Unleashing the Power of PAC-Bayes Training for Unbounded Loss

Anonymous Authors[1]

Abstract

Previous research on PAC-Bayes learning theory has focused extensively on establishing tight upper bounds for test errors. A recently proposed training procedure, called *PAC-Bayes training*, updates the network weights toward minimizing these bounds. Although this approach is theoretically sound, in practice, it has not achieved a test error as low as those obtained by empirical risk minimization (ERM) with carefully tuned regularization hyperparameters. Additionally, existing PAC-Bayes training algorithms (e.g., Pérez-Ortiz et al. (2021)) often require bounded loss functions and may need a search over priors with additional datasets, which limits their broader applicability. In this paper, we introduce a new PAC-Bayes training algorithm with improved performance and reduced reliance on prior tuning. This is achieved by establishing a new PAC-Bayes bound for unbounded loss and a theoretically grounded approach that involves jointly training the prior and posterior using the same dataset. Our comprehensive evaluations across various classification tasks and neural network architectures demonstrate that the proposed method not only outperforms existing PAC-Bayes training algorithms but also approximately matches the test accuracy of ERM that is optimized by SGD/Adam using various regularization methods with optimal hyperparameters.

1. Introduction

The PAC-Bayes bound plays a vital role in assessing generalization by estimating the upper limits of test errors without using validation data. It provides essential insights into the generalization ability of trained models and offers theoretical backing for practical training algorithms (Shawe-Taylor & Williamson, 1997). For example, PAC-Bayes bounds highlight the discrepancy between the training and generalization errors, indicating the need to incorporate regularizers in empirical risk minimization and explaining how larger datasets contribute to improved generalization. Furthermore, the effectiveness of the PAC-Bayes bounds in estimating the generalization capabilities of machine learning models has been supported by extensive experiments across different generalization metrics (Jiang et al., 2019).

Traditionally, PAC-Bayes bounds have been primarily used for quality assurance or model selection (McAllester, 1998; 1999; Herbrich & 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 & 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 overparameterized settings (Livni & 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 & 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 & Dherin, 2020), momentum (Ghosh et al., 2022; Cattaneo et al., 2023), smaller batch sizes (Lee & 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

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

weight decay (Loshchilov & 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. In this paper, we propose 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. 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. We will use $||\cdot||$ denoting 2-norm.

Theorem 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 $\mathcal{H}$,*

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

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 any data-independent prior $\mathcal{P}$ and any posterior $\mathcal{Q}$, which enables one to further optimize the bound by searching for the best posterior. In practice, the posterior expectation can be the well-trained model based on data, and the prior expectation can be the initial model.

3. Related Work

The PAC-Bayes bound was first used to train neural networks in Dziugaite & Roy (2017). Specifically, the bound McAllester (1999) is used 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 & Guedj, 2021; Perez-Ortiz et al., 2021; Zhou et al., 2018) 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 & Catoni, 2011; Alquier & Guedj, 2018; Holland, 2019; Kuzborskij & 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.

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 & 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, their 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 & 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.

4. New PAC-Bayes Bound for Unbounded loss

Existing PAC-Bayes training algorithms (Dziugaite & Roy, 2017; 2018; Pérez-Ortiz et al., 2021) are limited to bounded loss. When dealing with unbounded Cross-Entropy loss[1], these algorithms 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 losses 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 1 of Section 6), meaning that the numerical value of the bound is much larger than the actual test error.

We propose a more feasible PAC-Bayes bound that imposes milder conditions while remaining nearly non-vacuous, making it suitable for PAC-Bayes training in deep networks. The new bound is based on a modification of the existing assumption of the loss function, detailed as follows.

Definition 4.1 (Exponential moment on finite intervals)**.** Let X be a random variable and $0 < \gamma_1 \leq \gamma_2$ be two real numbers. We call any $K > 0$ an exponential moment bound of X over the interval $[\gamma_1, \gamma_2]$, when

$$\mathbb{E}[\exp(\gamma(\mathbb{E}[X] - X))] \leq \exp(\gamma^2 K) \tag{1}$$

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

Remark 4.2 (The first-order-moment condition). In the context of our main theorem, where Definition 4.1 is applied, the random variable X represents the loss function and is typically non-negative. This leads to the inequality $\mathbb{E}[X] - X \leq \mathbb{E}[X]$. With this plugged into (1), we can easily see that the existence of the exponential moment bound K is guaranteed by the boundedness of $\mathbb{E}[X]$. Explicitly, when $\mathbb{E}[X]$ is bounded, setting $K = \frac{\mathbb{E}[X]}{\gamma_1}$ is sufficient to make (1) satisfied. *This is to say, Definition 4.1 allows us to relax the current PAC-Bayes bound requirement of the loss from a finite second-order moment (Rodríguez-Gálvez et al., 2023) ~~to a more attainable finite first-order moment~~.*

Here, we provide some motivation for the above definition. Terms similar as inequality (1) arise naturally in various proofs [2] of PAC-Bayes bounds, serving as a key property that determines if the PAC-Bayes bound holds. Its resemblance to the sub-Gaussian distribution definition enables the straightforward extension of PAC-Bayes bounds from bounded to unbounded sub-Gaussian loss. However, PAC-Bayes bounds for sub-Gaussian loss are often vacuous in deep neural network applications. The key reason is that the sub-Gaussian bound requires the exponential moment inequality to hold across the entire range of $\gamma \in [0, \infty)$. This condition is overly stringent for most practical scenarios, where allowing γ to fall within a finite range suffices. Therefore, we relaxed the sub-Gaussian condition in the above new definition of the exponential moment.

Another modification we propose is making the exponential moment bound to depend on the prior. This adjustment allows for a further reduction of the bound with special priors. This is achieved by the following definition, where we extend the Definition 4.1 from a singular random variable to a family of random variables parameterized by models in a hypothesis space.

Definition 4.3 (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

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

Similar to Definition 4.1, when dealing with non-negative loss, the existence of the exponential moment bound $K_{\min}$ is guaranteed, provided that the first-order moment of the loss is bounded.

Now, we can establish the PAC-Bayes bound for losses that satisfy Definition 4.3. Specifically, we will set the $X(\mathbf{h})$ in Definition 4.3 to be the loss $\ell(\mathbf{h}, z_i)$, which is a random variable parameterized by $\mathbf{h}$ with randomness coming from the input data z_i.

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

[2] Such as the proof of Theorem 4.4 in Appendix A.1

Theorem 4.4 (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 function ℓ has bounded first-order moment with respect to the input data distribution and for any $\mathbf{h}$. Fix some $\delta \in (0,1)$. For any choice of i.i.d m-sized training dataset $\mathcal{S}$ according to $\mathcal{D}$, and all posterior distributions $\mathcal{Q}$ over $\mathcal{H}$, we have*

$$\mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\ell(\mathbf{h};\mathcal{D}) \leq \mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\ell(\mathbf{h};\mathcal{S}) + \frac{1}{\gamma m}(\log\frac{1}{\delta} + \mathrm{KL}(\mathcal{Q}||\mathcal{P}_{\boldsymbol{\lambda}})) + \gamma K(\boldsymbol{\lambda}) \tag{3}$$

holds for any fixed $\gamma \in [\gamma_1, \gamma_2]$ with probability at least $1-\delta$, provided that $\ell(\mathbf{h}, z)$ satisfies Definition 4.3 with bound $K(\boldsymbol{\lambda})$.

