), \quad \lambda_{\min} \leq \max(\boldsymbol{\lambda}_i) \leq \lambda_{\max}, i = 1, ..., s \tag{A.20}$$

where $\boldsymbol{\lambda}_i$ is a random sample from the domain Λ of $\boldsymbol{\lambda}$. Since each $\boldsymbol{\lambda_i}$ is k-dimensional, $\max(\boldsymbol{\lambda_i})$ represents the maximum of the k coordinates.

The idea of this formulation A.20 is as follows, we use the 1D function $\hat{K}(\max(\boldsymbol{\lambda}_i))$ as a surrogate function of the original k-dimension function $K_{\min}(\boldsymbol{\lambda})$ (i.e. $K_{\min}(\boldsymbol{\lambda}) \leq \hat{K}(\max(\boldsymbol{\lambda}_i))$). Then estimating this 1D surrogate function is easy by using the bisection method. This procedure will certainly overestimate the true $K_{\min}(\boldsymbol{\lambda})$ but since the surrogate function is also a valid exponential moment bound, it is safe to be used as a replacement for the $K(\boldsymbol{\lambda})$ in our PAC-Bayes bound for training. In practice, we tried to use $\text{mean}(\boldsymbol{\lambda}_i)$ to replace $\max(\boldsymbol{\lambda}_i)$ to mitigate the over-estimation, but the final performance stays the same. The details of the whole procedure are presented in Algorithm A.1.

Algorithm A.1 Compute $K(\boldsymbol{\lambda})$ given a set of query priors

Input: γ_1 and γ_2, sampling time s of prior variances, the initial neural network weight $\boldsymbol{\theta}_0$, the training dataset $\mathcal{S} = \{z_i\}_{i=1}^m$, model sampling time $n = 10$
Output: the piece-wise linear interpolation $\tilde{K}(\boldsymbol{\lambda})$ for $K_{\min}(\boldsymbol{\lambda})$
Draw s random samples for the prior variances $\mathcal{V} = \{\boldsymbol{\lambda}_i \in \Lambda \subseteq \mathbb{R}^k, i = 1, ..., s\}$
Set up a discrete grid Γ for the interval $[\gamma_1, \gamma_2]$ of γ.
for $\boldsymbol{\lambda}_i \in \mathcal{V}$ **do**
  for $l = 1 : n$ **do**
    Sampling weights from the Gaussian distribution $\boldsymbol{\theta}_l \sim \mathcal{N}(\boldsymbol{\theta}_0, \boldsymbol{\lambda}_i)$
    Use $\boldsymbol{\theta}_l$, Γ and $\mathcal{S}$ to compute one term in the sum in (A.20)
  end for
  Solve $\hat{K}(\max(\boldsymbol{\lambda}_i))$ using (A.20)
end for
Fit a piece-wise linear function $\tilde{K}(\boldsymbol{\lambda})$ to the data $\{(\boldsymbol{\lambda}_i, \hat{K}(\max(\boldsymbol{\lambda}_i))\}_{i=1}^s$

A.2.2 PAC-Bayes Training with layerwise prior

Similar to Algorithm 2.1, our PAC-Bayes training with a layerwise prior is stated here in Algorithm A.2.

Algorithm A.2 PAC-Bayes training (layerwise prior)

Input: initial weight $\boldsymbol{\theta}_0 \in \mathbb{R}^d$, the number of layers k, T_1, $\lambda_1 = e^{-12}, \lambda_2 = e^2$, $\gamma_1 = 0.5, \gamma_2 = 10$, *// $T_1, \lambda_1, \lambda_2, \gamma_1, \gamma_2$ can be fixed in all experiments of Sec2.2.8.*
Output: trained model $\hat{\boldsymbol{\theta}}$, posterior noise level $\hat{\boldsymbol{\sigma}}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$, $\mathbf{v} \leftarrow \mathbf{1}_d \cdot \log(\frac{1}{d}\sum_{i=1}^d |\boldsymbol{\theta}_{0,i}|)$, $\mathbf{b} \leftarrow \mathbf{1}_k \cdot \log(\frac{1}{d}\sum_{i=1}^d |\boldsymbol{\theta}_{0,i}|)$ *// Initialization*
Obtain the estimated $\tilde{K}(\bar{\boldsymbol{\lambda}})$ with $\Lambda = [\lambda_1, \lambda_2]^k$ using (A.20) and Appendix A.2.1
// Stage 1
for epoch = $1 : T_1$ **do**
  for sampling one batch s from $\mathcal{S}$ **do**
    $\boldsymbol{\lambda} \leftarrow \exp(\mathbf{b})$, $\boldsymbol{\sigma} \leftarrow \exp(\mathbf{v})$ *// Ensure non-negative variances*
    Construct the covariance of $\mathcal{P}_{\boldsymbol{\lambda}}$ from $\boldsymbol{\lambda}$ *// Setting the variance of the weights in layer-i all to the scalar $\boldsymbol{\lambda}(i)$*
    Draw one $\tilde{\boldsymbol{\theta}} \sim \mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})$ and evaluate $\ell(f_{\tilde{\boldsymbol{\theta}}}; \mathcal{S})$, *// Stochastic version of $\mathbb{E}_{\tilde{\boldsymbol{\theta}} \sim \mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})} \ell(f_{\tilde{\boldsymbol{\theta}}}; \mathcal{S})$*
    Compute the KL-divergence as (A.14)
    Compute γ as (2.10)
    Compute the loss function $\mathcal{L}$ as L_{PAC} in (P)
    $\mathbf{b} \leftarrow \mathbf{b} + \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b}}$, $\mathbf{v} \leftarrow \mathbf{v} + \eta \frac{\partial \mathcal{L}}{\partial \mathbf{v}}$, $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$ *// Update all parameters*
  end for
end for
$\hat{\boldsymbol{\sigma}} \leftarrow \exp(\mathbf{v})$ *// Fix the noise level from now on*
// Stage 2
while not converge **do**
  for sampling one batch s from $\mathcal{S}$ **do**
    Draw one sample $\tilde{\boldsymbol{\theta}} \sim \mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\boldsymbol{\theta})$ and evaluate $\ell(f_{\tilde{\boldsymbol{\theta}}}; \mathcal{S})$ as $\tilde{\mathcal{L}}$, *// Noise injection*
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \frac{\partial \tilde{\mathcal{L}}}{\partial \boldsymbol{\theta}}$ *// Update model parameters*
  end for
end while
$\hat{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$

A.2.3 Regularizations in PAC-Bayes bound

Only noise injection and weight decay are essential from our derived PAC-Bayes bound. Since many factors in normal training, such as mini-batch and dropout, enhance generalization by some sort of noise injection, it is unsurprising that they can be substituted by the well-calibrated noise injection in PAC-Bayes training. Like most commonly used implicit regularizations (large lr, momentum, small batch size), dropout and batch-norm are also known to penalize the loss function's

sharpness indirectly. Wei et al. (2020) studies that dropout introduces an explicit regularization that penalizes *sharpness* and an implicit regularization that is analogous to the effect of stochasticity in small mini-batch stochastic gradient descent. Similarly, it is well-studied that batch-norm Luo et al. (2018) allows the use of a large learning rate by reducing the variance in the layer batches, and large allowable learning rates regularize *sharpness* through the edge of stability Cohen et al. (2020). As shown in the equation below, the first term (noise-injection) in our PAC-Bayes bound explicitly penalizes the Trace of the Hessian of the loss, which directly relates to sharpness and is quite similar to the regularization effect of batch-norm and dropout. During training, suppose the current posterior is $Q_{\hat{\sigma}}(\hat{\theta}) = \mathcal{N}(\hat{\theta}, \mathrm{diag}(\hat{\sigma}))$, then the training loss expectation over the posterior is:

$$\begin{aligned}
\mathbb{E}_{\theta\sim Q_{\hat{\sigma}}(\hat{\theta})}\ell(f_\theta;\mathcal{D}) &= \mathbb{E}_{\Delta\theta\sim Q_{\hat{\sigma}}(\mathbf{0})}\ell(f_{\hat{\theta}+\Delta\theta};\mathcal{D}) \\
&\approx \ell(f_{\hat{\theta}},\mathcal{D}) + \mathbb{E}_{\Delta\theta\sim Q_{\hat{\sigma}}(\mathbf{0})}(\ell(f_{\hat{\theta}};\mathcal{D})\Delta\theta + \frac{1}{2}\Delta\theta^\top\nabla^2\ell(f_{\hat{\theta}};\mathcal{D})\Delta\theta) \\
&= \ell(\hat{f}_\theta;\mathcal{D}) + \frac{1}{2}\mathrm{Tr}(\mathrm{diag}(\hat{\sigma})\nabla^2\ell(f_{\hat{\theta}};\mathcal{D})).
\end{aligned}$$

The second regularization term (weight decay) in the bound additionally ensures that the minimizer found is close to initialization. Although the relation of this regularizer to sharpness is not very clear, empirical results suggest that weight decay may have a separate regularization effect from sharpness. In brief, we state that the effect of sharpness regularization from dropout and batch norm can also be well emulated by noise injection with the additional effect of weight decay.

A.2.4 Deterministic Prediction

Recall that for any $\theta \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}^d_+$, we used $Q_\sigma(\theta)$ to denote the multivariate normal distribition with mean θ and covariance matrix $\mathrm{diag}(\sigma)$. If we rewrite the left-hand side of the PAC-Bayes bound by Taylor expansion, we have:

$$\begin{aligned}
\mathbb{E}_{\theta\sim Q_{\hat{\sigma}}(\hat{\theta})}\ell(f_\theta;\mathcal{D}) &= \mathbb{E}_{\Delta\theta\sim Q_{\hat{\sigma}}(\mathbf{0})}\ell(f_{\hat{\theta}+\Delta\theta};\mathcal{D}) \\
&\approx \ell(f_{\hat{\theta}},\mathcal{D}) + \mathbb{E}_{\Delta\theta\sim Q_{\hat{\sigma}}(\mathbf{0})}(\nabla\ell(f_{\hat{\theta}};\mathcal{D})^T\Delta\theta + \frac{1}{2}\Delta\theta^\top\nabla^2\ell(f_{\hat{\theta}};\mathcal{D})\Delta\theta) \qquad \text{(A.21)} \\
&= \ell(\hat{f}_\theta;\mathcal{D}) + \frac{1}{2}\mathrm{Tr}(\mathrm{diag}(\hat{\sigma})\nabla^2\ell(f_{\hat{\theta}};\mathcal{D})) \geq \ell(f_{\hat{\theta}};\mathcal{D}).
\end{aligned}$$

Recall here $\hat{\theta}$ and $\hat{\sigma}$ are the minimizers of the PAC-Bayes loss, obtained by solving the optimization problem (P). Equation (A.21) states that the deterministic predictor has a smaller

prediction error than the Bayesian predictor. However, note that the last inequality in (A.21) is derived under the assumption that the term $\nabla^2 \ell(\hat{f}_{\hat{\boldsymbol{\theta}}}; \mathcal{D})$ is positive-semidefinite. This is a reasonable assumption as $\hat{\boldsymbol{\theta}}$ is the local minimizer of the PAC-Bayes loss, and the PAC-Bayes loss is close to the population loss when the number of samples is large. Nevertheless, since this property only approximately holds, the presented argument can only serve as an intuition that shows the potential benefits of using the deterministic predictor.

A.3 Extended Experimental Details

We conducted experiments using eight A5000 GPUs with four AMD EPYC 7543 32-core Processors. To speed up the training process for posterior and prior variance, we utilized a warmup method that involved updating the noise level in the posterior of each layer as a scalar for the first 50 epochs and then proceeding with normal updates after the warmup period. This method only affects the convergence speed, not the generalization, and it was only used for large models in image classification.

A.3.1 Parameter Settings

Recall that the exponential momentum bound $K(\boldsymbol{\lambda})$ is estimated over a range $[\gamma_1, \gamma_2]$ of γ as per Definition 2.2.5. It means that we need the inequality

$$\mathbb{E}_{\mathbf{h}\sim\mathcal{P}_{\boldsymbol{\lambda}}}\mathbb{E}[\exp\left(\gamma(\mathbb{E}[X(\mathbf{h})] - X(\mathbf{h}))\right)] \leq \exp\left(\gamma^2 K(\boldsymbol{\lambda})\right)$$

to hold for any γ in this range. One needs to be a little cautious when choosing the upper bound γ_2, because if it is too large, then the empirical estimate of $\mathbb{E}_{\mathbf{h}\sim\mathcal{P}_{\boldsymbol{\lambda}}}\mathbb{E}[\exp\left(\gamma(\mathbb{E}[X(\mathbf{h})] - X(\mathbf{h}))\right)]$ would have too large of a variance. Therefore, we recommended γ_2 to be set to no more than 10 or 20. The choice of γ_1 also does not seem to be very crucial, so we have fixed it to 0.5 throughout.

For large datasets (like in MNIST or CIFAR10), m is large. Then, according to Theorem 2.2.10, we can set the range M, T, a, b of the trainable parameters to be very large with only a little increase of the bound (as M, T, a, b are inside the logarithm), and then during training, the parameters would not exceed these bounds even if we don't clip them. Hence, no clipping is needed for very large networks or with small networks with proper initializations. But when the dataset size m is small, or

the initialization is not good enough, then the correction term could be large, and clipping will be needed.

The clipping is also needed from the usual numerical stability point of view. As λ is in the denominator of the KL-divergence, it cannot be too close to 0. Because of this, in the numerical experiments on GNN and CNN13/CNN15, we clip the domain of λ at a lower bound of 0.1 and $5e-3$, respectively. For the VGG and Resnet experiments, the clipping λ is optional.

A.3.2 Baseline PAC-Bayes bounds for unbounded loss functions

We compared two baseline PAC-Bayes bounds when training CNNs with our layerwise PAC-Bayes bound. The bounds are expressed in our notation.

- SubGaussian (Corollary 4 of Germain et al. (2016)):

$$\mathbb{E}_{\boldsymbol{\theta}\sim Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})}\ell(f_{\boldsymbol{\theta}};\mathcal{D}) \leq \mathbb{E}_{\boldsymbol{\theta}\sim Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})}\ell(f_{\boldsymbol{\theta}};\mathcal{S}) + \frac{1}{m}(\log\frac{1}{\delta} + \mathrm{KL}(Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P})) + \frac{1}{2}s^2, \qquad \text{(A.22)}$$

  where s^2 is the variance factor by assuming the loss function ℓ is sub-Gaussian as defined below:

$$\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}}\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}\exp\left[\gamma(\ell(f_{\boldsymbol{\theta}};\mathcal{D}) - \ell(f_{\boldsymbol{\theta}};\mathcal{S}))\right] \leq \exp\left(\frac{\gamma^2 s^2}{2}\right), \forall\gamma\in\mathbb{R}^+.$$

- CGF (Theorem 9 of Rodríguez-Gálvez et al. (2023)):

$$\mathbb{E}_{\boldsymbol{\theta}\sim Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})}\ell(f_{\boldsymbol{\theta}};\mathcal{D}) \leq \mathbb{E}_{\boldsymbol{\theta}\sim Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})}\ell(f_{\boldsymbol{\theta}};\mathcal{S}) + \frac{1}{\gamma}\left((\log\frac{1}{\delta} + \mathrm{KL}(Q_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P})) + \psi(\gamma)\right), \qquad \text{(A.23)}$$

  where $\psi(\gamma)$ is a convex and continuously differentiable function defined on $[0, b)$ for some $b \in \mathbb{R}^+$ such that $\psi(\gamma) = \psi'(\gamma) = 0$ and $\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}}\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[\exp(\gamma(\ell(f_{\boldsymbol{\theta}};\mathcal{D}) - \ell(f_{\boldsymbol{\theta}};\mathcal{S})))] \leq \exp(\psi(\gamma))$ for all $\gamma \in [0, b)$. There is no specific form of $\psi(\gamma)$ provided in the original paper, so we set $\psi(\gamma) = K\gamma^2$. Moreover, γ is on the denominator of the bound, so we optimized γ when evaluating this bound and clipped γ to the same range $[0.5, 10)$ as we did to our algorithm.

A.3.3 Image classification

There is no data augmentation in the experiment results reported in the main text. The ones with data augmentation can be found below. For the layerwise prior, we treated each parameter in

the PyTorch object model.parameters() as an independent layer, i.e., the weights and bias of one convolution/batch-norm layer were treated as two different layers. The number of training epochs of Stage 1 is 500 epochs for PAC-Bayes training. Moreover, a learning rate scheduler was added to both our method and the baseline to make the training fully converge. Specifically, the learning rate will be reduced by 0.1 whenever the training accuracy does not increase for 20 epochs. For PAC-Bayes training, the scheduler is only activated in Stage 2. The training will be terminated when the training accuracy is above 99.9% for 20 epochs or when the learning rate decreases to below $1e{-}5$. We also add label smoothing (0.1) (Szegedy et al., 2016) to neural networks when comparing SGD/Adam with our method on image classification tasks to enhance the final test accuracy for all training methods.