The proof is available in Appendix A.1.

In addition to relaxing the requirements on the loss function, our bound offers a more practical approach for training by allowing theoretically grounded optimization over both the posterior and the prior. We will first outline the training process that focuses on joint optimizing the prior and posterior, followed by a discussion of its theoretical guarantees. The procedure is similar to the one in Dziugaite & Roy (2017), but has been adapted to align with our newly proposed bound. We begin by parameterizing the posterior distribution as $\mathcal{Q}_{\boldsymbol{\sigma}}(\mathbf{h})$, 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, like 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, $k \ll m, d$; the relevance of this assumption will become apparent upon examining Theorem 4.7. For our PAC-Bayes training, we propose to optimize over all four variables: $\mathbf{h}$, γ, $\boldsymbol{\sigma}$, and $\boldsymbol{\lambda}$:

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

where

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

Although optimizing the parameters in the prior will likely help reduce the bound, this approach might initially appear to conflict with the conventional view that the prior should be independent of the data. However, through our theoretical analysis presented in this and the following section, we will demonstrate that permitting the prior with a limited number of training parameters is acceptable. It will not cause a break of the PAC-Bayes bound while helping reduce its value.

To derive our theorem, we need the following assumptions:

Assumption 4.5 (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 $\mathrm{KL}(\mathcal{Q}||\mathcal{P}_{\boldsymbol{\lambda}})$ 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 $|\mathrm{KL}(\mathcal{Q}||\mathcal{P}_{\boldsymbol{\lambda}}) - \mathrm{KL}(\mathcal{Q}||\mathcal{P}_{\tilde{\boldsymbol{\lambda}}})| \leq \eta_1(\|\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}}\|)$, for all pairs $\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}} \in \Lambda$ and for all $\mathcal{Q} \in \mathfrak{Q}$.

Assumption 4.6 (Continuity of the exponential moment bound)**.** Let $K_{\min}(\boldsymbol{\lambda})$ be as defined in Definition 4.3. 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 $\boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}} \in \Lambda$.

We will first present a general theorem applicable to all distribution families, which ensures the performance of the minimizer of (P). Following this, we will demonstrate that the Gaussian prior/posterior distribution, commonly used in practice, satisfies the two assumptions above.

Theorem 4.7 (PAC-Bayes bound for unbounded losses and **trainable** priors)**.** *Assume the loss function ℓ has bounded first-order moment with respect to the input data distribution and for any $\mathbf{h}$. 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 4.5 and Assumption 4.6, 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 $1-\epsilon$:*

$$\mathbb{E}_{\mathbf{h}\sim\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})}\ell(\mathbf{h};\mathcal{D}) \leq L_{PAC}(\hat{\mathbf{h}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}}) + \eta, \tag{4}$$

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 $\gamma_1, \gamma_2, \eta_2, m$ and the upper bounds of the parameters in the prior and posterior.

The proof is available in Appendix A.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.

5. A New PAC-Bayes Training Algorithm

5.1. Gaussian prior and posterior

For the L_{PAC} objective to have a closed-form formula, in this paper, we employ the Gaussian distribution family, consistent with previous research on PAC-Bayes training. 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 $\mathcal{N}(\boldsymbol{\theta}, \mathrm{diag}(\boldsymbol{\sigma}))$, denoted by $\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})$, 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 & 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 $\mathrm{KL}(\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P}_\lambda(\boldsymbol{\theta}_0))$ in (P) is:

$$\frac{1}{2}\left[-\mathbf{1}_d^\top \log(\boldsymbol{\sigma}) + d(\log(\lambda) - 1) + \frac{(\|\boldsymbol{\sigma}\|_1 + \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|^2)}{\lambda}\right]. \tag{5}$$

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 $\mathcal{N}(\boldsymbol{\theta}_0, \mathrm{BlockDiag}(\boldsymbol{\lambda}))$, where $\mathrm{BlockDiag}(\boldsymbol{\lambda})$ 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.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 (5) for $\mathrm{KL}(\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P}_{\boldsymbol{\lambda}}(\boldsymbol{\theta}_0))$ into the PAC-Bayes bound in Theorem 4.7, we have the following corollary that justifies the usage of PAC-Bayes bound on large neural networks with the trainable prior.

Corollary 5.1. *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 4.5 holds with $\eta_1(x) = L_1 x$, where $L_1 = \frac{1}{2}\max\{d, e^a(2\sqrt{d}M + dT)\}$*
- *Assumption 4.6 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*

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

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 $\gamma_1, \delta, M,\ d,\ T,\ a,\ b,\ m$[3].

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 η (and therefore the bound) would be small if prior's degree of freedom k is substantially less than the dataset size m. The proof and more discussions can be found in Appendix A.4.

5.2. Training algorithm

Estimating $K_{\min}(\boldsymbol{\lambda})$: In practice, the function $K_{\min}(\boldsymbol{\lambda})$ must be estimated first. Since we showed in Corollary 5.1 and Remark 4.2 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 independent of the data, this procedure of estimating $K_{\min}(\boldsymbol{\lambda})$ can be carried out before training. More details are in Appendix B.1.

Two-stage PAC-Bayes training: Algorithm 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 B.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 C.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,

$$\gamma' = \frac{1}{K_{\min}}\sqrt{\frac{\log\frac{1}{\delta} + \mathrm{KL}(\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P}_{\boldsymbol{\lambda}}(\boldsymbol{\theta}_0))}{m}},$$
$$\gamma^* = \min\{\max\{\gamma_1, \gamma'\}, \gamma_2\} \tag{6}$$

[3] See Appendix A.4 for the explicit form of B.

Algorithm 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.6.*
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 (23) (Appendix Algorithm 2)
*/*Stage 1*/*
for epoch = $1 : T_1$ **do**
  for sampling one batch s from $\mathcal{S}$ **do**
    //Ensure non-negative variances
    $\lambda \leftarrow \exp(b)$, $\boldsymbol{\sigma} \leftarrow \exp(\mathbf{v})$
    $\mathcal{P}_\lambda \leftarrow \mathcal{N}(\boldsymbol{\theta}_0; \lambda I_d)$, $\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta}) \leftarrow \boldsymbol{\theta} + \mathcal{N}(\mathbf{0}; \mathrm{diag}(\boldsymbol{\sigma}))$
    //Get the stochastic version of $\mathbb{E}_{\tilde{\boldsymbol{\theta}} \sim \mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})} \ell(f_{\tilde{\boldsymbol{\theta}}}; \mathcal{S})$
    Draw one $\tilde{\boldsymbol{\theta}} \sim \mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})$ and evaluate $\ell(f_{\tilde{\boldsymbol{\theta}}}; \mathcal{S})$
    Compute the KL divergence as (5)
    Compute γ as (6)
    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 $\tilde{\boldsymbol{\theta}} \sim \mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\boldsymbol{\theta})$ 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
$\hat{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$

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 & 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 $\mathbb{E}_{\boldsymbol{\theta} \sim \mathcal{Q}_{\boldsymbol{\sigma}}} \ell(f_{\boldsymbol{\theta}}; \mathcal{S})$ 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 & 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, 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 (5) 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 the weight decay term is now centered at $\boldsymbol{\theta}_0$ instead of $\mathbf{0}$, the coefficients of the weight decay change from layer to layer when using layerwise prior, and the noise injection has anisotropic variances. More discussions are available in Appendix B.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 B.4, we provide some mathematical intuition of why the deterministic predictor is expected to perform even better than the Bayesian predictor.

6. 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 baselines contain a variety of 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 1,

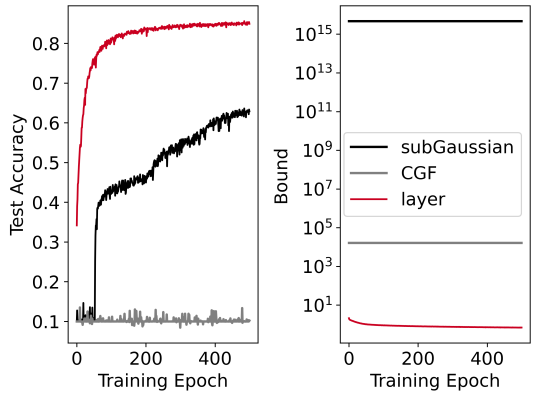


Figure 1: Training process when minimizing different PAC-Bayes bounds on CNN9 using CIFAR10 (Stage 1). Minimizing our bound (*layer*) achieves a tighter bound and better test accuracy compared with optimizing the other two (*subGaussian* and *CGF*).

show that our algorithm outperforms the other PAC-Bayes methods regarding test accuracy. It is important to note that all four baselines are designed 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 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 [4] performance than the proposed one as shown in Table 1. We also visualized the test accuracy when minimizing different PAC-Bayes bounds for unbounded loss in Stage 1. As shown in Figure 1, minimizing our PAC-Bayes bound can achieve better generalization performance. The details of the two baseline bounds are in Appendix C.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*[5] on various popular deep neural networks, VGG13, VGG19 (Simonyan & 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, the test accuracy of our method is still competitive. Please refer to Appendix C.3, C.5 for more details.

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

Table 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**

Table 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

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

[5] Result with data augmentation can be found in Appendix C.4

Table 3: Test accuracy of GNNs trained with AdamW (*AdW*) 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	AdW	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	AdW	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	AdW	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	AdW	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

CNNs, optimal GNN performance has been reported using the AdamW 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 & Welling, 2016), GAT (Veličković et al., 2017), SAGE (Hamilton et al., 2017), and APPNP (Gasteiger et al., 2018) on 5 benchmark datasets CoraML, Citeseer, PubMed, Cora and DBLP (Bojchevski & 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 3 demonstrates that the performance of our algorithm closely approximates the best outcome of the baseline. Appendix C.6 provides additional details and more results. Extra analysis on few-shot text classification with transformers is in Appendix C.7.

Evaluation on the sensitivity of hyperparameters: In previous experiments, we selected 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 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 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. Please refer to Appendix C.8 for more results.

Table 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 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

7. Conclusion and Discussion

In this paper, we demonstrated 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. Looking ahead, several promising avenues for future research emerge. While this study concentrates on Gaussian priors and posteriors, exploring alternative distributions could unveil unique advantages. In the realm of PAC-Bayes training, managing additional parameters such as $\boldsymbol{\lambda}, \boldsymbol{\sigma}$ leads to higher complexity, particularly in computing gradients for these parameters. Future research could focus on designing more efficient parameterization of the prior and posterior. Additionally, the increased difficulty in optimization due to additional parameters in PAC-Bayes training suggests a need for a thorough convergence analysis.

References

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

Audibert, J.-Y. and Catoni, O. Robust linear least squares regression. *The Annals of Statistics*, 39(5):2766 – 2794, 2011. doi: 10.1214/11-AOS918. URL https://doi.org/10.1214/11-AOS918.

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

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

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

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

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

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

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

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

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

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

Dziugaite, G. K. and Roy, D. M. 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)*, 2017.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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

Lee, S. and Jang, C. 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*, 2022.

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

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

Liu, K. pytorchcifar, Feb 2021. URL https://github.com/kuangliu/pytorch-cifar. GitHub repository.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

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

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

Appendix

A. Proofs

A.1. Proofs of Theorem 4.4

Theorem A.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 $\mathcal{Q}$ over $\mathcal{H}$, we have*

$$\mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\ell(\mathbf{h};\mathcal{D}) \leq \mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\ell(\mathbf{h};\mathcal{S}) + \frac{1}{\gamma m}(\log\frac{1}{\delta} + \mathrm{KL}(\mathcal{Q}||\mathcal{P}_{\boldsymbol{\lambda}})) + \gamma K(\boldsymbol{\lambda}) \tag{7}$$

holds with probability at least $1-\delta$ when $\ell(\mathbf{h},\cdot)$ satisfies Definition 4.3 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 $\mathcal{Q}$ and prior distribution $\mathcal{P}_{\boldsymbol{\lambda}}$ evaluated over a hypothesis space $\mathcal{H}$.

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

$$\begin{aligned}
&\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[\exp\left(\gamma m(\mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\ell(\mathbf{h};\mathcal{D}) - \mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\ell(\mathbf{h};\mathcal{S})) - \mathrm{KL}(\mathcal{Q}||\mathcal{P}_{\boldsymbol{\lambda}})\right)] \\
=&\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[\exp\left(\gamma m(\mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\ell(\mathbf{h};\mathcal{D}) - \mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\ell(\mathbf{h};\mathcal{S})) - \mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\log\frac{\mathrm{d}\mathcal{Q}}{\mathrm{d}\mathcal{P}_{\boldsymbol{\lambda}}}(\mathbf{h})\right)] \qquad (8)\\
\leq&\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}\mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}[\exp\left(\gamma m(\ell(\mathbf{h};\mathcal{D}) - \ell(\mathbf{h};\mathcal{S})) - \log\frac{\mathrm{d}\mathcal{Q}}{\mathrm{d}\mathcal{P}_{\boldsymbol{\lambda}}}(\mathbf{h})\right)] \qquad (9)\\
=&\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}\mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}[\exp(\gamma m(\ell(\mathbf{h};\mathcal{D}) - \ell(\mathbf{h};\mathcal{S})))\frac{\mathrm{d}\mathcal{P}_{\boldsymbol{\lambda}}}{\mathrm{d}\mathcal{Q}}(\mathbf{h})] \\
=&\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}\mathbb{E}_{\mathbf{h}\sim\mathcal{P}_{\boldsymbol{\lambda}}}[\exp(\gamma m(\ell(\mathbf{h};\mathcal{D}) - \ell(\mathbf{h};\mathcal{S})))\frac{\mathrm{d}\mathcal{P}_{\boldsymbol{\lambda}}}{\mathrm{d}\mathcal{Q}}(\mathbf{h})\frac{\mathrm{d}\mathcal{Q}}{\mathrm{d}\mathcal{P}_{\boldsymbol{\lambda}}}(\mathbf{h})] \qquad (10)\\
=&\mathbb{E}_{\mathbf{h}\sim\mathcal{P}_{\boldsymbol{\lambda}}}\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[\exp(\gamma m(\ell(\mathbf{h};\mathcal{D}) - \ell(\mathbf{h};\mathcal{S})))], \qquad (11)
\end{aligned}$$

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

In (8), we use $\mathrm{KL}(\mathcal{Q}||\mathcal{P}_\lambda) = \mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\log\frac{\mathrm{d}\mathcal{Q}}{\mathrm{d}\mathcal{P}_\lambda}(\mathbf{h})$. From (8) to (9), Jensen's inequality is used over the convex exponential function. And in (10), the expectation is changed to the prior distribution from the posterior.