The detailed searched values of hyperparameters include momentum for SGD $(0.3, 0.6, 0.9)$, learning rates $(1e{-}3, 5e{-}3, 1e{-}2, 5e{-}2, 1e{-}1, 2e{-}1)$, weight decay $(1e{-}4, 5e{-}4, 1e{-}3, 5e{-}3, 1e{-}2)$, and noise injection $(5e{-}4, 1e{-}3, 5e{-}3, 1e{-}2)$. The best learning rate for Adam and AdamW is the same since weight decay is the only difference between the two optimizers. We adjusted one hyper-parameter at a time while keeping the others fixed to accelerate the search. To determine the optimal hyper-parameter for a variable, we compared the mean test accuracy of the last five epochs. We then used this selected hyper-parameter to tune the next one. We used an extensive grid search as a baseline to ensure the best achievable test accuracy in the literature (Table 4 of Geiping et al. (2021)). The noise injection is only applied to Adam/AdamW, as it sometimes causes instability to SGD and does not seem to increase the test performance. We compared the mean test accuracy of the last five epochs to determine each optimal hyper-parameter. The test accuracy from all experiments with batch size 128 with the learning rate $1e{-}4$ is shown in Figure A.1 and Figure A.2.

To best demonstrate the sensitivity of the hyper-parameter selection of baselines and motivate our PAC-Bayes training, we organized the test accuracy below for ResNet18. Considering the search efficiency, we searched the hyperparameter one by one. For SGD, we first searched the learning rate, set the momentum and the weight decay as 0 (both are default values for SGD), and then used

the best learning rate to search for the momentum. At last, the best-searched learning rate and momentum are used to search for weight decay. For Adam, we searched the learning rate, weight decay, and noise injection in an order similar to SGD. Since AdamW and Adam are the same when setting the weight decay as 0, we searched for the best weight decay based on the best learning rate obtained from searching on Adam.

A.3.4 Compatibility with Data Augmentation

We didn't include data augmentation in the experiments in the main text. Because with data augmentation, there is no rigorous way of choosing the sample size m that appears in the PAC-Bayes bound. More specifically, for the PAC-Bayes bound to be valid, the training data has to be i.i.d. samples from some underlying distribution. However, most data augmentation techniques would break the i.i.d. assumption. As a result, if we have 10 times more samples after augmentation, the new information they bring in would be much less than those from 10 times i.i.d. samples. In this case, how to determine the effective sample size m to be used in the PAC-Bayes bound is a problem.

Since knowing whether a training method can work well with data augmentation is important, we carried out the PAC-Bayes training with an ad-hoc choice of m, that is, we set m to be the size of the augmented data. We compared the grid-search result of SGD and Adam versus PAC-Bayes training on CIFAR10 with ResNet18. The augmentation is achieved by random flipping and random cropping. The data augmentation increased the size of the training sample by 128 times. The test accuracy for SGD is 95.2%, it is 94.3% for Adam, it is 94.4% for AdamW, and it is 94.3% for PAC-Bayes training with the layerwise prior. In contrast, the test accuracy without data augmentation is lower than 90% for all methods. It suggests that data augmentation does not conflict with the PAC-Bayes training in practice.

A.3.5 Model analysis

We examined the learning process of PAC-Bayes training by analyzing the posterior variance $\boldsymbol{\sigma}$ for different layers in models trained by Algorithm A.2. Typically, batch norm layers have smaller $\boldsymbol{\sigma}$ values than convolution layers. Additionally, shadow convolution and the last few layers have smaller $\boldsymbol{\sigma}$ values than the middle layers. We also found that skip-connections in ResNet18 have

smaller $\boldsymbol{\sigma}$ values than nearby layers, suggesting that important layers with a greater impact on the output have smaller $\boldsymbol{\sigma}$ values.

In Stage 1, the training loss is higher than the testing loss, which means the adopted PAC-Bayes bound is able to bound the generalization error throughout the PAC-Bayes training stage. Additionally, we observed that the final value of K is usually very close to the minimum of the sampled function values. The average value of $\boldsymbol{\sigma}$ experienced a rapid update during the initial 50 warmup epochs but later progressed slowly until Stage 2. The details can be found in Figure A.9 and A.13. Based on the figures, shadow convolution, and the last few layers have smaller $\boldsymbol{\sigma}$ values than the middle layers for all models. We also found that skip-connections in ResNet18 and ResNet34 have smaller $\boldsymbol{\sigma}$ values than nearby layers on both datasets, suggesting that important layers with a greater impact on the output have smaller $\boldsymbol{\sigma}$ values.

Computational cost: In PAC-Bayes training, we have four parameters $\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\sigma}, \gamma$. Among these variables, γ can be computed on the fly or whenever needed, so there is no need to store them. We need to store $\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\sigma}$, where $\boldsymbol{\sigma}$ has the same size as $\boldsymbol{\theta}$ and the size of $\boldsymbol{\lambda}$ is the same as the number of layers which is much smaller. Hence the total storage is approximately doubled. Likewise, when computing the gradient for $\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\sigma}$, the cost of automatic differentiation in each iteration is also approximately doubled. In the inference stage, the complexity is the same as in conventional training.

Effect of two stages: We have tested the effect of the two stages. Without the first stage, the algorithm cannot automatically learn the noise level and weight decay to be used in the second stage. If the first stage is there but too short (10 epochs for example), then the final performance of VGG13 on CIFAR100 will reduce to 64.0% . Without Stage 2, the final performance is not as good as reported either. The test accuracy of models like VGG13 and ResNet18 on CIFAR10 would be 10% lower as in Figure A.9 and A.13.

A.3.6 Node classification by GNNs

We test the PAC-Bayes training algorithm on the following popular GNN models, tuning the learning rate $(1e{-}3, 5e{-}3, 1e{-}2)$, weight decay $(0, 1e{-}4, 1e{-}3, 1e{-}2)$, noise injection

$(0, 1e{-}3, 5e{-}3, 1e{-}2)$, and dropout $(0, 0.4, 0.8)$. The number of filters per layer is 32 in GCN (Kipf and Welling, 2016) and SAGE (Hamilton et al., 2017). For GAT (Veličković et al., 2018), the number of filters is 8 per layer, the number of heads is 8, and the dropout rate of the attention coefficient is 0.6. Fpr APPNP (Gasteiger et al., 2018), the number of filters is 32, $K = 10$ and $\alpha = 0.1$. We set the number of layers to 2, achieving the best baseline performance. A ReLU activation and a dropout layer are added between the convolution layers for baseline training only. Since GNNs are faster to train than convolutional neural networks, we tested all possible combinations of the above parameters for the baseline, conducting 144 searches per model on one dataset. We use Adam as the optimizer with the learning rate as $1e{-}2$ for all models using both training and validation nodes for PAC-Bayes training.

We also did a separate experiment using both training and validation nodes for training. For baselines, we need first to train the model to detect the best hyperparameters as before and then train the model again on the combined data. Our PAC-Bayes training can also match the best generalization of baselines in this setting.

All results are visualized in Figure A.5-A.8. The AdamW+val and scalar+val record the performances of the baseline and the PAC-Bayes training, respectively, with both training and validation datasets for training. We can see that test accuracy after adding validation nodes increased significantly for both methods but still, the results of our algorithm match the best test accuracy of baselines. Our proposed PAC-Bayes training with the scalar prior is better than most of the settings during searching and achieved comparable test accuracy when adding validation nodes to training.

A.3.7 Few-shot text classification with transformers

The proposed method is also observed to work on transformer networks. We conducted experiments on two text classification tasks of the GLUE benchmark as shown in Table A.1. SST is the sentiment analysis task, whose performance is evaluated as the classification accuracy. Sentiment analysis is the process of analyzing the sentiment of a given text to determine if the emotional tone of the text is positive, negative, or neutral. QNLI (Question-answering Natural Language Inference) focuses on determining the logical relationship between a given question and a corresponding

sentence. The objective of QNLI is to determine whether the sentence contradicts, entails, or is neutral with respect to the question.

We use classification accuracy as the evaluation metric. The baseline method uses grid search over the hyper-parameter choices of the learning rate ($1e{-}1$, $1e{-}2$, $1e{-}3$), batch size $(2, 8, 16, 32, 80)$, dropout ratio $(0, 0.5)$, optimization algorithms (SGD, AdamW), noise injection $(0, 1e{-}5, 1e{-}4, 1e{-}3, 1e{-}2, 1e{-}1)$, and weight decay $(0, 1e{-}1, 1e{-}2, 1e{-}3, 1e{-}4)$. The learning rate and batch size of our method are set to $1e{-}3$ and 100 (i.e., full-batch), respectively. In this task, the number of training samples is small (80). As a result, the preset $\gamma_2 = 10$ is a bit large and thus prevents the model from achieving the best performance with PAC-Bayes training.

We adopt BERT (Devlin et al., 2018) as our backbone and added one fully connected layer as the classification layer. Only the added classification layer is trainable, and the pre-trained model is frozen without gradient update. To simulate a few-shot learning scenario, we randomly sample 100 instances from the original training set and take the whole development set to evaluate the classification performance. We split the training set into 5 splits, taking one split as the validation data and the rest as the training set. Each experiment was conducted five times, and we report the average performance. We used the PAC-Bayes training with the scalar prior in this experiment. According to Table A.1, our method is competitive to the baseline method on the SST task, and the performance gap is only 0.4 points. On the QNLI task, our method outperforms the baseline by a large margin, and the variance of our proposed method is less than that of the baseline method.

Table A.1 Test accuracy on the development sets of 2 GLUE benchmarks.

	SST	**QNLI**
baseline	**72.9±0.99**	62.6±0.10
scalar	72.5±0.99	**64.2±0.02**

A.3.8 Additional experiments stability

We conducted extra experiments to showcase the robustness of the proposed PAC-Bayes training algorithm. Specifically, we tested the effect of different learning rates on ResNet18 and VGG13

models trained with layerwise prior. Learning rate has long been known as an important impact factor of the generalization for baseline training. Within the stability range of gradient descent, the larger the learning rate is, the better the generalization has been observed (Lewkowycz et al., 2020). In contrast, the generalization of the PAC-Bayes trained model is less sensitive to the learning rate. We do observe that due to the newly introduced noise parameters, the stability of the optimization gets worse, which in turn requires a lower learning rate to achieve stable training. But as long as the stability is guaranteed by setting the learning rate low enough, our results, as Table A.2, indicated that the test accuracy remained stable across various learning rates for VGG13 and Resnet18. The dash in the table means that the learning rate for that particular setting is too large to maintain the training stability. For learning rates below $1e{-}4$, we trained the model in Stage 1 for more epochs (700) to fully update the prior and posterior variance.

We also demonstrate that the warmup iterations (as discussed at the beginning of this section) do not affect generalization. As shown in Table A.4, the test accuracy is insensitive to different numbers of warmup iterations. Furthermore, additional evaluations of the effects of batch size (Table A.5), optimizer (Tables A.6), and γ_1 and γ_2 (Table A.7)

We further visualize the sorted test accuracy of baselines and our proposed PAC-Bayes training with large batch sizes and a fixed learning rate $5e{-}4$ in Figure A.3 and Figure A.4. These figures demonstrate that our PAC-Bayes training algorithm achieves better test accuracy than most searched settings. For models VGG13 and ResNet18, the large batch size is 2048, and for large models VGG19 and ResNet34, the large batch size is set to 1280 due to the GPU memory limitation.

Table A.2 Test accuracy of ResNet18 trained with different learning rates.

lr	$3e{-}5$	$5e{-}5$	$1e{-}4$	$2e{-}4$	$3e{-}4$	$5e{-}4$
CIFAR10	88.4	88.8	89.3	88.6	88.3	89.2
CIFAR100	69.2	69.0	68.9	69.1	69.1	69.6

Table A.3 Test accuracy of VGG13 trained with different learning rates.

lr	$3e{-5}$	$5e{-5}$	$1e{-4}$	$2e{-4}$	$3e{-4}$	$5e{-4}$
CIFAR10	88.6	88.9	89.7	89.6	89.6	89.5
CIFAR100	67.7	68.0	67.1	-	-	-

Table A.4 Test accuracy of ResNet18 trained with warmup epochs of σ.

	10	20	50	80	100	150
CIFAR10	88.5	88.5	89.3	89.5	89.5	88.9
CIFAR100	69.4	69.6	68.9	69.1	69.0	68.1

Table A.5 Test accuracy of VGG13 with different batch sizes.

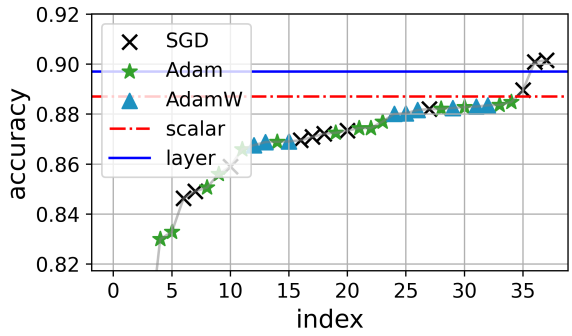
Batch Size	128	256	1024	2048	2500
Test Acc	89.7	89.7	88.7	89.4	88.3

Table A.6 Test accuracy of ResNet18 using SGD: Effects of different momentum values (with learning rate 1×10^{-3}) and different learning rates (with momentum 0.9).

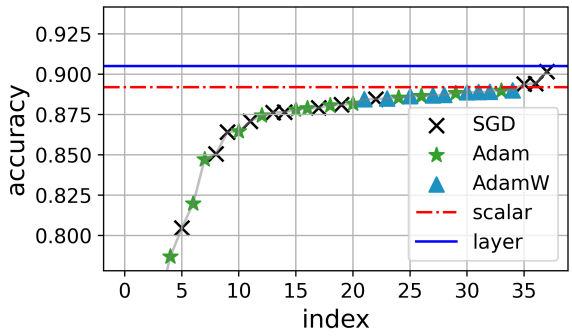
	Momentum			Learning Rate		
	0.3	0.6	0.9	1×10^{-4}	3×10^{-4}	1×10^{-3}
Test Acc	88.6	88.8	89.2	88.3	88.8	89.2

Table A.7 Test accuracy of ResNet18 with different settings for γ_1 (with $\gamma_2 = 20$) and γ_2 (with $\gamma_1 = 0.1$).

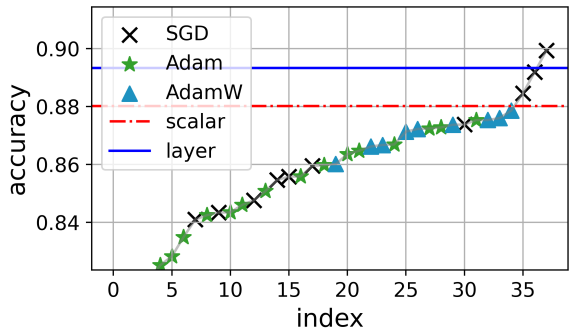
	γ_1			γ_2		
	0.1	0.5	1.0	10	15	20
Test Acc	88.8	89.3	88.8	89.3	89.4	89.4



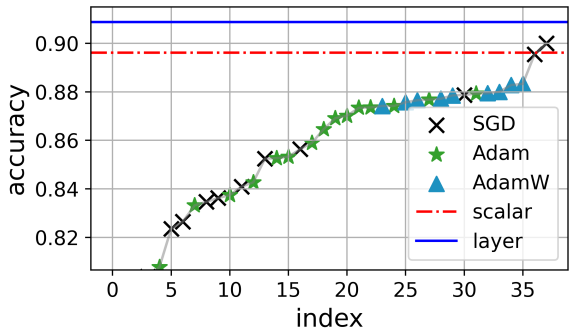
(a) VGG13



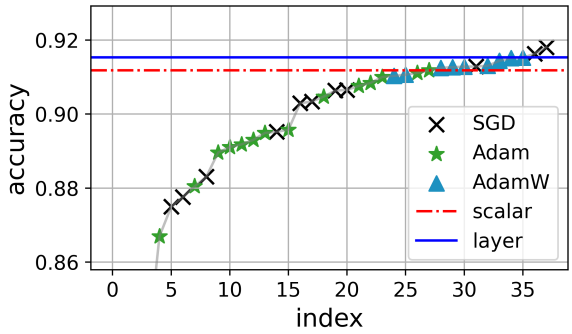
(b) VGG19



(c) ResNet18

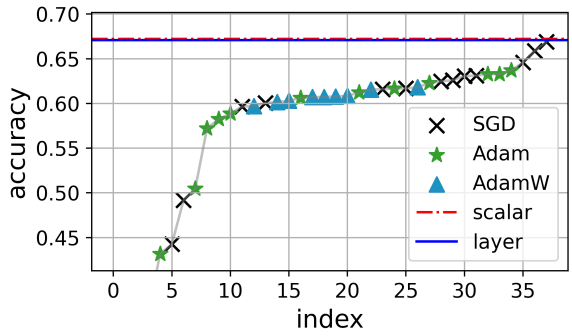


(d) ResNet34

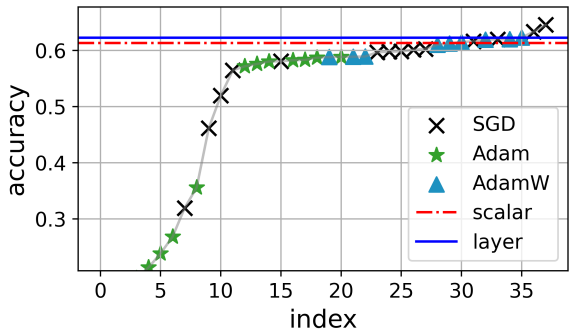


(e) Desnse121

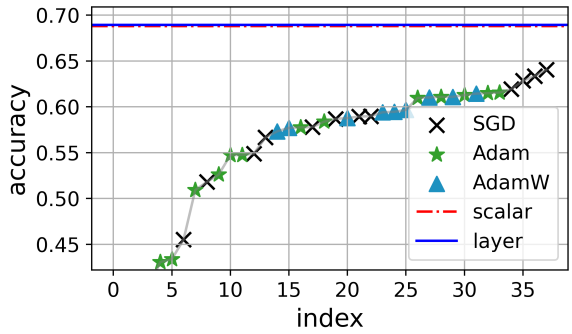
Figure A.1 Sorted test accuracy of CIFAR10. The x-axis represents the experiment index.



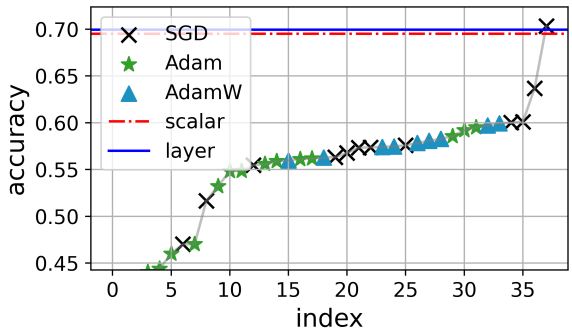
(a) VGG13



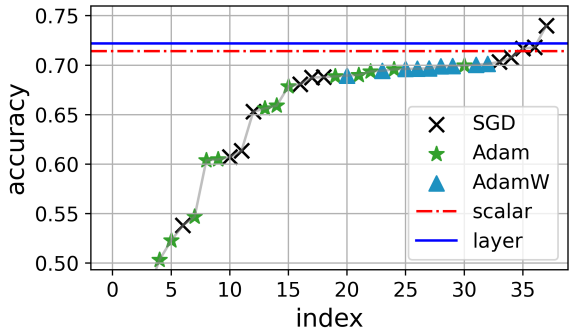
(b) VGG19



(c) ResNet18

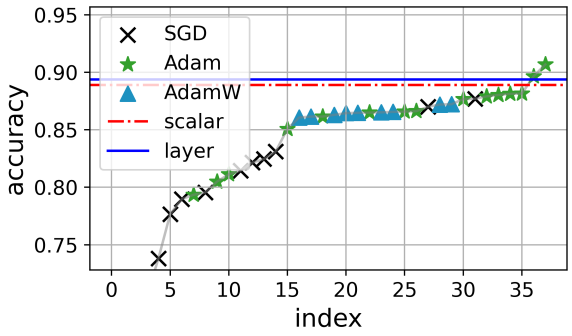


(d) ResNet34

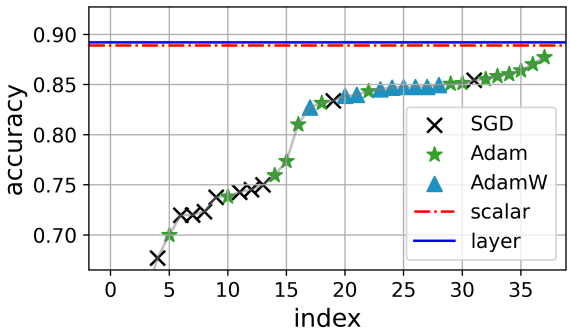


(e) Desnse121

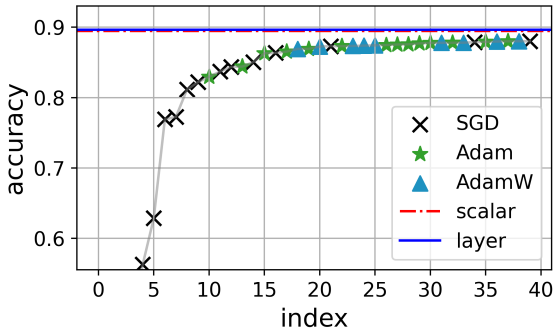
Figure A.2 Sorted test accuracy of CIFAR100. The x-axis represents the experiment index.



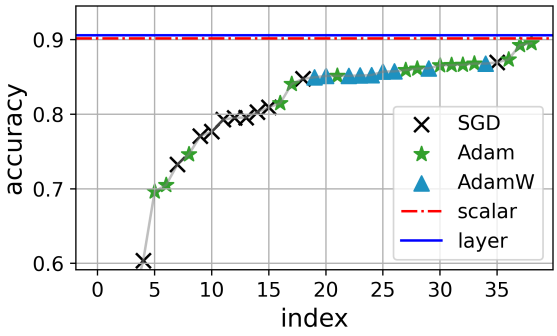
(a) VGG13 (batch: 2048)



(b) ResNet18 (batch: 2048)

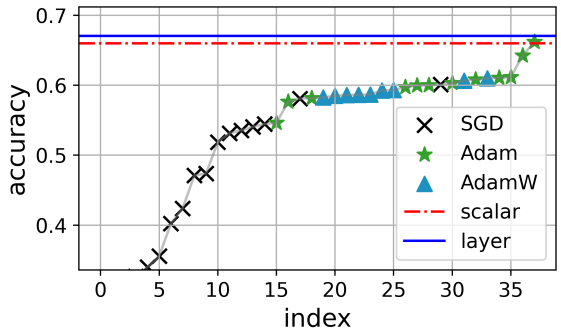


(c) VGG19 (batch: 1280)

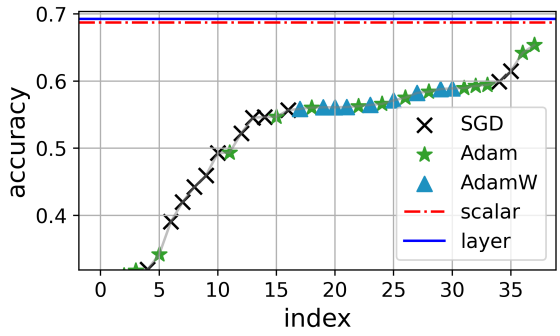


(d) ResNet34 (batch: 1280)

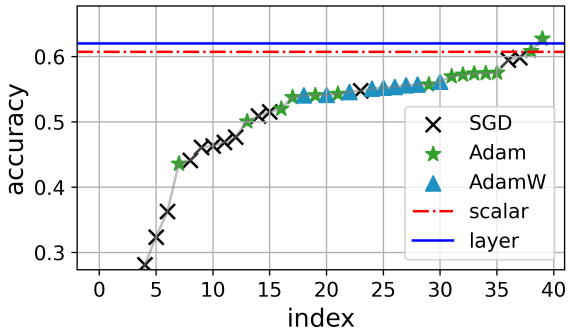
Figure A.3 Sorted test accuracy of CIFAR10 with large batch sizes. The x-axis represents the experiment index.



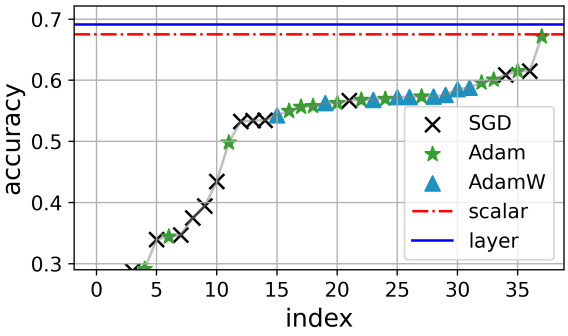
(a) VGG13 (batch: 2048)



(b) ResNet18 (batch: 2048)

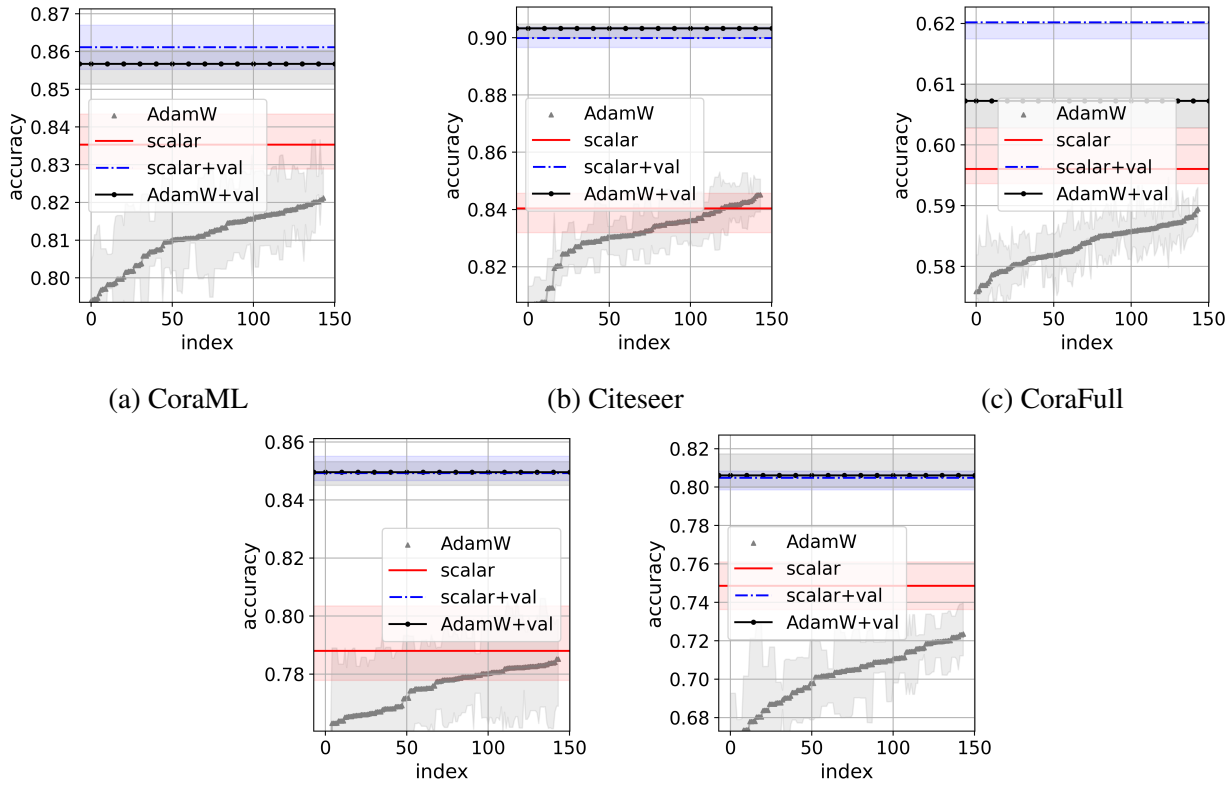


(c) VGG19 (batch: 1280)



(d) ResNet34 (batch: 1280)

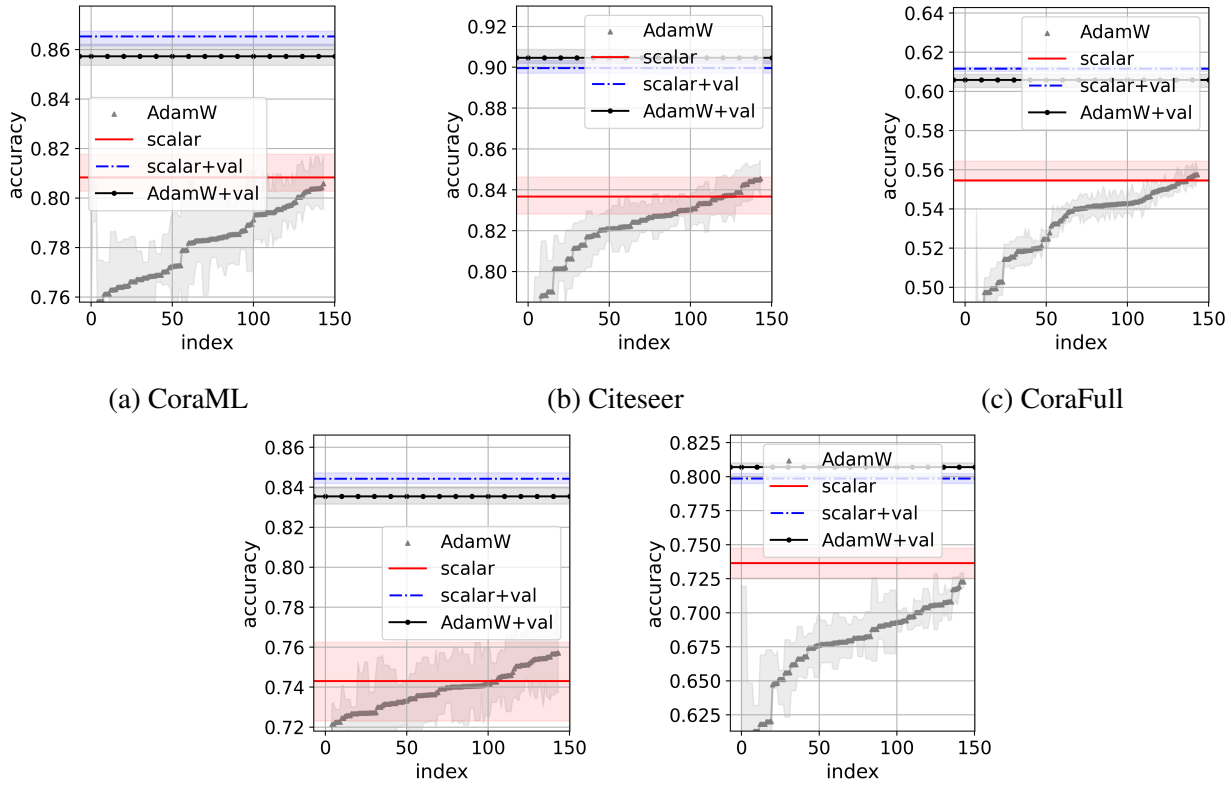
Figure A.4 Sorted test accuracy of CIFAR100 with large batch sizes. The x-axis represents the experiment index.