Let $X = \ell(\mathbf{h};\mathcal{D}) - \ell(\mathbf{h};\mathcal{S})$, then X is centered with $\mathbb{E}[X] = 0$. Then, by Definition 4.3,

$$\exists K(\boldsymbol{\lambda}),\ \mathbb{E}_{\mathbf{h}\sim\mathcal{P}_\lambda}\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}[\exp\left(\gamma m X\right)] \leq \exp\left(m\gamma^2 K(\boldsymbol{\lambda})\right). \tag{12}$$

Using Markov's inequality, (13) holds with probability at least $1-\delta$.

$$\exp\left(\gamma m X\right) \leq \frac{\exp\left(m\gamma^2 K(\boldsymbol{\lambda})\right)}{\delta}. \tag{13}$$

Combining (11) and (13), the following inequality holds with probability at least $1-\delta$.

$$\begin{aligned}
&\exp\left(\gamma m(\mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\ell(\mathbf{h};\mathcal{D}) - \mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\ell(\mathbf{h};\mathcal{S})) - \mathrm{KL}(\mathcal{Q}||\mathcal{P}_{\boldsymbol{\lambda}})\right) \leq \frac{\exp\left(m\gamma^2 K(\boldsymbol{\lambda})\right)}{\delta} \\
\Rightarrow&\gamma m(\mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\ell(\mathbf{h};\mathcal{D}) - \mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\ell(\mathbf{h};\mathcal{S})) - \mathrm{KL}(\mathcal{Q}||\mathcal{P}_{\boldsymbol{\lambda}}) \leq \log\frac{1}{\delta} + m\gamma^2 K(\boldsymbol{\lambda}) \\
\Rightarrow&\mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\ell(\mathbf{h};\mathcal{D}) \leq \mathbb{E}_{\mathbf{h}\sim\mathcal{Q}}\ell(\mathbf{h};\mathcal{S}) + \frac{1}{\gamma m}(\log\frac{1}{\delta} + \mathrm{KL}(\mathcal{Q}||\mathcal{P}_{\boldsymbol{\lambda}})) + \gamma K(\boldsymbol{\lambda}) \qquad (14)
\end{aligned}$$

The bound 14 is exactly the statement of the Theorem.

□

A.2. Proof of Theorem 4.7

Theorem A.2. *Let $n(\varepsilon) := \mathcal{N}(\Lambda, \|\cdot\|, \varepsilon)$ be the covering number of the set of the prior parameters. Under Assumption 4.5 and Assumption 4.6, the following inequality holds for the minimizer $(\hat{\mathbf{h}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}})$ of upper bound in (3) with probability as least $1-\epsilon$:*

$$\begin{aligned}\mathbb{E}_{\mathbf{h}\sim\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})}\ell(\mathbf{h};\mathcal{D}) &\le \mathbb{E}_{\mathbf{h}\sim\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})}\ell(\mathbf{h};\mathcal{S}) + \frac{1}{\hat{\gamma}m}\left[\log\frac{n(\varepsilon)+\frac{\gamma_2-\gamma_1}{\varepsilon}}{\epsilon} + \mathrm{KL}(\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\hat{\boldsymbol{\lambda}}})\right] + \hat{\gamma}K(\hat{\boldsymbol{\lambda}}) + \eta \\ &= L_{PAC}(\hat{\mathbf{h}}, \hat{\gamma}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\lambda}}) + \eta \end{aligned} \tag{15}$$

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 $B := \sup_{\boldsymbol{\lambda}\in\Lambda}\frac{1}{m\gamma_1^2}(\mathrm{KL}(\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\boldsymbol{\lambda}}) + \log\frac{1}{\delta}) + K(\boldsymbol{\lambda})$.

Proof: In this proof, we extend the PAC-Bayes bound 3 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 $||\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}}|| \le \varepsilon$.

Now we select the posterior distribution as $\mathcal{Q}_{\boldsymbol{\sigma}}(\mathbf{h})$, 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 $\boldsymbol{\lambda} \in \mathbb{R}^k$ ($k \ll m, d$).

Then the PAC-Bayes bound 3 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.,

$$\mathbb{E}_{\tilde{\mathbf{h}}\sim\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})}\ell(\tilde{\mathbf{h}};\mathcal{D}) \le \mathbb{E}_{\tilde{\mathbf{h}}\sim\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})}\ell(\tilde{\mathbf{h}};\mathcal{S}) + \frac{1}{\gamma m}(\log\frac{1}{\delta} + \mathrm{KL}(\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\boldsymbol{\lambda}})) + \gamma K(\boldsymbol{\lambda}) \tag{16}$$

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 4.5 and Assumption 4.6, we have:

$$\min_{\tilde{\boldsymbol{\lambda}}\in\mathbb{T}} |\mathrm{KL}(\mathcal{Q}||\mathcal{P}_{\boldsymbol{\lambda}}) - \mathrm{KL}(\mathcal{Q}||\mathcal{P}_{\tilde{\boldsymbol{\lambda}}})| \le \eta_1(\|\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}}\|) \le \eta_1(\varepsilon),$$

and

$$\min_{\tilde{\boldsymbol{\lambda}}\in\mathbb{T}} |K(\boldsymbol{\lambda}) - K(\tilde{\boldsymbol{\lambda}})| \le \eta_2(\|\boldsymbol{\lambda} - \tilde{\boldsymbol{\lambda}}\|) \le \eta_2(\varepsilon).$$

Similarly, for γ, a ε-net on its range $\gamma_1 \le \gamma \le \gamma_2$ 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:

$$\begin{aligned}\min_{\tilde{\gamma}\in\mathbb{T}'} |L_{PAC}(\hat{\mathbf{h}}, \gamma, \hat{\boldsymbol{\sigma}}, \boldsymbol{\lambda}) - L_{PAC}(\hat{\mathbf{h}}, \tilde{\gamma}, \hat{\boldsymbol{\sigma}}, \boldsymbol{\lambda})| &= \frac{1}{m}\left(\mathrm{KL}(\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\boldsymbol{\lambda}}) + \log\frac{1}{\delta}\right)\left|\frac{1}{\gamma} - \frac{1}{\tilde{\gamma}}\right| + |\gamma - \tilde{\gamma}|K(\boldsymbol{\lambda}) \\ &= \left(\frac{1}{m\gamma\tilde{\gamma}}(\mathrm{KL}(\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\boldsymbol{\lambda}}) + \log\frac{1}{\delta}) + K(\boldsymbol{\lambda})\right)|\gamma - \tilde{\gamma}| \\ &\le \left(\frac{1}{m\gamma_1^2}(\mathrm{KL}(\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\boldsymbol{\lambda}}) + \log\frac{1}{\delta}) + K(\boldsymbol{\lambda})\right)\varepsilon \\ &\le B\varepsilon,\end{aligned}$$

where $B := \sup_{\boldsymbol{\lambda}\in\Lambda}\frac{1}{m\gamma_1^2}(\mathrm{KL}(\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\boldsymbol{\lambda}}) + \log\frac{1}{\delta}) + K(\boldsymbol{\lambda})$, 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 (16), we have

$$\mathbb{E}_{\mathbf{h}\sim\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})}\ell(\mathbf{h};\mathcal{D}) \leq L_{PAC}(\hat{\mathbf{h}},\hat{\gamma},\hat{\boldsymbol{\sigma}},\hat{\boldsymbol{\lambda}}) + B\varepsilon + C(\eta_1(\varepsilon) + \eta_2(\varepsilon)),$$

where $B := \sup_{\boldsymbol{\lambda}\in\Lambda} \frac{1}{m\gamma_1^2}(\mathrm{KL}(\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\boldsymbol{\lambda}}) + \log\frac{1}{\delta}) + K(\boldsymbol{\lambda})$.

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

$$\begin{aligned}
\mathbb{E}_{\mathbf{h}\sim\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})}\ell(\mathbf{h};\mathcal{D}) &\leq \mathbb{E}_{\mathbf{h}\sim\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})}\ell(\mathbf{h};\mathcal{S}) + \frac{1}{\hat{\gamma}m}\left[\log\frac{n(\varepsilon)+\frac{\gamma_2-\gamma_1}{\varepsilon}}{\epsilon} + \mathrm{KL}(\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\mathbf{h}})||\mathcal{P}_{\hat{\boldsymbol{\lambda}}})\right] + \hat{\gamma}K(\hat{\boldsymbol{\lambda}}) + B\varepsilon + C(\eta_1(\varepsilon)+\eta_2(\varepsilon)) \\
&\leq L_{PAC}(\hat{\mathbf{h}},\hat{\gamma},\hat{\boldsymbol{\sigma}},\hat{\boldsymbol{\lambda}}) + \eta \qquad (17)
\end{aligned}$$

and the proof is completed. □

A.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 $\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta}) := \mathcal{N}(\boldsymbol{\theta}, \mathrm{diag}(\boldsymbol{\sigma}))$, 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 $\mathfrak{Q} := \{\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta}), \boldsymbol{\sigma} \in \Sigma, \boldsymbol{\theta} \in \Theta\}$, and the KL divergence between all such prior and posterior in $\mathfrak{P}$ and $\mathfrak{Q}$ is:

$$\mathrm{KL}(\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P}_{\boldsymbol{\lambda}}(\boldsymbol{\theta}_0)) = \frac{1}{2}\sum_{i=1}^{k}\left[-\mathbf{1}_{d_i}^\top \log(\boldsymbol{\sigma}_i) + d_i(\log(\boldsymbol{\lambda}_i) - 1) + \frac{\|\boldsymbol{\sigma}_i\|_1 + \|(\boldsymbol{\theta}-\boldsymbol{\theta}_0)_i\|^2)}{\boldsymbol{\lambda}_i}\right], \qquad (18)$$

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[6].

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

$$\mathrm{KL}(\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P}_{\lambda}(\boldsymbol{\theta}_0)) = \frac{1}{2}\left[-\mathbf{1}_d^\top \log(\boldsymbol{\sigma}) + d(\log(\lambda) - 1) + \frac{1}{\lambda}(\|\boldsymbol{\sigma}\|_1 + \|\boldsymbol{\theta}-\boldsymbol{\theta}_0\|^2)\right]. \qquad (19)$$

A.4. Proof of Corollary 5.1

Recall for the training, we proposed to optimize over all four variables: $\boldsymbol{\theta}$, γ, $\boldsymbol{\sigma}$, and $\boldsymbol{\lambda}$.

$$(\hat{\boldsymbol{\theta}},\hat{\gamma},\hat{\boldsymbol{\sigma}},\hat{\boldsymbol{\lambda}}) = \arg\min_{\substack{\boldsymbol{\theta},\boldsymbol{\lambda},\boldsymbol{\sigma},\\ \gamma\in[\gamma_1,\gamma_2]}} \underbrace{\mathbb{E}_{\tilde{\boldsymbol{\theta}}\sim\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})}\ell(f_{\tilde{\boldsymbol{\theta}}};\mathcal{S}) + \frac{1}{\gamma m}(\log\frac{1}{\delta} + \mathrm{KL}(\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P}_{\boldsymbol{\lambda}})) + \gamma K(\boldsymbol{\lambda})}_{\equiv L_{PAC}(\boldsymbol{\theta},\gamma,\boldsymbol{\sigma},\boldsymbol{\lambda})}. \qquad (20)$$

[6]Note that with a little ambiguity, the $\boldsymbol{\lambda}_i$ here has a different meaning from that in (23) and Algorithm 2, here $\boldsymbol{\lambda}_i$ means the ith element in $\boldsymbol{\lambda}$, whereas in (23) and Algorithm 2, $\boldsymbol{\lambda}_i$ means the ith element in the discrete set.

Corollary A.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 4.5 holds with $\eta_1(x) = L_1 x$, where $L_1 = \frac{1}{2}\max\{d, e^a(2\sqrt{d}M + dT)\}$*
- *Assumption 4.6 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*

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

where $\eta = \frac{k}{\gamma_1 m}\left(1 + \log\frac{2(CL+B)\Delta\gamma_1 m}{k}\right)$, $L = L_1 + L_2$, $\Delta := \max\{b + a, 2(\gamma_2 - \gamma_1)\}$, $B = \sup_{\boldsymbol{\lambda}\in\Lambda}\frac{1}{m\gamma_1^2}(\mathrm{KL}(\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})||\mathcal{P}_{\boldsymbol{\lambda}}) + \log\frac{1}{\delta}) + K(\boldsymbol{\lambda})$, 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 4.5 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

$$\frac{\partial \mathrm{KL}(\mathcal{Q}_{\boldsymbol{\sigma}}||\tilde{\mathcal{P}}_{\mathbf{v}})}{\partial v_i} = \frac{1}{2}[-d_i + e^{v_i}(\|\boldsymbol{\sigma}_i\|_1 + \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_{0,i}\|^2)],$$

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:

$$\|\nabla_{\mathbf{v}}\mathrm{KL}(\mathcal{Q}_{\boldsymbol{\sigma}}||\tilde{\mathcal{P}}_{\mathbf{v}})\|_2 \leq \|\nabla_{\mathbf{v}}\mathrm{KL}(\mathcal{Q}_{\boldsymbol{\sigma}}||\tilde{\mathcal{P}}_{\mathbf{v}})\|_1 \leq \frac{1}{2}\max\{d, e^a(2\sqrt{d}M + dT)\} \equiv L_1(d, M, T, a), \tag{21}$$

where we used the assumption $\|\boldsymbol{\sigma}\|_1 \leq dT$ and $\|\boldsymbol{\theta}_0\|_2, \|\boldsymbol{\theta}\|_2 \leq \sqrt{d}M$.

Equation 21 says $L_1(d, M, T, a)$ is a valid Lipschitz bound on the KL divergence and therefore Assumption 4.5 is satisfied by setting $\eta_1(x) = L_1(d, M, T, a)x$.