(a) CoraML (b) Citeseer (c) CoraFull

(d) DBLP (e) DBLP

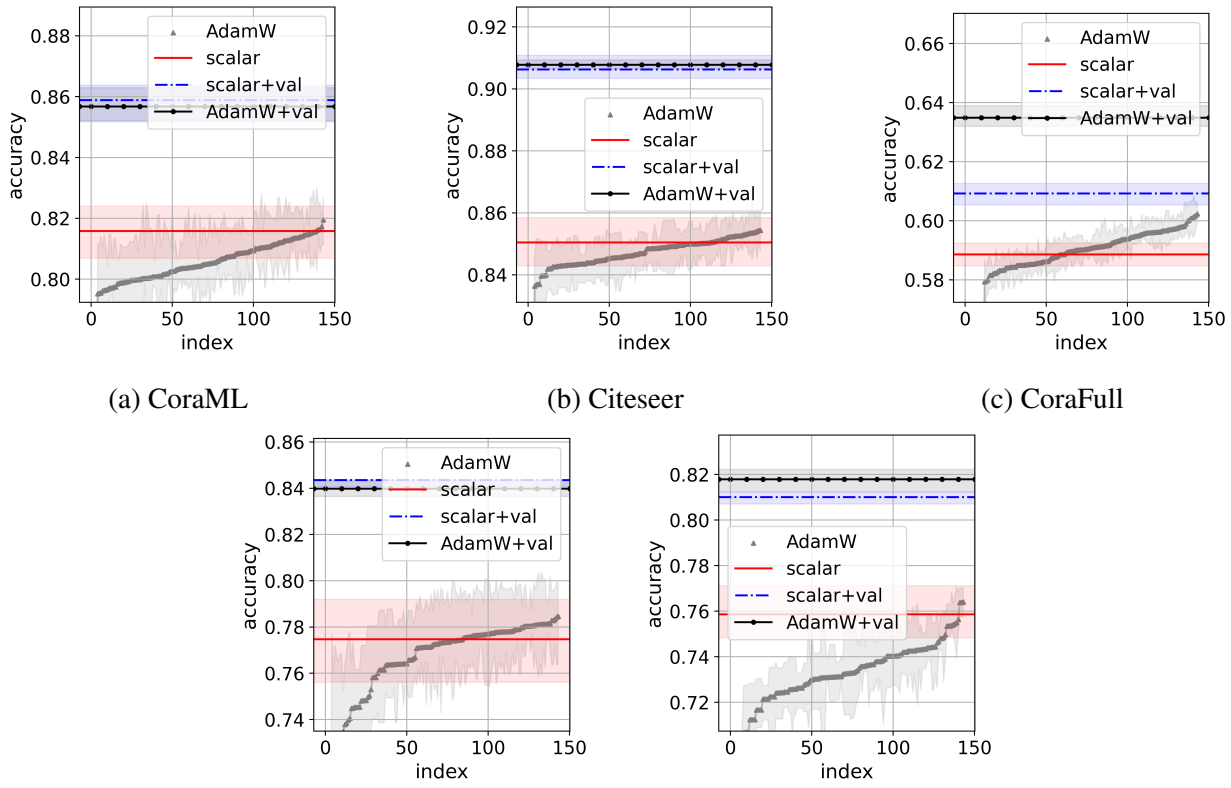
Figure A.5 Test accuracy of GCN. The first and third quartiles construct the interval over the ten random splits. {+val} denotes the performance with both training and validation datasets for training.



(a) CoraML (b) Citeseer (c) CoraFull

(d) DBLP (e) DBLP

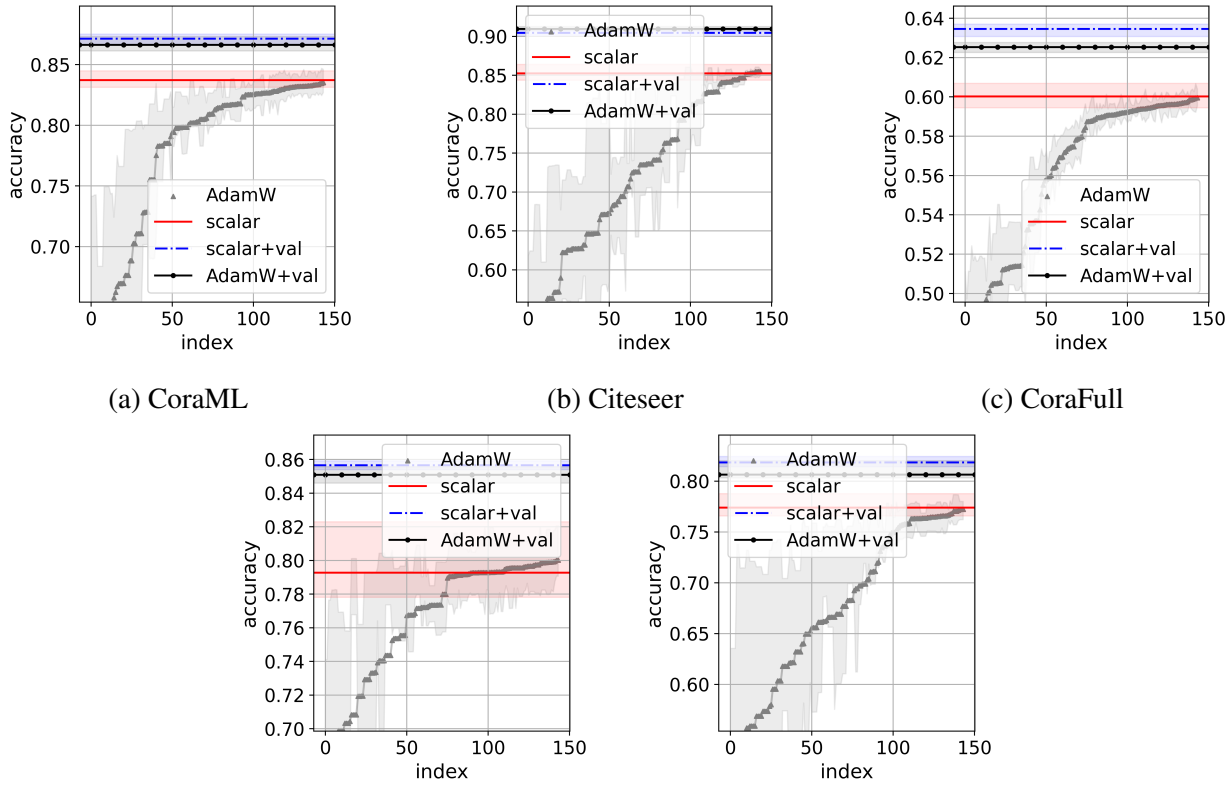
Figure A.6 Test accuracy of SAGE. The first and third quartiles construct the interval over the ten random splits. {+val} denotes the performance with both training and validation datasets for training.



(a) CoraML (b) Citeseer (c) CoraFull

(d) DBLP (e) DBLP

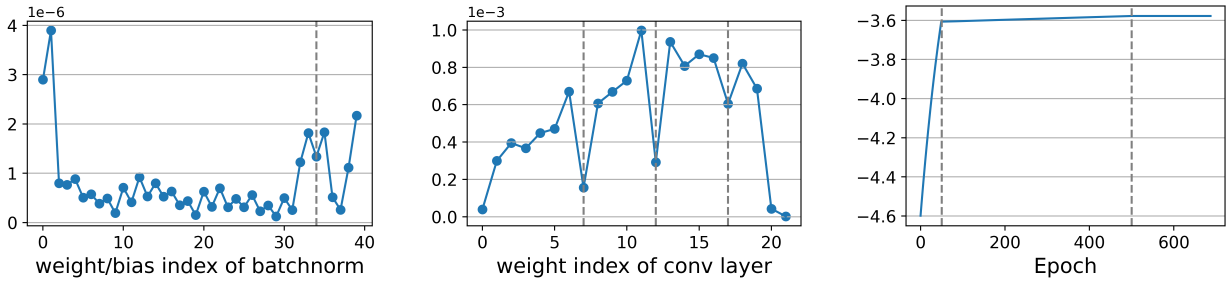
Figure A.7 Test accuracy of GAT. The first and third quartiles construct the interval over the ten random splits. {+val} denotes the performance with both training and validation datasets for training.



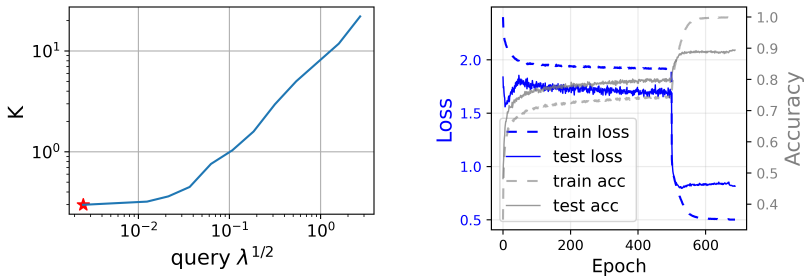
(a) CoraML (b) Citeseer (c) CoraFull

(d) DBLP (e) DBLP

Figure A.8 Test accuracy of APPNP. The first and third quartiles construct the interval over the ten random splits. {+val} denotes the performance with both training and validation datasets for training.

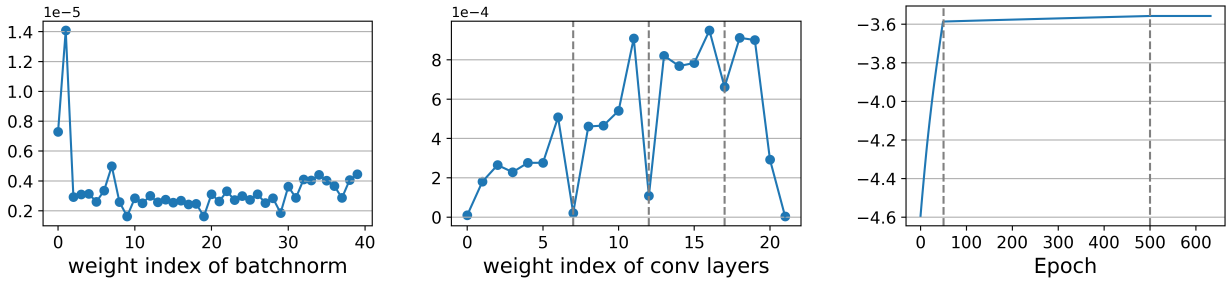


(a) mean($\boldsymbol{\sigma}$) of batch-norm layers. (b) mean($\boldsymbol{\sigma}$) of convolution layers. (c) mean($\boldsymbol{\sigma}$) in training.

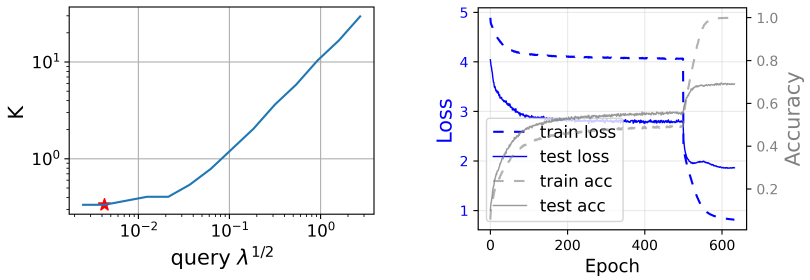


(d) Function $\tilde{K}(\bar{\boldsymbol{\lambda}})$. (e) Training and testing process.

Figure A.9 Training details of ResNet18 on CIFAR10. The red star denotes the final K.

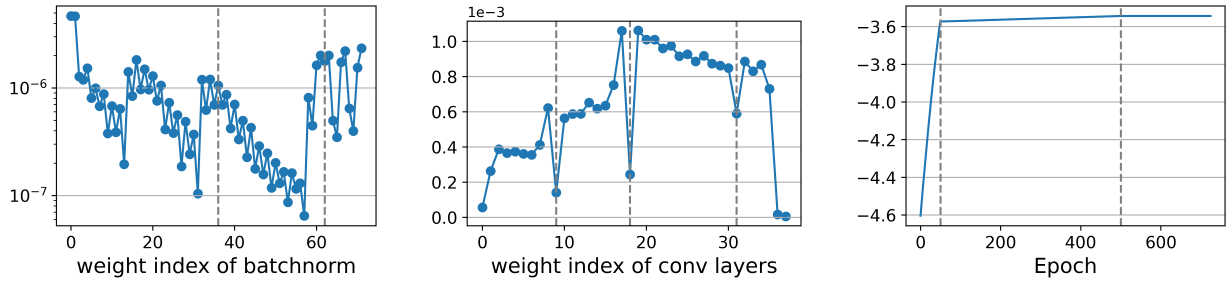


(a) mean($\boldsymbol{\sigma}$) of batch-norm layers. (b) mean($\boldsymbol{\sigma}$) of convolution layers. (c) mean($\boldsymbol{\sigma}$) in training.

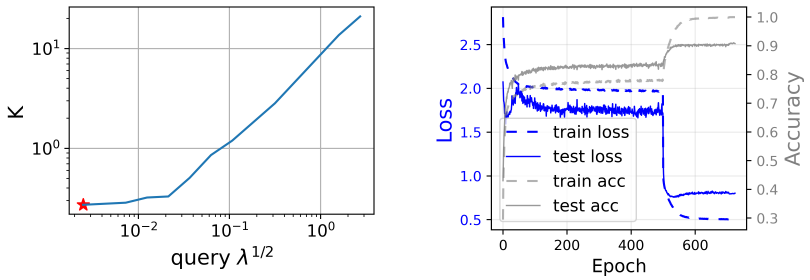


(d) Function $\tilde{K}(\bar{\boldsymbol{\lambda}})$. (e) Training and testing process.

Figure A.10 Training details of ResNet18 on CIFAR100. The red star denotes the final K.

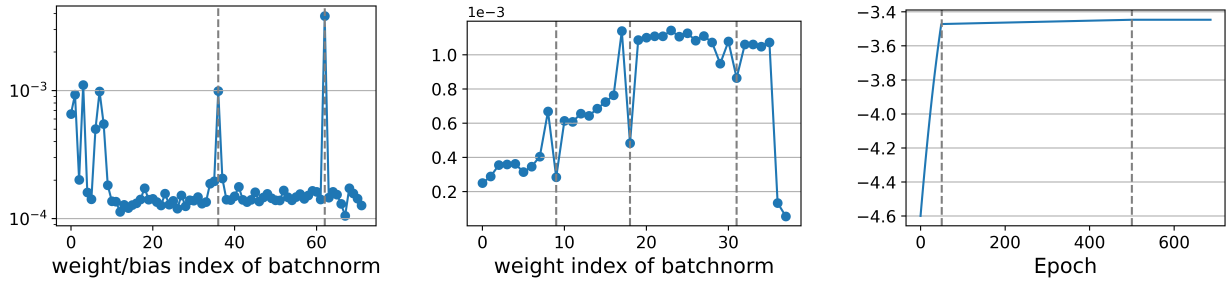


(a) mean($\boldsymbol{\sigma}$) of batch-norm layers. (b) mean($\boldsymbol{\sigma}$) of convolution layers. (c) mean($\boldsymbol{\sigma}$) in training.

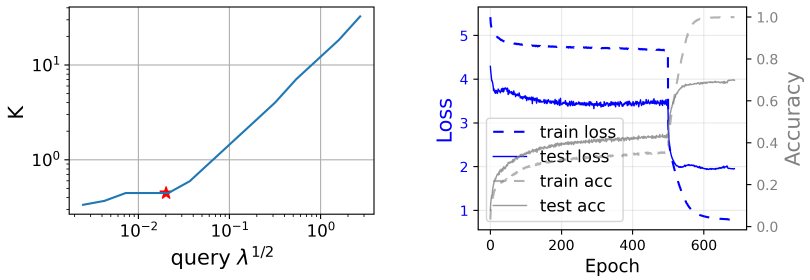


(d) Function $\tilde{K}(\bar{\boldsymbol{\lambda}})$. (e) Training and testing process.

Figure A.11 Training details of ResNet34 on CIFAR10. The red star denotes the final K.

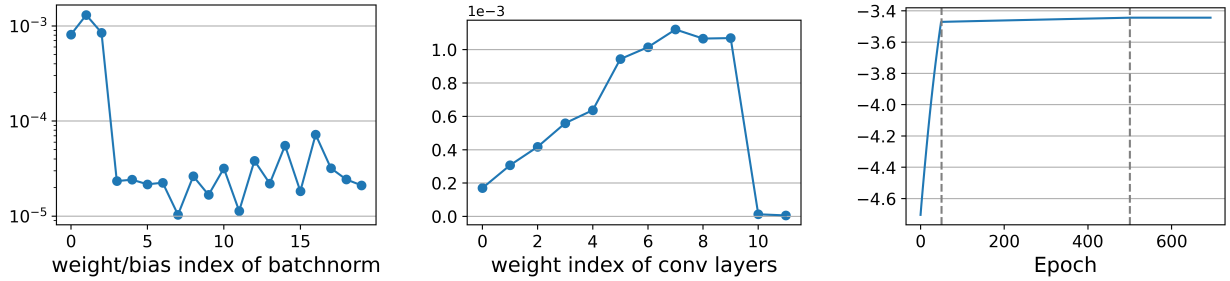


(a) mean($\boldsymbol{\sigma}$) of batch-norm layers. (b) mean($\boldsymbol{\sigma}$) of convolution layers. (c) mean($\boldsymbol{\sigma}$) in training.



(d) Function $\tilde{K}(\bar{\boldsymbol{\lambda}})$. (e) Training and testing process.

Figure A.12 Training details of ResNet34 on CIFAR100. The red star denotes the final K.



(a) mean($\boldsymbol{\sigma}$) of batch-norm layers. (b) mean($\boldsymbol{\sigma}$) of convolution layers. (c) mean($\boldsymbol{\sigma}$) in training.

(d) Function $\tilde{K}(\bar{\boldsymbol{\lambda}})$. (e) Training and testing process.

Figure A.13 Training details of VGG13 on CIFAR10. The red star denotes the final K.