Next, we prove Assumption 4.6 is satisfied. We use $K_{\min}(\boldsymbol{\lambda})$ defined in Definition 4.3 as the $K(\boldsymbol{\lambda})$ in the PAC-Bayes training, and verify that it makes Assumption 4.6 hold.

$$\begin{aligned}
&|K_{\min}(\boldsymbol{\lambda}_1) - K_{\min}(\boldsymbol{\lambda}_2)| \\
&= \left|\sup_{\gamma\in[\gamma_1,\gamma_2]}\frac{1}{\gamma^2}\log(\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}_{\boldsymbol{\lambda}_1}}\mathbb{E}_{z\sim\mathcal{D}}[\exp(\gamma\ell(f_{\boldsymbol{\theta}}; z))]) - \sup_{\gamma\in[\gamma_1,\gamma_2]}\frac{1}{\gamma^2}\log(\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}_{\boldsymbol{\lambda}_2}}\mathbb{E}_{z\sim\mathcal{D}}[\exp(\gamma\ell(f_{\boldsymbol{\theta}}; z))])\right| \\
&\leq \sup_{\gamma\in[\gamma_1,\gamma_2]}\frac{1}{\gamma^2}\left|\log(\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}_{\boldsymbol{\lambda}_1}}\mathbb{E}_{z\sim\mathcal{D}}[\exp(\gamma\ell(f_{\boldsymbol{\theta}}; z))]) - \log(\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}_{\boldsymbol{\lambda}_2}}\mathbb{E}_{z\sim\mathcal{D}}[\exp(\gamma\ell(f_{\boldsymbol{\theta}}; z))])\right| \\
&= \sup_{\gamma\in[\gamma_1,\gamma_2]}\frac{1}{\gamma^2}\left|\log(\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}_{\boldsymbol{\lambda}_2}}\mathbb{E}_{z\sim\mathcal{D}}[\exp(\gamma\ell(f_{\boldsymbol{\theta}}; z))]\frac{p_{\boldsymbol{\lambda}_1}(\boldsymbol{\theta})}{p_{\boldsymbol{\lambda}_2}(\boldsymbol{\theta})}) - \log(\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}_{\boldsymbol{\lambda}_2}}\mathbb{E}_{z\sim\mathcal{D}}[\exp(\gamma\ell(f_{\boldsymbol{\theta}}; z))])\right| \\
&\leq \sup_{\gamma\in[\gamma_1,\gamma_2]}\frac{1}{\gamma^2}\sup_{\boldsymbol{\theta}\in\Theta}\left|\log\frac{p_{\boldsymbol{\lambda}_1}(\boldsymbol{\theta})}{p_{\boldsymbol{\lambda}_2}(\boldsymbol{\theta})}\right| \\
&\leq \frac{1}{\gamma_1^2}\sup_{\mathbf{h}\in\mathcal{H}}\left|\log\frac{p_{\boldsymbol{\lambda}_1}(\boldsymbol{\theta})}{p_{\boldsymbol{\lambda}_2}(\boldsymbol{\theta})}\right| \\
&\leq \frac{1}{\gamma_1^2}\left(2dM^2e^{2a} + \frac{d(a+b)}{2}\right)\|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\|,
\end{aligned}$$

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 4.6 is satisfied by setting $\eta_2(x) = L_2(d,M,\gamma_1,a)x$, where $L_2(d,M,\gamma_1,a) = \frac{1}{\gamma_1^2}\left(2dM^2e^{2a} + \frac{d(a+b)}{2}\right)$.

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 4.7, to get with probability $1-\epsilon$,

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})}\ell(f_{\boldsymbol{\theta}};\mathcal{D}) \\
&\leq \mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})}\ell(f_{\boldsymbol{\theta}};\mathcal{S}) + \frac{1}{\hat{\gamma}m}\left[\log\frac{n(\varepsilon)+\frac{\gamma_2-\gamma_1}{2\varepsilon}}{\epsilon} + \mathrm{KL}(\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})||\mathcal{P}_{\boldsymbol{\lambda}})\right] + \hat{\gamma}K_{\min}(\hat{\boldsymbol{\lambda}}) + \\
&\quad (CL(d,M,T,\gamma_1,a)) + B)\varepsilon.
\end{aligned} \tag{22}$$

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 (22), we obtain with probability $1-\epsilon$:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})}\ell(f_{\boldsymbol{\theta}};\mathcal{D}) \\
&\leq \mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{Q}_{\hat{\sigma}}(\hat{\boldsymbol{\theta}})}\ell(f_{\boldsymbol{\theta}};\mathcal{S}) + \frac{1}{\hat{\gamma}m}\left[\log\frac{1}{\epsilon} + \mathrm{KL}(\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})||\mathcal{P}_{\boldsymbol{\lambda}})\right] \\
&+ \hat{\gamma}K_{\min}(\hat{\boldsymbol{\lambda}}) + \rho + \frac{k}{\gamma_1 m}\log\frac{2(CL(d,M,T,\gamma_1,a)+B)\Delta}{\rho}.
\end{aligned}$$

where $\Delta := \max\{b+a, 2(\gamma_2-\gamma_1)\}$.

Optimizing over ρ, we obtain:

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})}\ell(f_{\boldsymbol{\theta}};\mathcal{D}) \\
&\leq \mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})}\ell(f_{\boldsymbol{\theta}};\mathcal{S}) + \frac{1}{\hat{\gamma}m}\left[\log\frac{1}{\epsilon} + \mathrm{KL}(\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}})||\mathcal{P}_{\boldsymbol{\lambda}})\right] \\
&+ \hat{\gamma}K_{\min}(\hat{\boldsymbol{\lambda}}) + \frac{k}{\gamma_1 m}\left(1 + \log\frac{2(CL(d,M,T,\gamma_1,a)+B)\Delta\gamma_1 m}{k}\right) \\
&= L_{PAC}(\hat{\boldsymbol{\theta}},\hat{\gamma},\hat{\boldsymbol{\sigma}},\hat{\boldsymbol{\lambda}}) + \frac{k}{\gamma_1 m}\left(1 + \log\frac{2(CL(d,M,T,\gamma_1,a)+B)\Delta\gamma_1 m}{k}\right).
\end{aligned}$$

Hence we have

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

where $\eta = \max\left(\frac{1}{\gamma_1 m}(1+\log(2(CL(d,M,T,\gamma_1,a)+B)(\gamma_2-\gamma_1)\gamma_1 m)), \frac{k}{\gamma_1 m}\left(1+\log\frac{2(CL(d,M,T,\gamma_1,a)+B)\Delta\gamma_1 m}{k}\right)\right)$. □

Remark A.4. In defining the boundedness of the domain Θ of $\boldsymbol{\theta}$ in Corollary 5.1, 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.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 5.1, 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 5.1 can be used as the generalization guarantee for large neural networks.