APPENDIX B

MAGNET: A NEURAL NETWORK FOR DIRECTED GRAPHS

B.1 List of method abbreviations

- MagNet (this paper)
- ChebNet Defferrard et al. (2016)
- GCN Kipf and Welling (2016)
- APPNP Klicpera et al. (2019a)
- GAT Veličković et al. (2018)
- SAGE Hamilton et al. (2017)
- GIN Xu et al. (2018)
- DGCN Tong et al. (2020b)
- DiGraph Tong et al. (2020a)
- DiGraphIB Tong et al. (2020a): DiGraph with inception blocks
- BiGCN: applying GCN on the original adjacency matrix and its transpose matrix separately
- BiSAGE: applying SAGE on the original adjacency matrix and its transpose matrix separately
- BiGAT: applying GAT on the original adjacency matrix and its transpose matrix separately
- KNN: K-nearest neighbors based on the eigenvectors with the smallest eigenvalues of magnetic Laplacian Fanuel et al. (2017).

B.2 Further implementation details

We set the parameter $K = 1$ in our implementation of both ChebNet and MagNet, except for synthetic noisy cylcic graphs with random input features. For sythetic noisy cylcic graphs with random input features, we also tried $K = 2$ for MagNet. We train all models with a maximum of 3000 epochs and stop early if the validation error doesn't decrease after 500 epochs for both node classification and link prediction tasks. One dropout layer with a probability of 0.5 is created before the last linear layer. The model is picked with the best validation accuracy during training for testing. We tune the number of filters in $[16, 32, 48]$ for the graph convolutional layers for all models, except DigraphIB, since the inception block has more trainable parameters. For node classification, we tune the learning rate in $[1e^{-3}, 5e^{-3}, 1e^{-2}]$ for all models. Compared with node classification, the

number of available samples for link prediction is much larger. Thus, we set a relatively small learning rate of $1e^{-3}$.

We use Adam as the optimizer and ℓ_2 regularization with the hyperparameter set as $5e^{-4}$ to avoid overfitting. We post the best testing performance by grid-searching based on validation accuracy. For node classification on the synthetic datasets, we generate a one-dimensional node feature sampled from the standard normal distribution. We use the original features for the other node classification datasets. For link prediction, we use the in-degree and out-degree as the node features for all datasets instead the original features. This allows all models to learn directed information from the adjacency matrix. Our experiments were conducted on 8 compute nodes each with 1 Nvidia Tesla V100 GPU, 120G RAM, and 32 Intel Xeon E5-2660 v3 CPUs; as well as on a compute node with 8 Nvidia RTX 8000 GPUs, 1000GB RAM, and 48 Intel Xeon Silver 4116 CPUs.

Here are implementation details specific to certain methods:

- We set the parameter ϵ to 0 in GIN for both tasks.
- For GAT and BiGAT, the number of heads tuned is in $[2, 4, 8]$.
- For APPNP, we set $K = 10$ for node classification (following the original paper Klicpera et al. (2019a)), and search K in $[1, 5, 10]$ for link prediction.
- The coefficient α for PageRank-based models (APPNP, DiGraph) is searched in $[0.05, 0.1, 0.15, 0.2]$.
- For DiGraph, the model includes graph convolutional layers without the high-order approximation and inception module. The high order Laplacian and the inception module is included in DigraphIB.
- DigraphIB is a bit different than other networks because it requires generating a three-channel Laplacian tensor. For this network, the number of filters for each channel is searched in $[6, 11, 21]$ for node classification and link prediction.
- For GCN, the out-degree normalized, directed adjacency matrix, including self-loops is also tried in addition to the symmetrized adjacency matrix for node classification tasks, except for synthetic datasets since symmetrization will break the cluster pattern.

- For other spatial methods, including APPNP, GAT, SAGE, and GIN, we tried both the symmetrized adjacency matrices and the original directed adjacency matrices for node classification tasks except for synthetic datasets.
- For KNN, we set $q = 0.25$ and $K = 5$.

B.3 Datasets

B.4 Node classification

We use six real datasets for node classification. A directed edge is defined as follows. If the edge $(u, v) \in E$ but $(v, u) \notin E$, then (u, v) is a directed edge. If $(u, v) \in E$ and $(v, u) \in E$, then (u, v) and (v, u) are undirected edges (in other words, undirected edges that are not self-loops are counted twice). For the citation datasets, *Cora-ML* and *Citeseer*, we randomly select 20 nodes in each class for training, 500 nodes for validation, and the rest for testing following Tong et al. (2020a). For the synthetic datasets (*ordered DSBM graphs*, *cyclic DSBM graphs*, *noisy cyclic DSBM graphs*), we generate a one-dimensional node feature sampled from the standard normal distribution.

Ten folds are generated randomly for each dataset, except for *Cornell*, *Texas* and *Wisconsin*. For *Cornell*, *Texas*, and *Wisconsin*, we use the same training, validation, and testing folds as Pei et al. (2020). For *Telegram*, we treat it as a directed, unweighted graph and randomly generate 10 splits for training/validation/testing with 60%/20%/20% of the nodes. The node features are sampled from the normal distribution.

B.5 Link prediction

We use eight real datasets in link prediction. Instead of using the original features, we use the in-degree and out-degree as the node features in order to allow the models to learn structural information from the adjacency matrix directly. The connectivity is maintained by getting the undirected minimum spanning tree before removing edges for validation and testing. For the results in the main text, undirected edges and, if they exist, pairs of vertices with multiple edges between them, may be placed in the training/validation/testing sets. However, labels that indicate the direction of such edges are not well defined, and therefore can be considered as noisy labels from the machine

learning perspective. In order to obtain a full set of well-defined, noiseless labels, in the supplement we also run experiments in which undirected edges and pairs of vertices with multiple edges between them are ignored when sampling edges for training/validation/testing (in other words, only directed edges, and the absence of an edge, are included). We evaluated all models on four prediction tasks, which we now describe.

To construct the datasets that we use for training, validation and testing, which consist of pairs of vertices in the graph, we do the following. (1) Existence prediction. If $(u, v) \in E$, we give (u, v) the label 0, otherwise its label is 1. The proportion of the two classes of edges is 25% and 75%, respectively, when undirected edges and multi-edges are included, and 50% and 50%, respectively, when only directed edges are included. (2) Direction prediction. Given an ordered node pair (u, v), we give the label 0 if $(u, v) \in E$ and the label 1 if $(v, u) \in E$, conditioning on $(u, v) \in E$ or $(v, u) \in E$. The proportion of the two types of edges is 50% and 50%. We randomly generated ten folds for all datasets. We used 15% and 5% of edges for testing and validation for all datasets.

B.6 Eigenvalues of the magnetic Laplacian

In this section we state and prove three theorems. Theorem B.6.1, which shows that both the normalized and unnormalized magnetic Laplacian a postive semidefinite, is well known (see e.g. Fanuel et al. (2018)). Theorem B.6.2, which shows that the eigenvalues of the normalized magnetic Laplacian lie in the interval $[0, 2]$, is a straightforward adaption of the corresponding result for the traditional normalized graph Laplacian. Finally, Theorem B.6.4 proves the un-normalized magnetic Laplacian may be factored in terms of a complex valued incidence matrix, analogous to the well-known result for the standard graph Laplacian. We give full proofs of all three results for completeness.

Theorem B.6.1. *Let $G = (V, E)$ be a directed graph where V is a set of N vertices and $E \subseteq V \times V$ is a set of directed edges. Then, for all $q \geq 0$, both the unnormalized magnetic Laplacian $\mathbf{L}_U^{(q)}$ and its normalized counterpart $\mathbf{L}_N^{(q)}$ are positive semidefinite.*

Proof. Let $\mathbf{x} \in \mathbb{C}^N$. We first note that since $\mathbf{L}_U^{(q)}$ is Hermitian we have $\text{Imag}(\mathbf{x}^\dagger \mathbf{L}_U^{(q)} \mathbf{x}) = 0$. Next,

we use the definition of $\mathbf{D}_s$ and the fact that $\mathbf{A}_s$ is symmetric to observe that

$$
\begin{aligned}
&2\text{Real}\left(\mathbf{x}^\dagger \mathbf{L}_U^{(q)} \mathbf{x}\right) \\
=&2\sum_{u,v=1}^{N} \mathbf{D}_s(u,v)\mathbf{x}(u)\overline{\mathbf{x}(v)} - 2\sum_{u,v=1}^{N} \mathbf{A}_s(u,v)\mathbf{x}(u)\overline{\mathbf{x}(v)}\cos(i\mathbf{\Theta}^{(q)}(u,v)) \\
=&2\sum_{u=1}^{N} \mathbf{D}_s(u,u)\mathbf{x}(u)\overline{\mathbf{x}(u)} - 2\sum_{u,v=1}^{N} \mathbf{A}_s(u,v)\mathbf{x}(u)\overline{\mathbf{x}(v)}\cos(i\mathbf{\Theta}^{(q)}(u,v)) \\
=&2\sum_{u,v=1}^{N} \mathbf{A}_s(u,v)|\mathbf{x}(u)|^2 - 2\sum_{u,v=1}^{N} \mathbf{A}_s(u,v)\mathbf{x}(u)\overline{\mathbf{x}(v)}\cos(i\mathbf{\Theta}^{(q)}(u,v)) \\
=&\sum_{u,v=1}^{N} \mathbf{A}_s(u,v)|\mathbf{x}(u)|^2 + \sum_{u,v=1}^{N} \mathbf{A}_s(v,u)|\mathbf{x}(v)|^2 - 2\sum_{u,v=1}^{N} \mathbf{A}_s(u,v)\mathbf{x}(u)\overline{\mathbf{x}(v)}\cos(i\mathbf{\Theta}^{(q)}(u,v)) \\
=&\sum_{u,v=1}^{N} \mathbf{A}_s(u,v)\left(|\mathbf{x}(u)|^2 + |\mathbf{x}(v)|^2 - 2\mathbf{x}(u)\overline{\mathbf{x}(v)}\cos(i\mathbf{\Theta}^{(q)}(u,v))\right) \qquad \text{(B.1)} \\
\geq&\sum_{u,v=1}^{N} \mathbf{A}_s(u,v)\left(|\mathbf{x}(u)|^2 + |\mathbf{x}(v)|^2 - 2|\mathbf{x}(u)||\mathbf{x}(v)|\right) \\
=&\sum_{u,v=1}^{N} \mathbf{A}_s(u,v)(|\mathbf{x}(u)| - |\mathbf{x}(v)|)^2 \\
\geq&0.
\end{aligned}
$$

Thus, $\mathbf{L}_U^{(q)}$ is positive semidefinite. For the normalized magnetic Laplacian, we note that

$$\left(\mathbf{D}_s^{-1/2}\mathbf{A}_s\mathbf{D}_s^{-1/2}\right) \odot \exp(i\mathbf{\Theta}^{(q)}) = \mathbf{D}_s^{-1/2}\left(\mathbf{A}_s \odot \exp(i\mathbf{\Theta}^{(q)})\right)\mathbf{D}_s^{-1/2},$$

and therefore

$$\mathbf{L}_N^{(q)} = \mathbf{D}_s^{-1/2}\mathbf{L}_U^{(q)}\mathbf{D}_s^{-1/2}. \qquad \text{(B.2)}$$

Thus, letting $\mathbf{y} = \mathbf{D}_s^{-1/2}\mathbf{x}$, the fact that $\mathbf{D}_s$ is diagonal implies

$$\mathbf{x}^\dagger \mathbf{L}_N^{(q)} \mathbf{x} = \mathbf{x}^\dagger \mathbf{D}_s^{-1/2}\mathbf{L}_U^{(q)}\mathbf{D}_s^{-1/2}\mathbf{x} = \mathbf{y}^\dagger \mathbf{L}_U^{(q)}\mathbf{y} \geq 0.$$

□

Theorem B.6.2. *Let $G = (V, E)$ be a directed graph where V is a set of N vertices and $E \subseteq V \times V$ is a set of directed edges. Then, for all $q \geq 0$, the eigenvalues of the normalized magnetic Laplacian $\mathbf{L}_N^{(q)}$ are contained in the interval $[0, 2]$.*

Proof. By Theorem B.6.1, we know that $\mathbf{L}_N^{(q)}$ has real, nonnegative eigenvalues. Therefore, we need to show that the lead eigenvalue, λ_N, is less than or equal to 2. The Courant-Fischer theorem shows that

$$\lambda_N = \max_{\mathbf{x}\neq 0} \frac{\mathbf{x}^\dagger \mathbf{L}_N^{(q)} \mathbf{x}}{\mathbf{x}^\dagger \mathbf{x}}.$$

Therefore, using (B.2) and setting $\mathbf{y} = \mathbf{D}_s^{-1/2}\mathbf{x}$, we have

$$\lambda_N = \max_{\mathbf{x}\neq 0} \frac{\mathbf{x}^\dagger \mathbf{D}_s^{-1/2} \mathbf{L}_U^{(q)} \mathbf{D}_s^{-1/2} \mathbf{x}}{\mathbf{x}^\dagger \mathbf{x}} = \max_{\mathbf{y}\neq 0} \frac{\mathbf{y}^\dagger \mathbf{L}_U^{(q)} \mathbf{y}}{\mathbf{y}^\dagger \mathbf{D}_s \mathbf{y}}.$$

First, we observe that since $\mathbf{D}_s$ is diagonal, we have

$$\mathbf{y}^\dagger \mathbf{D}_s \mathbf{y} = \sum_{u,v=1}^{N} \mathbf{D}_s(u,v)\mathbf{y}(u)\overline{\mathbf{y}(v)} = \sum_{u=1}^{N} \mathbf{D}_s(u,u)|\mathbf{y}(u)|^2$$

Next, we note that by (B.1), we have

$$\begin{aligned}
\mathbf{y}^\dagger \mathbf{L}_U^{(q)} \mathbf{y} &= \frac{1}{2} \sum_{u,v=1}^{N} \mathbf{A}_s(u,v) \left(|\mathbf{x}(u)|^2 + |\mathbf{x}(v)|^2 - 2\mathbf{x}(u)\overline{\mathbf{x}(v)} \cos(i\boldsymbol{\Theta}^{(q)}(u,v)) \right) \\
&\leq \frac{1}{2} \sum_{u,v=1}^{N} \mathbf{A}_s(u,v) (|\mathbf{x}(u)| + |\mathbf{x}(v)|)^2 \\
&\leq \sum_{u,v=1}^{N} \mathbf{A}_s(u,v) (|\mathbf{x}(u)|^2 + |\mathbf{x}(v)|^2).
\end{aligned}$$

Therefore, since $\mathbf{A_s}$ is symmetric, we have

$$\begin{aligned}
\mathbf{y}^\dagger \mathbf{L}_U^{(q)} \mathbf{y} &\leq 2 \sum_{u,v=1}^{N} \mathbf{A}_s(u,v) |\mathbf{x}(u)|^2 \\
&= 2 \sum_{u=1}^{N} |\mathbf{x}(u)|^2 \left(\sum_{v=1}^{N} \mathbf{A}_s(u,v) \right) \\
&= 2 \sum_{u=1}^{N} \mathbf{D}_s(u,u) |\mathbf{x}(u)|^2 \\
&= 2\mathbf{y}^\dagger \mathbf{D}_s \mathbf{y}.
\end{aligned}$$

□

Definition B.6.3. Let $G = (V, E)$ be a directed graph where V is a set of N vertices and $E \subseteq V \times V$ is a set of directed edges. We say that a link $(u, v) \in E$ is *bidirectional* if the "reverse" link (v, u) is also also in E. If a link is not bidirectional we say that it is *unidirectional*.

Theorem B.6.4. *Let $G = (V, E)$ be a directed graph where V is a set of N vertices and $E \subseteq V \times V$ is a set of directed edges. Then, for all $q \geq 0$, the unnormalized magnetic Laplacian may be factored as $\mathbf{L}_U^{(q)} = \mathbf{B}^{(q)}(\mathbf{B}^{(q)})^\dagger$, where $\mathbf{B}^{(q)}$ is a modified incidence matrix defined by*

$$\mathbf{B}^{(q)}(j,\ell) = \begin{cases} \frac{1}{\sqrt{2}} e^{i\pi q} & \text{if } j \text{ is the source of link } \ell \text{ and } \ell \text{ is unidirectional} \\ \frac{-1}{\sqrt{2}} e^{-i\pi q} & \text{if } j \text{ is the sink of the link } \ell \text{ and } \ell \text{ is unidirectional} \\ 1 & \text{if } j \text{ is the source of the link } \ell \text{ and } \ell \text{ is bidirectional} \cdot \\ 1 & \text{if } j \text{ is the sink of the link } \ell \text{ and } \ell \text{ is bidirectional} \\ 0 & \text{otherwise} \end{cases}$$

Proof. Let $\mathbf{B} = \mathbf{B}^{(q)}$ for the remainder of the proof. By definition we have,

$$(\mathbf{B}\mathbf{B}^\dagger)(j,k) = \sum_\ell \mathbf{B}(j,\ell)\overline{\mathbf{B}(k,\ell)}$$

If $j = k$, we have

$$\begin{aligned}
&(\mathbf{B}\mathbf{B}^\dagger)(j,j) \\
=&\sum_\ell \mathbf{B}(j,\ell)\overline{\mathbf{B}(j,\ell)} \\
=&\sum_{\substack{\ell \text{ unidirectional} \\ \text{st. } j \text{ is a source}}} \mathbf{B}(j,\ell)\overline{\mathbf{B}(j,\ell)} + \sum_{\substack{\ell \text{ unidirectional} \\ \text{st. } j \text{ is a sink}}} \mathbf{B}(j,\ell)\overline{\mathbf{B}(j,\ell)} \\
&+ \sum_{\substack{\ell \text{ bidirectional} \\ \text{st. } j \text{ is a source}}} \mathbf{B}(j,\ell)\overline{\mathbf{B}(j,\ell)} + \sum_{\substack{\ell \text{ bidirectional} \\ \text{st. } j \text{ is a sink}}} \mathbf{B}(j,\ell)\overline{\mathbf{B}(j,\ell)} \\
=&\sum_{\substack{\ell \text{ unidirectional} \\ \text{st. } j \text{ is a source}}} \frac{1}{\sqrt{2}} e^{i\pi q} \overline{\frac{1}{\sqrt{2}} e^{i\pi q}} + \sum_{\substack{\ell \text{ unidirectional} \\ \text{st. } j \text{ is a sink}}} \frac{-1}{\sqrt{2}} e^{-i\pi q} \left(\overline{\frac{-1}{\sqrt{2}} e^{i\pi q}}\right) + \sum_{\substack{\ell \text{ bidirectional} \\ \text{st. } j \text{ is a source}}} 1 + \sum_{\substack{\ell \text{ bidirectional} \\ \text{st. } j \text{ is a sink}}} 1 \\
=&\frac{1}{2}(d_{in}(j) + d_{out}(j)) \\
=&d_s(j).
\end{aligned}$$

If $j \neq k$ and there is a link from j to k but not from k to j, then

$$(\mathbf{B}\mathbf{B}^\dagger)(j,k) = \sum_\ell \mathbf{B}(j,\ell)\overline{\mathbf{B}(k,\ell)} = \frac{1}{\sqrt{2}} e^{i\pi q}\left(\frac{-1}{\sqrt{2}} e^{i\pi q}\right) = \frac{-1}{2} e^{2\pi i q} = -\mathbf{H}^{(q)}(j,k)$$

Likewise, if there is a link from k to j but not from j to k we have

$$(\mathbf{B}\mathbf{B}^{\dagger})(j,k) = \sum_{\ell} \mathbf{B}(j,\ell)\overline{\mathbf{B}(k,\ell)} = \left(\frac{-1}{\sqrt{2}}e^{-i\pi q}\right)\frac{1}{\sqrt{2}}e^{-i\pi q} = \frac{-1}{2}e^{-2\pi i q} = -\mathbf{H}^{(q)}(j,k).$$

Lastly, if there is neither a link from k to j or j to k we have $(\mathbf{B}\mathbf{B}^{\dagger})(j,k) = 0$. □

B.7 The eigenvectors and eigenvalues of directed stars and cycles

In this section, we examine the eigenvectors and eigenvalues of the unnormalized magnetic Laplacian on two example graphs. As alluded to in the main text, in the directed star graph directional information is contained in the eigenvectors only. For the directed cycle, on the other hand, the magnetic Laplacian encodes the directed nature of the graph only through the eigenvalues. Both examples can be verified via direct pen and paper calculation.

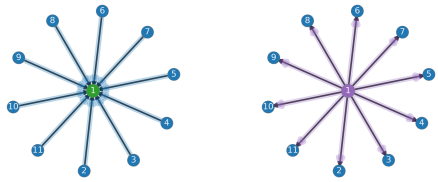


Figure B.1 Directed stars (a) $G^{(\text{in})}$, and (b) $G^{(\text{out})}$

Example 1. *Let $G^{(in)}$ and $G^{(out)}$ be the directed star graphs with vertices $V = \{1,\dots,N\}$ and edges pointing in/out to the central vertex as shown in Figure B.1. Then the eigenvalues of $\mathbf{L}_U^{(q,in)}$, the unnormalized magnetic Laplacian on G^{in}, are given by*

$$\lambda_1^{in} = 0, \quad \lambda_k^{in} = \frac{1}{2} \text{ for } 2 \le k \le N-1, \quad \text{and} \quad \lambda_N^{in} = \frac{N}{2}.$$

If we let $v = 1$ be the central vertex, then the lead eigenvector is given by

$$\mathbf{u}_1^{in}(1) = e^{2\pi i q}, \quad \mathbf{u}_1^{in}(n) = 1,\ 2 \le n \le N.$$

For $2 \le k \le N-1$, the eigenvectors are

$$\mathbf{u}_k^{in} = \boldsymbol{\delta}_k - \boldsymbol{\delta}_{k+1},$$

and the final eigenvector is given by

$$\mathbf{u}_N^{in}(1) = -e^{2\pi iq}, \quad \mathbf{u}_N^{in}(n) = \frac{1}{N-1},\ 2 \le n \le N.$$

The phase matrices satisfies $\mathbf{\Theta}^{(q,in)} = -\mathbf{\Theta}^{(q,out)}$. *Therefore, the associated magnetic Laplacians satisfy* $\mathbf{L}_U^{(q,in)}(u,v) = \overline{\mathbf{L}_U^{(q,out)}(u,v)}$. *Since these matrices are Hermitian, this implies that the corresponding eigenvalue-eigenvector pairs satisfy* $\lambda_k^{in} = \lambda_k^{out}$, *and* $\mathbf{u}_k^{in} = \overline{\mathbf{u}_k^{out}}$. *Hence,* $\mathbf{u}_1^{in}$ *and* $\mathbf{u}_1^{out}$ *identify the central vertex, and the sign of their imaginary parts at this vertex identifies whether it is a source or a sink. On the other hand, the eigenvalues give no directional information.*

Example 2. *Let G be the directed cycle. Then, then the eigenvalues of* $\mathbf{L}_U^{(q)}$ *is are the classical Fourier modes* $\mathbf{u}_k(n) = e^{(2\pi ikn/N)}$, *independent of q. The eigenvalues, however, do depend on q and are given by*

$$\lambda_k = 1 - \cos\left(2\pi\left(\frac{k}{N} + q\right)\right), \quad 1 \le k \le N.$$

APPENDIX C

UNSUPERVISED LEARNING OF FULL-WAVEFORM INVERSION: CONNECTING CNN AND PARTIAL DIFFERENTIAL EQUATION IN A LOOP

C.1 Network Architecture

Since the number of receivers R and the number of timesteps T in seismic measurements are unbalanced ($T \gg R$), we first stack a 7×1 and six 3×1 convolutional layers (with stride 2 every the other layer to reduce dimension) to extract temporal features until the temporal dimension is close to R. Then, six 3×3 convolutional layers are followed to extract spatial-temporal features. The resolution is down-sampled every the other layer by using stride 2. Next, the feature map is flattened and a fully connected layer is applied to generate the latent feature with dimension 512. The decoder first repeats the latent vector by 25 times to generate a 5×5×512 tensor. Then it is followed by five 3×3 convolutional layers with nearest neighbor upsampling in between, resulting in a feature map with size 80×80×32. Finally, we center-crop the feature map (70×70) and apply a 3×3 convolution layer to output a single channel velocity map.

All the aforementioned convolutional and upsampling layers are followed by a batch normalization (Ioffe and Szegedy, 2015) and a leaky ReLU (Nair and Hinton, 2010) as activation function.

C.2 Derivation of Forward Modeling in Practice

Similar to the finite difference in time domain, in 2D situation, by applying the fourth-order central finite difference in space, the Laplacian of $p(\boldsymbol{r}, t)$ can be discretized as

$$\begin{aligned}\nabla^2 p(\boldsymbol{r}, t) =& \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial z^2}, \\ \approx& \frac{1}{(\Delta x)^2} \sum_{i=-2}^{2} c_i p^t_{x+i,z} + \frac{1}{(\Delta z)^2} \sum_{i=-2}^{2} c_i p^t_{x,z+i} \\ &+ O\left[(\Delta x)^4 + (\Delta z)^4\right],\end{aligned} \tag{C.1}$$

where $c_0 = -\frac{5}{2}$, $c_1 = \frac{4}{3}$, $c_2 = -\frac{1}{12}$, $c_i = c_{-i}$, and x and z stand for the horizontal offset and the depth of a 2D velocity map, respectively. For convenience, we assume that the vertical grid spacing Δz is identical to the horizontal grid spacing Δx.

Given the approximation in Equations 3.21 and C.1, we can rewrite the Equation 3.14 as

$$p_{x,z}^{t+1} = (2 - 5\alpha)p_{x,z}^{t} - p_{x,z}^{t-1} - (\Delta x)^2 \alpha s_{x,z}^{t} + \alpha \sum_{\substack{i=-2 \\ i\neq 0}}^{2} c_i (p_{x+i,z}^{t} + p_{x,z+i}^{t}) \ , \tag{C.2}$$

where $\alpha = (\frac{v\Delta t}{\Delta x})^2$.

During the simulation of the forward modeling, the boundaries of the velocity maps should be carefully handled because they may cause reflection artifacts that interfere with the desired waves. One of the standard methods to reduce the boundary effects is to add absorbing layers around the original velocity map. Waves are trapped and attenuated by a damping parameter when propagating through those absorbing layers. Here, we follow Collino and Tsogka (2001) and implement the damping parameter as

$$\kappa = d(u) = \frac{3uv}{2L^2} ln(R) \ , \tag{C.3}$$

where L denotes the overall thickness of absorbing layers, u indicates the distance between the current position and the closest boundary of the original velocity map, and R is the theoretical reflection coefficient chosen to be 10^{-7}. With absorbing layers added, Equation 3.22 can be ultimately written as

$$p_{x,z}^{t+1} = (2 - 5\alpha - \kappa)p_{x,z}^{t} - (1 - \kappa)p_{x,z}^{t-1} - (\Delta x)^2 \alpha s_{x,z}^{t} + \alpha \sum_{\substack{i=-2 \\ i\neq 0}}^{2} c_i (p_{x+i,z}^{t} + p_{x,z+i}^{t}) \ . \tag{C.4}$$

C.3 OpenFWI Examples and Inversion Results of Different Methods

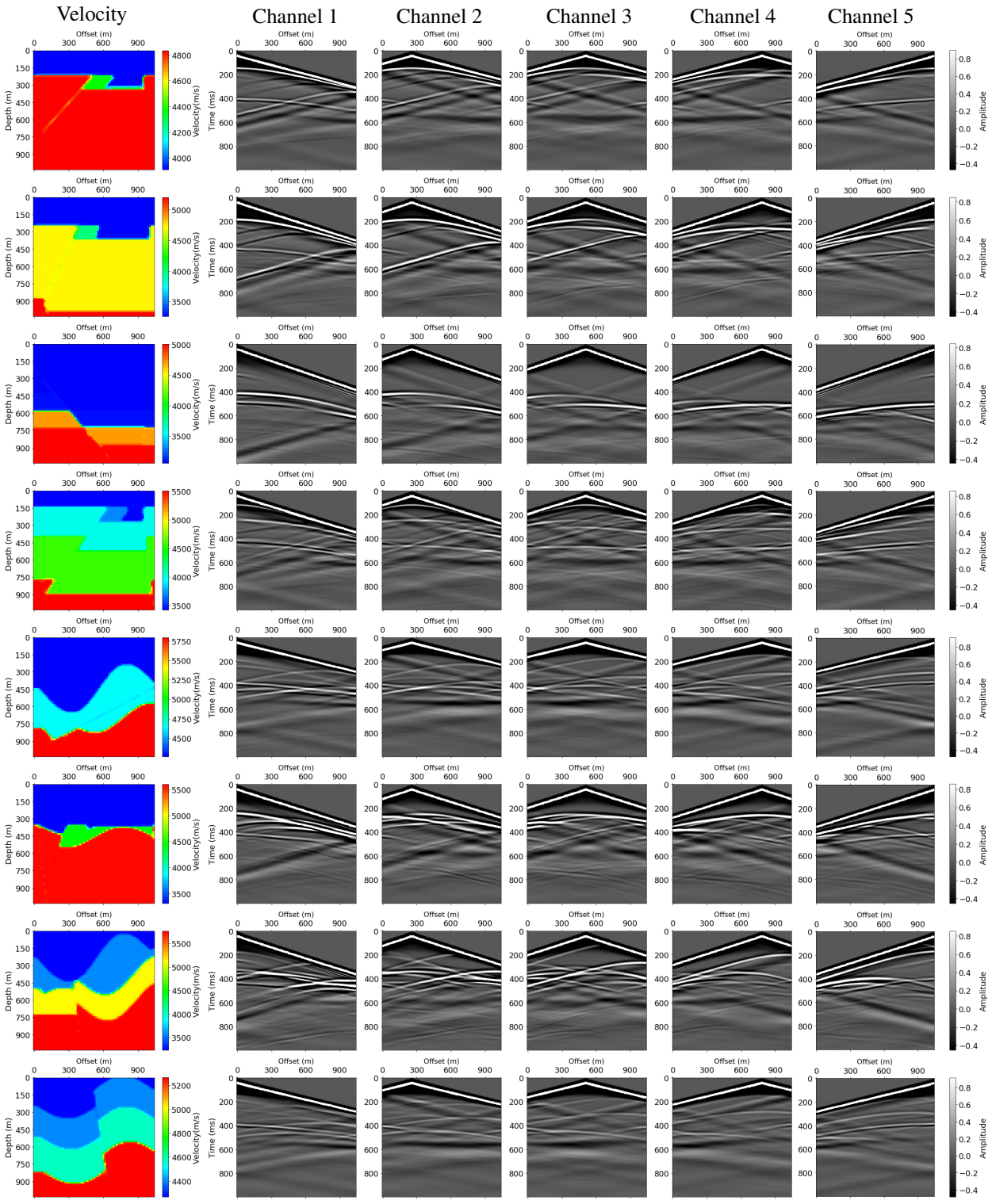


Figure C.1 More examples of velocity maps and their corresponding seismic measurements in OpenFWI dataset.

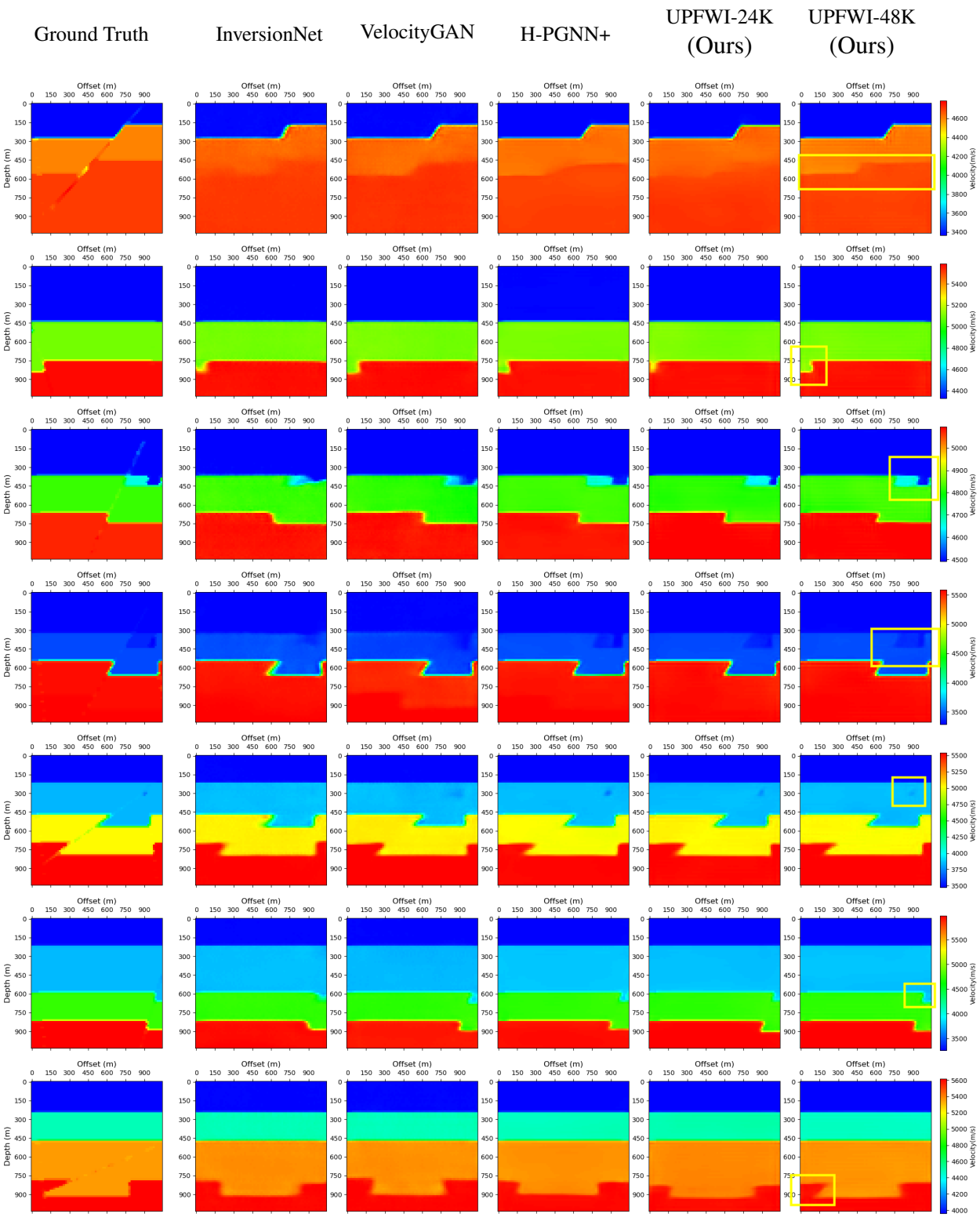


Figure C.2 Comparison of different methods on inverted velocity maps of FlatFault. The details revealed by our UPFWI are highlighted.