B. Algorithm Details

B.1. Algorithms to estimate $K(\boldsymbol{\lambda})$

Algorithm 2 Compute $K(\boldsymbol{\lambda})$ given a set of query priors

Input: γ_1 and γ_2, s query prior variances $\mathcal{V} = \{\boldsymbol{\lambda}_i \in \Lambda \subseteq \mathbb{R}^k, i = 1, ..., s\}$, 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 $K_{\min}(\boldsymbol{\lambda})$
for $\boldsymbol{\lambda}_i \in \mathcal{V}$ **do**
  Set up a discrete grid Γ for the interval $[\gamma_1, \gamma_2]$ of γ.
  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 (23)
  end for
  Solve $\hat{K}_i$ using (23)
end for
Fit a piece-wise linear function $\tilde{K}(\boldsymbol{\lambda})$ to the data $\{(\boldsymbol{\lambda}_i, \hat{K}_i)\}_{i=1}^s$

For a discrete set, $\{\boldsymbol{\lambda}_1,....,\boldsymbol{\lambda}_s\} \subseteq \Lambda$, we estimate the corresponding K_1,....K_s using the empirical version of (2), that is for any $i = 1, ..., s$, we solve

$$K_{\min}(\boldsymbol{\lambda}_i) = \arg\min_{K>0} K \\ \text{s.t.} \exp(\gamma^2 K) \geq \frac{1}{nm} \sum_{l=1}^{n} \sum_{j=1}^{m} \exp(\gamma(\ell(f_{\boldsymbol{\theta}_l}; \mathcal{S}) - \ell(f_{\boldsymbol{\theta}_l}; z_j))), \quad \forall\, \gamma \in [\gamma_1, \gamma_2], \tag{23}$$

where $\boldsymbol{\theta}_l \sim \mathcal{P}_{\boldsymbol{\lambda}_i}(\boldsymbol{\theta}_0), l = 1, ..., n$, are samples from the prior distribution and are fixed when solving (23) for $K_{\min}(\boldsymbol{\lambda}_i)$. This optimization problem can be solved by a bisection search. From the pairs $(\boldsymbol{\lambda}_i, K_{\min}(\boldsymbol{\lambda}_i))$, we construct $K_{\min}(\boldsymbol{\lambda})$ using piecewise-linear interpolation.

Evaluating $\boldsymbol{\lambda}$ when $k = 1$ using the algorithm above is simple. When using the layerwise prior version of our method, the exponential moment bound $K_{\min}(\boldsymbol{\lambda})$ is a k-dimensional function, and estimating it accurately requires many samples. This paper adopts a simple empirical approach to address this issue. We assume that the original function $K_{\min}(\boldsymbol{\lambda})$ can be well-approximated by a one-dimensional function $\tilde{K}$ that solely depends on the mean value of the input. That is, we assume there is a one-dimensional function $\tilde{K}$ such that $K_{\min}(\boldsymbol{\lambda}) \approx \tilde{K}(\bar{\boldsymbol{\lambda}})$, where $\bar{\boldsymbol{\lambda}}$ denotes the mean of $\boldsymbol{\lambda}$. Then, we only need to estimate this one-dimensional function $\tilde{K}$ through function interpolation. We note that one can always choose to directly interpolate the k-dimensional function $K_{\min}(\boldsymbol{\lambda})$ directly, but at the expense of increased sampling complexity and computational time.

B.2. PAC-Bayes Training with layerwise prior

Similar to Algorithm 1, our PAC-Bayes training with a layerwise prior is stated here in Algorithm 3.

B.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*

Algorithm 3 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 Sec6.*
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 (23) and Appendix B.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 (18)
    Compute γ as (6)
    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
$\hat{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta}$

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 $\mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\hat{\boldsymbol{\theta}}) = \mathcal{N}(\hat{\boldsymbol{\theta}}, \mathrm{diag}(\hat{\boldsymbol{\sigma}}))$, then the training loss expectation over the posterior is:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\theta} \sim \mathcal{Q}_{\hat{\sigma}}(\hat{\boldsymbol{\theta}})} \ell(f_{\boldsymbol{\theta}}; \mathcal{D}) &= \mathbb{E}_{\Delta\boldsymbol{\theta} \sim \mathcal{Q}_{\hat{\boldsymbol{\sigma}}}(\mathbf{0})} \ell(f_{\hat{\boldsymbol{\theta}} + \Delta\boldsymbol{\theta}}; \mathcal{D}) \\
&\approx \ell(f_{\hat{\boldsymbol{\theta}}}, \mathcal{D}) + \mathbb{E}_{\Delta\boldsymbol{\theta} \sim \mathcal{Q}_{\hat{\sigma}}(\mathbf{0})} (\ell(f_{\hat{\boldsymbol{\theta}}}; \mathcal{D}) \Delta\boldsymbol{\theta} + \frac{1}{2} \Delta\boldsymbol{\theta}^\top \nabla^2 \ell(f_{\hat{\boldsymbol{\theta}}}; \mathcal{D}) \Delta\boldsymbol{\theta}) \\
&= \ell(\hat{f_{\boldsymbol{\theta}}}; \mathcal{D}) + \frac{1}{2} \mathrm{Tr}(\mathrm{diag}(\hat{\boldsymbol{\sigma}}) \nabla^2 \ell(f_{\hat{\boldsymbol{\theta}}}; \mathcal{D})).
\end{aligned}$$

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.

B.4. Deterministic Prediction

Recall that for any $\boldsymbol{\theta} \in \mathbb{R}^d$ and $\boldsymbol{\sigma} \in \mathbb{R}^d_+$, we used $\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})$ to denote the multivariate normal distribition with mean $\boldsymbol{\theta}$ and covariance matrix $\text{diag}(\boldsymbol{\sigma})$. If we rewrite the left-hand side of the PAC-Bayes bound by Taylor expansion, we have:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{Q}_{\hat{\sigma}}(\hat{\boldsymbol{\theta}})}\ell(f_{\boldsymbol{\theta}};\mathcal{D}) &= \mathbb{E}_{\Delta\boldsymbol{\theta}\sim\mathcal{Q}_{\hat{\sigma}}(\mathbf{0})}\ell(f_{\hat{\boldsymbol{\theta}}+\Delta\boldsymbol{\theta}};\mathcal{D}) \\ &\approx \ell(f_{\hat{\boldsymbol{\theta}}},\mathcal{D}) + \mathbb{E}_{\Delta\boldsymbol{\theta}\sim\mathcal{Q}_{\hat{\sigma}}(\mathbf{0})}(\ell(f_{\hat{\boldsymbol{\theta}}};\mathcal{D})\Delta\boldsymbol{\theta} + \frac{1}{2}\Delta\boldsymbol{\theta}^\top\nabla^2\ell(f_{\hat{\boldsymbol{\theta}}};\mathcal{D})\Delta\boldsymbol{\theta}) \\ &= \ell(\hat{f_{\boldsymbol{\theta}}};\mathcal{D}) + \frac{1}{2}\text{Tr}(\text{diag}(\hat{\boldsymbol{\sigma}})\nabla^2\ell(f_{\hat{\boldsymbol{\theta}}};\mathcal{D})) \geq \ell(f_{\hat{\boldsymbol{\theta}}};\mathcal{D}).\end{aligned} \tag{24}$$

Recall here $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\sigma}}$ are the minimizers of the PAC-Bayes loss, obtained by solving the optimization problem (P). Equation (24) states that the deterministic predictor has a smaller prediction error than the Bayesian predictor. However, note that the last inequality in (24) is derived under the assumption that the term $\nabla^2\ell(\hat{f_{\boldsymbol{\theta}}};\mathcal{D})$ 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 assumption does not hold for all cases, the presented argument can only serve as an intuition that shows the potential benefits of using the deterministic predictor.