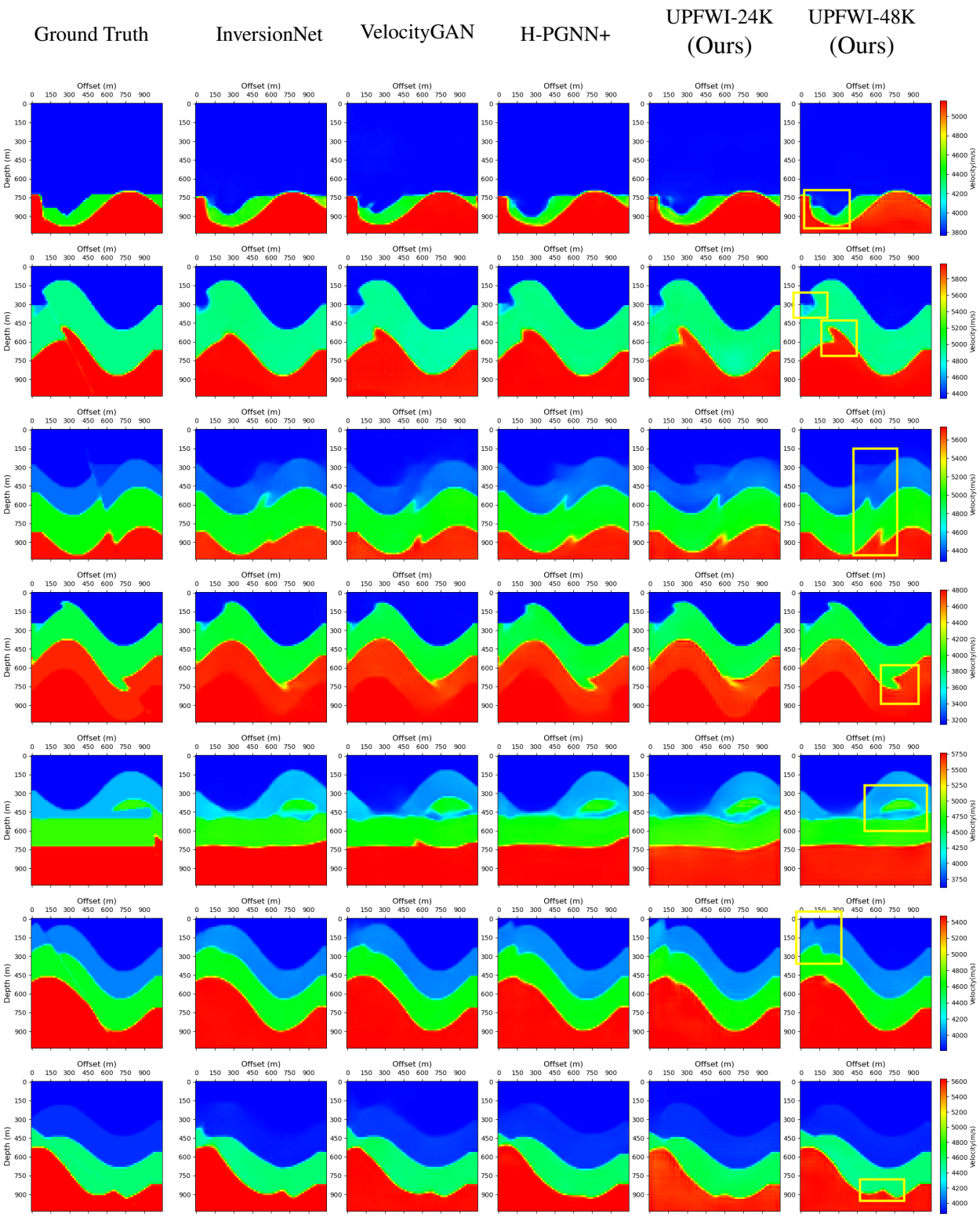


Figure C.3 Comparison of different methods on inverted velocity maps of CurvedFault. The details revealed by our UPFWI are highlighted.

C.4 Additional Experiment Results

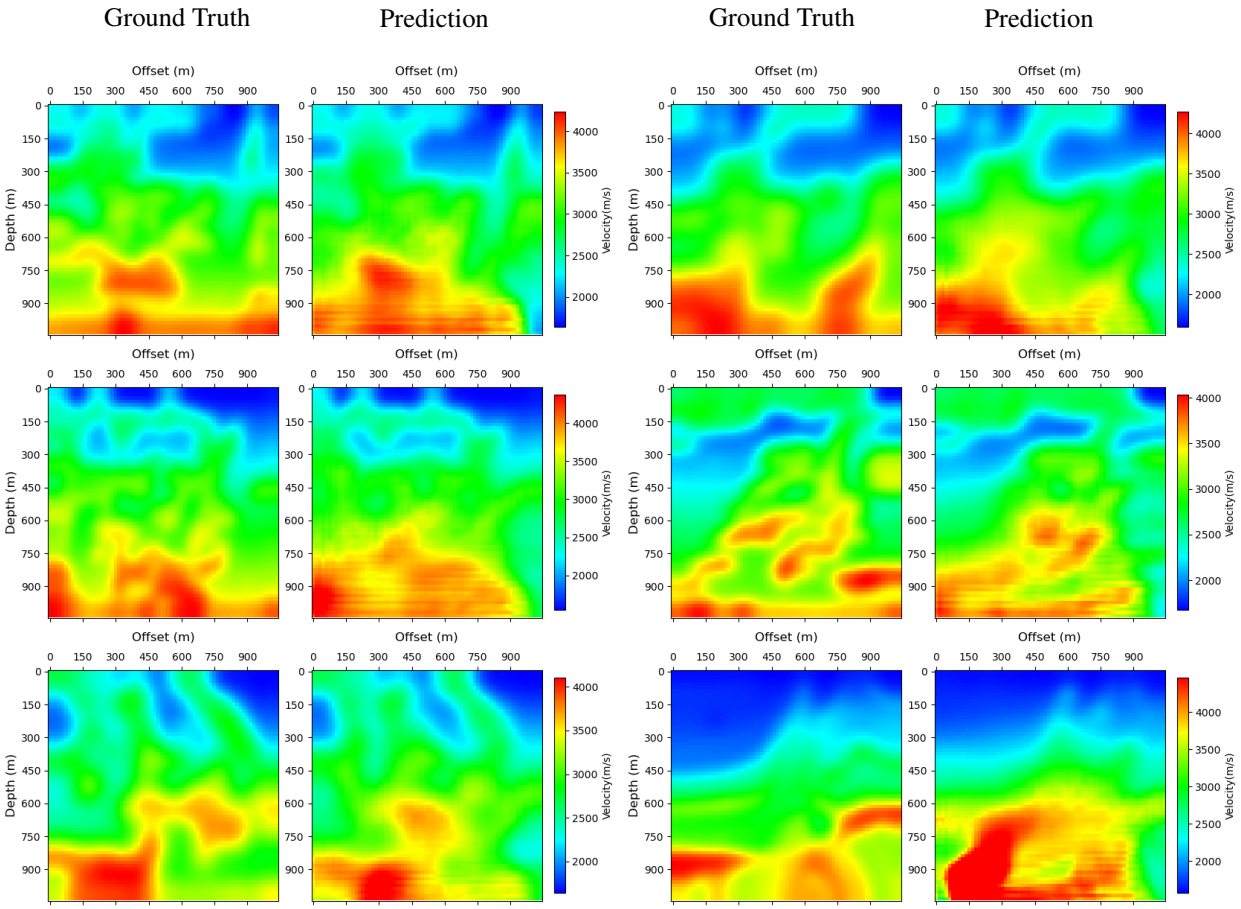


Figure C.4 Results of low-resolution Marmousi Dataset. This dataset contains low-resolution velocity maps generated using style transfer with the Marmousi velocity map as the style images. Our UPFWI model yields good results in shallow regions, and it also captures some geological structures in deeper regions. Similar phenomenon is also observed in the prediction of the smoothed Marmousi velocity map (bottom-right corner).

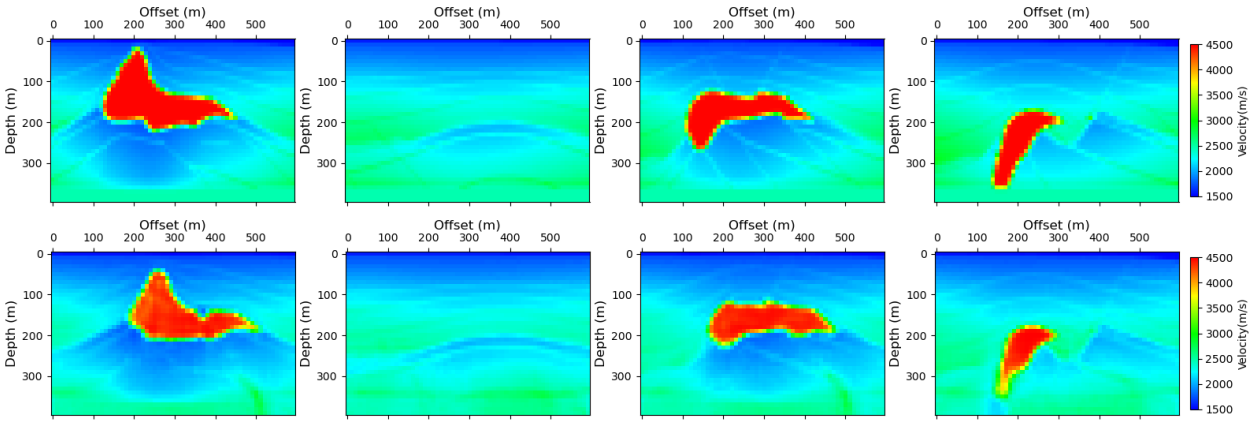


Figure C.5 Results of salt bodies dataset. This dataset contains more complicated velocity maps. Our UPFWI model yields good velocity map prediction (bottom) on both salt bodies and background geological structures compared to the ground truth (top).

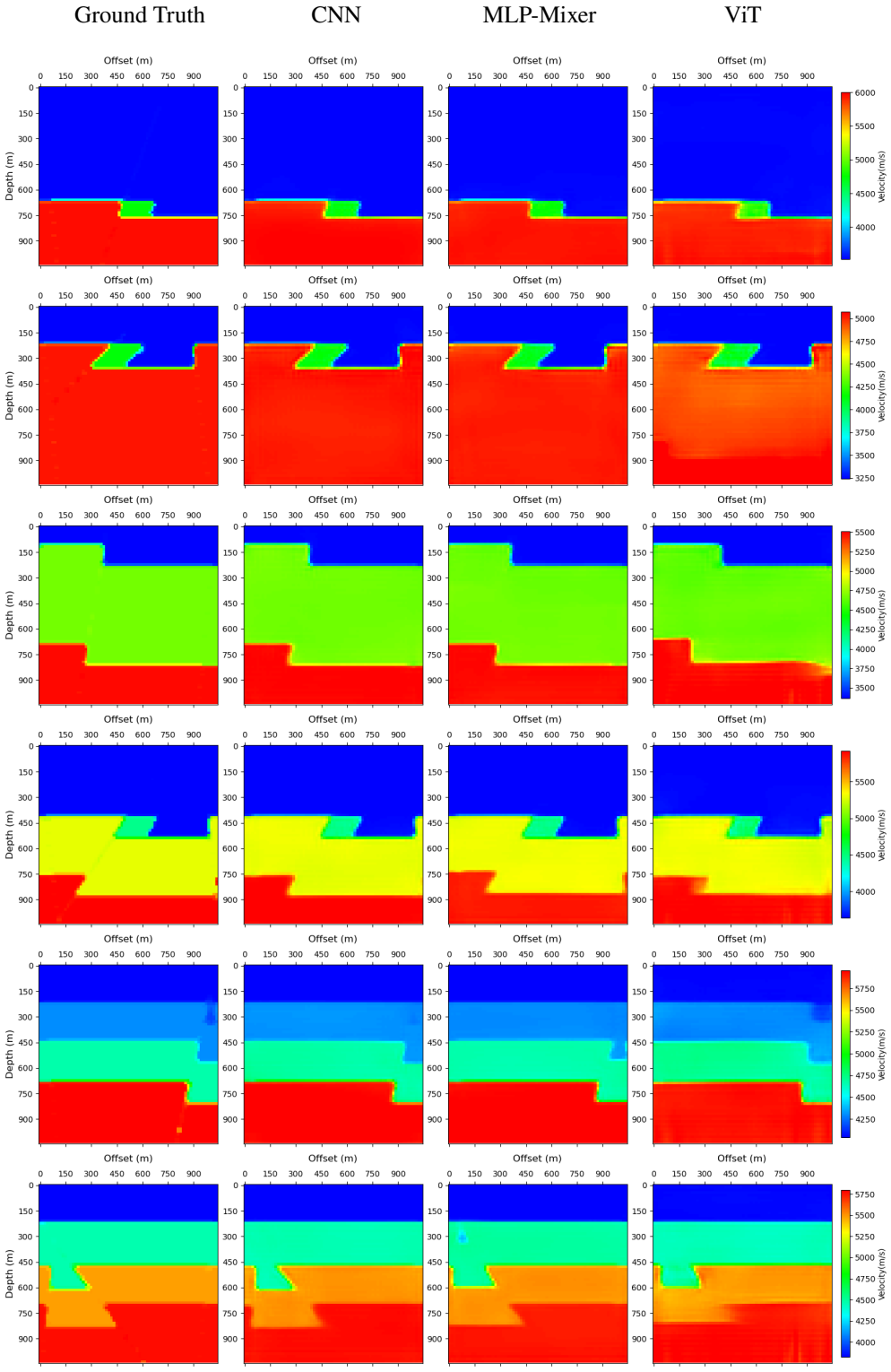


Figure C.6 Results of UPFWI with different network architectures. We replace the CNN in our model with Vision Transformer (ViT) and MLP-Mixer as the encoder and test them on the FlatFault dataset. Both models yield reasonable velocity maps. This demonstrates that our proposed learning paradigm is model-agnostic.

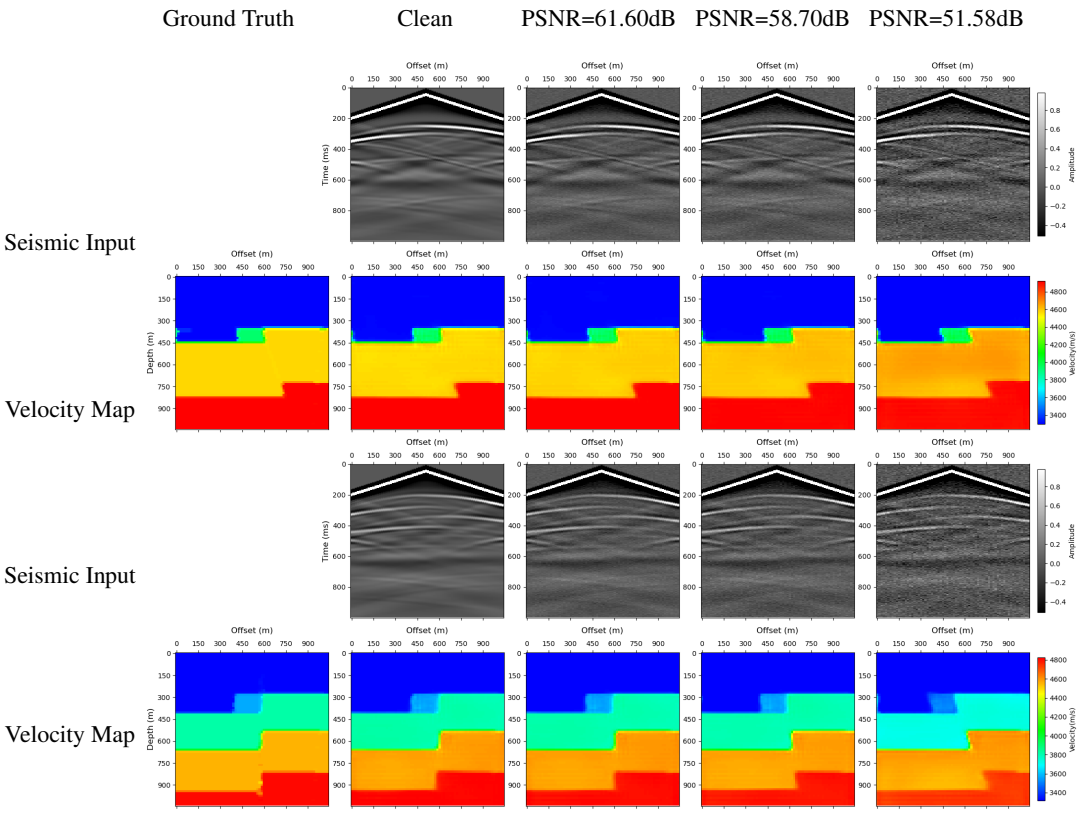


Figure C.7 Results of adding Gaussian noise to FlatFault. The model is trained on the clean data (without noise) and tested on different levels (PSNR) of Gaussian noises. Clearly, our method is robust to the noise although slight degradation is observed when noise level increases.

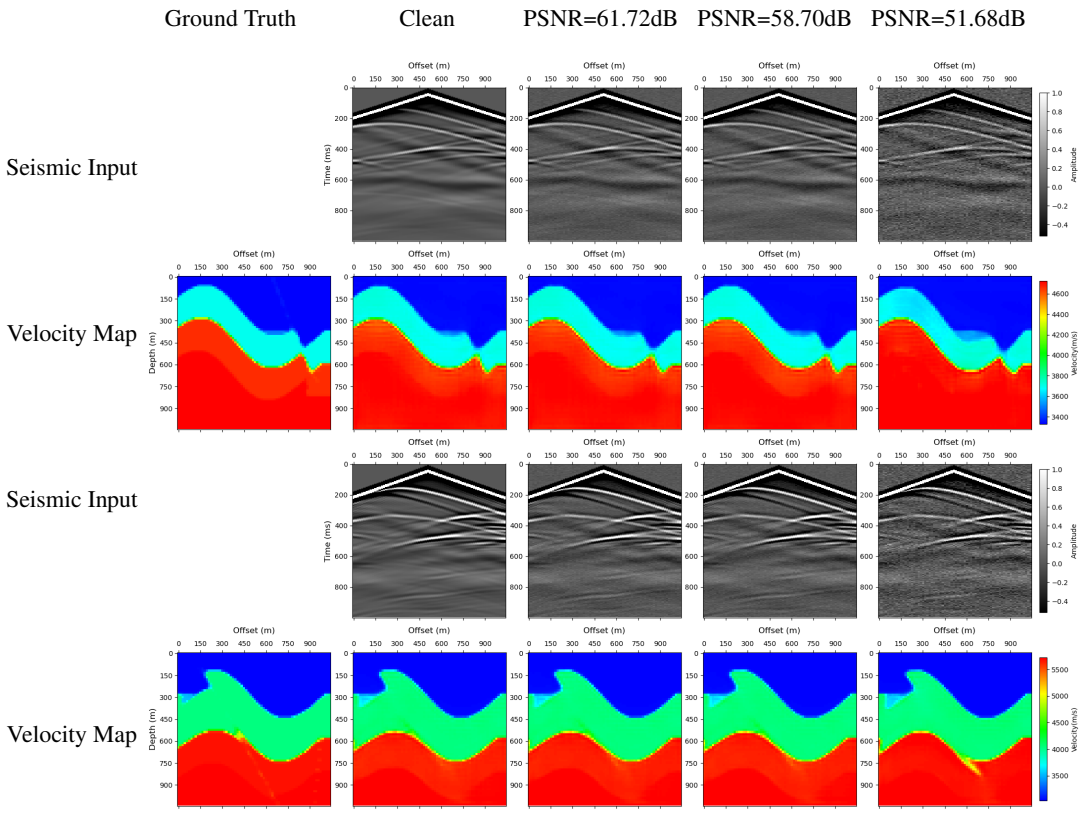


Figure C.8 Results of adding Gaussian noise to CurvedFault. The model is trained on the clean data (without noise) and tested on different levels (PSNR) of Gaussian noises. Similar to the results of FlatFault, our method is robust to the noise although slight degradation is observed when noise level increases.

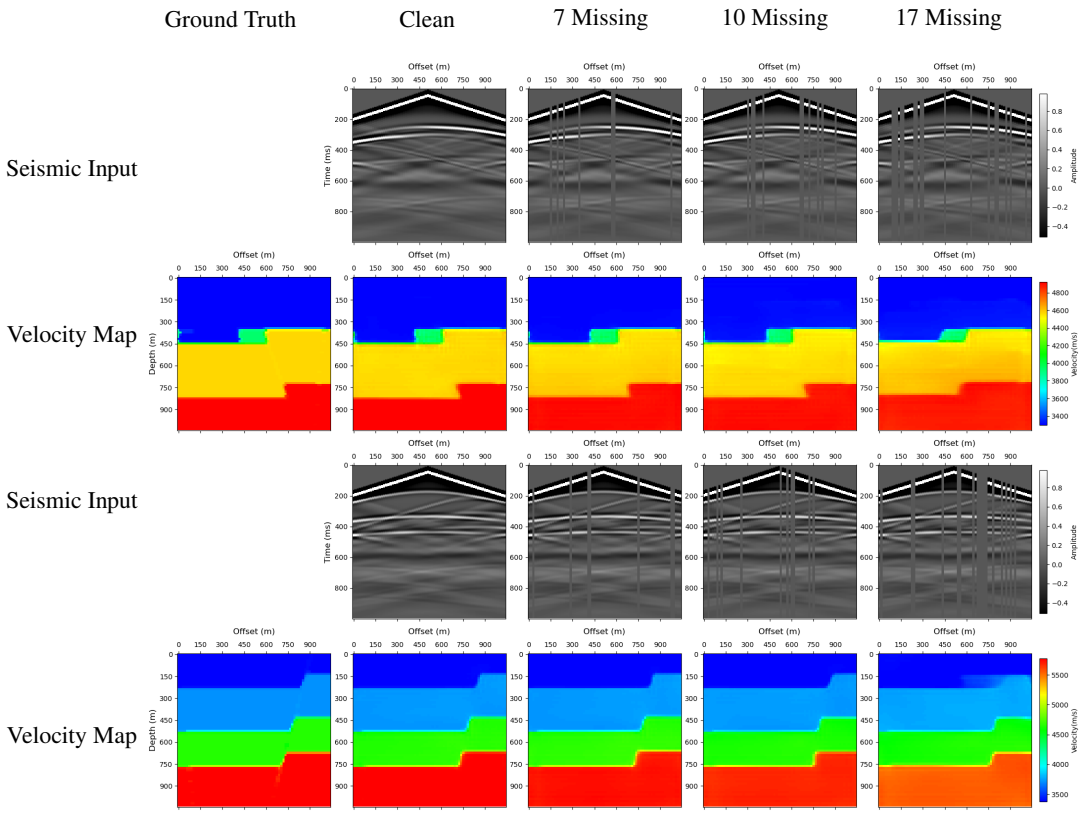


Figure C.9 Results of randomly missing traces on FlatFault. The model is trained on the clean data (without missing traces) and tested on multiple missing rates from 5% to 25%. Our method is robust to the missing traces. Although the higher missing rate leads to shifts in velocity values, the geological structures are well preserved.

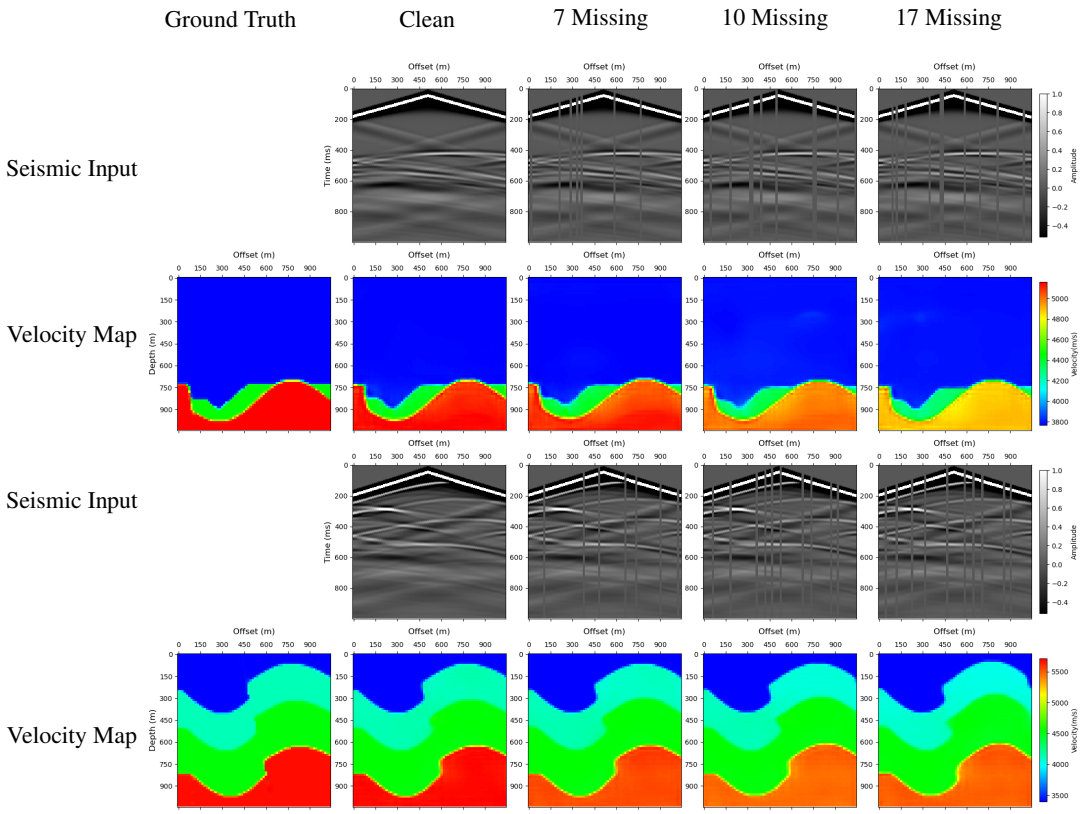


Figure C.10 Results of randomly missing traces on CurvedFault. The model is trained on the clean data (without missing traces) and tested on multiple missing rates from 5% to 25%. Similar to the results of FlatFault, our method is robust to the missing traces. Although the higher missing rate leads to shifts in velocity values, the geological structures are well preserved.

Additional experiments to investigate generalization. We conducted two additional experiments: (1) training our model on the CurvedFault dataset and further testing on the FlatFault dataset (visualization results are listed in Figure C.11, and quantitative results are shown in Table C.1); (2) testing our model on time-lapse imaging problems (visualization results are listed in Figure C.12). The results demonstrate that our proposed model yields generalization ability to a certain degree.

Table C.1 Quantitative results of our UPFWI models evaluated on FlatFault.

Training Dataset	Test Dataset	MAE↓	MSE↓	SSIM↑
FlatFault	FlatFault	14.60	1146.09	0.9895
CurvedFault	FlatFault	50.80	17627.65	0.9253

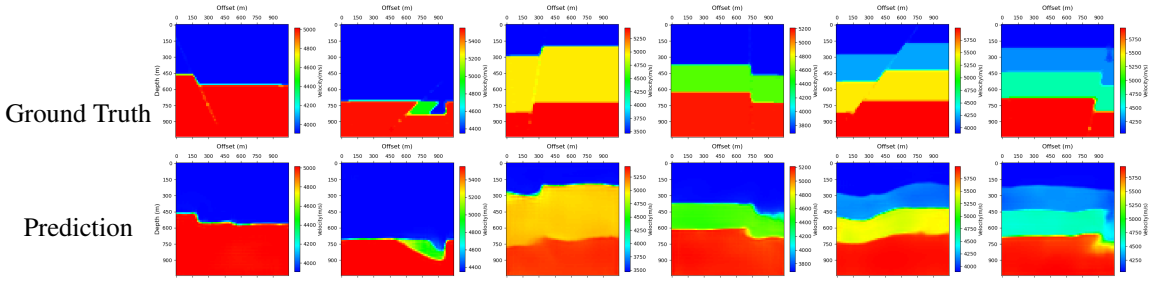


Figure C.11 Results on generalization across datasets. The test is performed on FlatFault by applying a UPFWI model that is trained on CurvedFault dataset. Although the artifact is not negligible, the fault structures and velocity values are well preserved. This demonstrates that our model has generalizability to a certain degree.

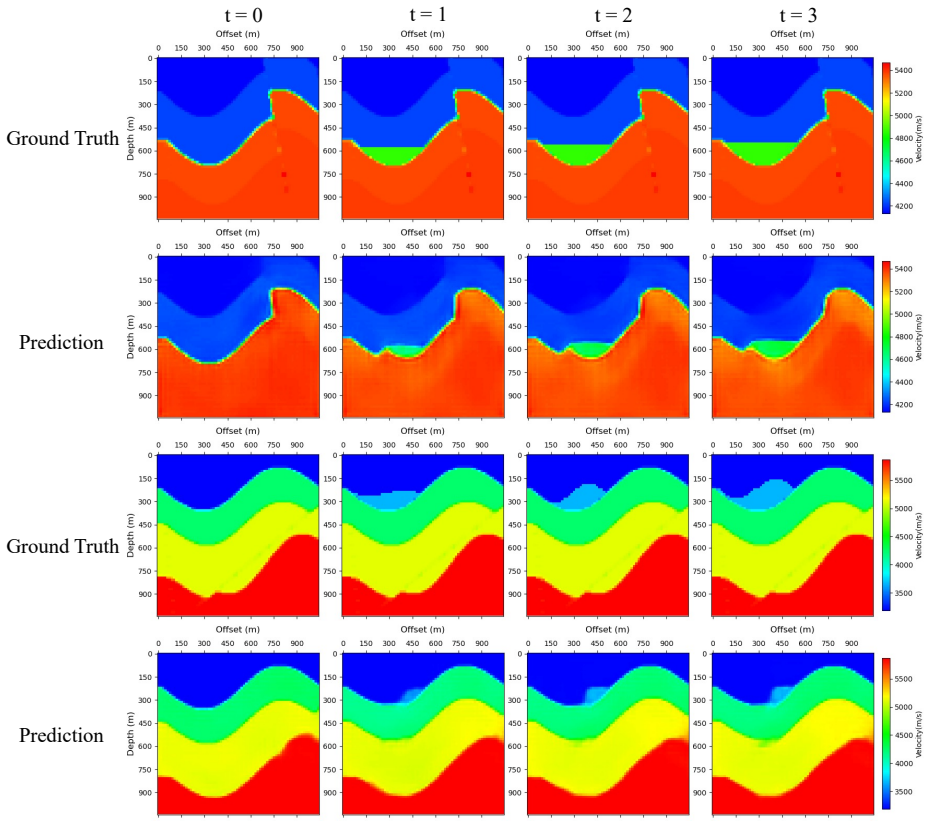


Figure C.12 Results on generalizability over geological anomalies. The test is performed on a dataset where we add additional geological anomalies to simulate time-lapse imaging problems. The velocity maps containing those anomalies are not included during training. However, our model captures the spatial and temporal dynamics of anomalies in prediction. This demonstrates that our model has generalizability to a certain degree.