C. 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.

C.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 4.3. 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 value of $\mathbb{E}_{\mathbf{h}\sim\mathcal{P}}\mathbb{E}[\exp\left(\gamma(\mathbb{E}[X(\mathbf{h})] - X(\mathbf{h}))\right)]$ would be too large that may result in overflow issues. Therefore, we recommended γ_2 to be set to 10 or 20. If γ_2 is too large, we may also need more empirical samples to estimate K, which is more time-consuming. 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 4.7, 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.

Because of this, in the numerical experiments on GNN where the dataset is small, we clip the domain of λ at a lower bound of 0.1 at initialization. In training CNN13/CNN15, where the initialization is not as good as residual neural networks due to the lack of a batch norm layer or skip connection, we clip the domain of λ at $5e-3$ in training. This clipping is also necessary to ensure the derivative of the KL does not blow up, as λ is in the denominator of the KL-divergence, it cannot be too close to 0. In contrast, for neural networks that incorporate batch normalization, such as ResNet and DenseNet, no such clipping on λ is required.

C.2. Baseline PAC-Bayes bounds for unbounded loss functions

We compared two baseline PAC-Bayes bounds when training ResNet18 with our layerwise PAC-Bayes bound. The bounds are expressed in our notation.

- SubGaussian (Corollary 4 of Germain et al. (2016)):

$$\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})}\ell(f_{\boldsymbol{\theta}};\mathcal{D}) \leq \mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})}\ell(f_{\boldsymbol{\theta}};\mathcal{S}) + \frac{1}{m}(\log\frac{1}{\delta} + \mathrm{KL}(\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P})) + \frac{1}{2}s^2, \tag{25}$$

  where s^2 is the variance factor by assuming the loss function ℓ is sub-Gaussian as defined below:

$$\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{P}}\mathbb{E}_{\mathcal{S}\sim\mathcal{D}}\exp\left[\gamma(\ell(f_{\boldsymbol{\theta}};\mathcal{D}) - \ell(f_{\boldsymbol{\theta}};\mathcal{S}))\right] \leq \exp\left(\frac{\gamma^2 s^2}{2}\right), \forall\gamma\in\mathbb{R}^+.$$

  Theoretically, we need to evaluate s using all potential models sampled from prior to making sure the bound always holds in training, which is too difficult to achieve in real applications using deep neural networks. Given that s increases with the prior variance, we assessed s using the prior with the largest variance ($\boldsymbol{\lambda}_2 = e^2$) that we explored in our proposed training algorithm. This ensures that (25) is consistently maintained throughout our training.

- CGF (Theorem 9 of Rodríguez-Gálvez et al. (2023)):

$$\mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})}\ell(f_{\boldsymbol{\theta}};\mathcal{D}) \leq \mathbb{E}_{\boldsymbol{\theta}\sim\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})}\ell(f_{\boldsymbol{\theta}};\mathcal{S}) + \frac{1}{\gamma}\left((\log\frac{1}{\delta} + \mathrm{KL}(\mathcal{Q}_{\boldsymbol{\sigma}}(\boldsymbol{\theta})||\mathcal{P})) + \psi(\gamma)\right), \tag{26}$$

  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 $\psi(\gamma) = K\gamma^2$ and evaluated K using the prior with the largest variance ($\boldsymbol{\lambda}_2 = e^2$) that we explored in our proposed training algorithm to ensure (26) always holds. Moreover, γ is on the denominator of the bound, so we optimized γ when evaluating this bound and clipped γ to the same range $[0.5, 10)$ used in our algorithm.

C.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. The implementation is based on the GitHub repo Liu (2021). 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 2 and Figure 3.

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.

C.4. Compatibility with Data Augmentation

We didn't include data augmentation in most experiments for rigorousness considerations. 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 grid search is the same as in Appendix C.3. 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.

C.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 3. 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 10 and 11. 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.

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.

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 10 and 11.

C.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 & Welling, 2016) and SAGE (Hamilton et al., 2017). For GAT (Veličković et al., 2017), 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 6-9. 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.

C.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 6. 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 6, 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 6: 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**

C.8. Ablation study on image classification

We conducted an ablation study to showcase some extra benefits 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 7, 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.

Table 7: 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 8: Test accuracy of VGG13 trained with different learning rates.

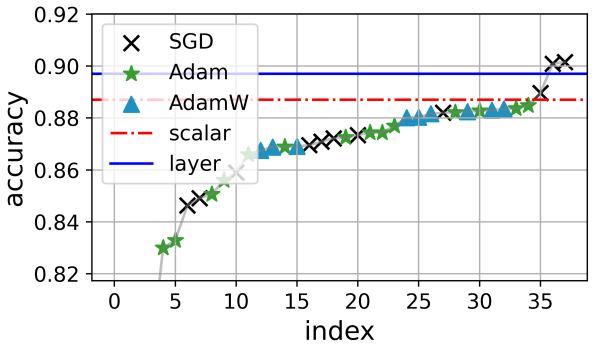
lr	$3e{-5}$	$5e{-5}$	$1e{-4}$	$2e{-4}$	$3e{-4}$	$5e{-4}$
CIFAR10	88.6	88.9	89.7	89.6	89.6	89.5
CIFAR100	67.7	68.0	67.1	-	-	-

Table 9: Test accuracy of ResNet18 trained with warmup epochs of σ.

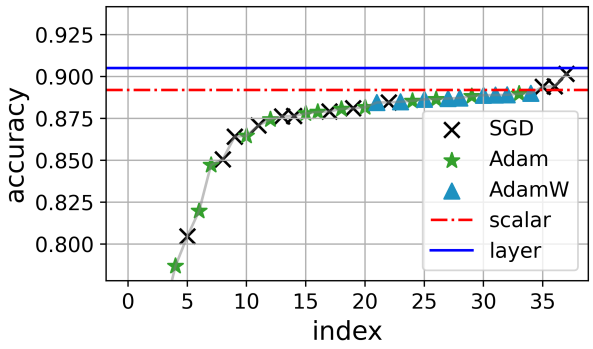
	10	20	50	80	100	150
CIFAR10	88.5	88.5	89.3	89.5	89.5	88.9
CIFAR100	69.4	69.6	68.9	69.1	69.0	68.1

As shown in Table 9, the test accuracy is insensitive to different numbers of warmup iterations.

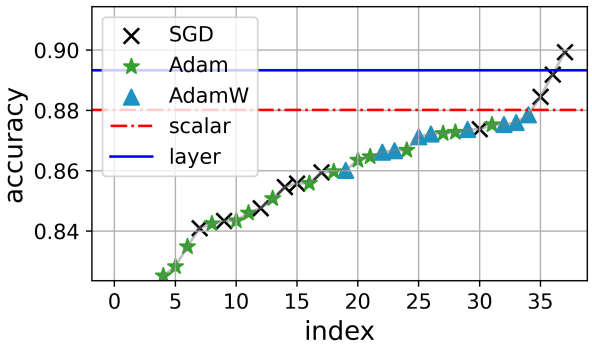
We also visualize the sorted test accuracy of baselines and our proposed PAC-Bayes training with large batch sizes and a fixed learning rate $5e{-4}$ in Figure 4 and Figure 5. These figures demonstrate that our PAC-Bayes training algorithm achieves better test accuracy than most searched settings. For 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.



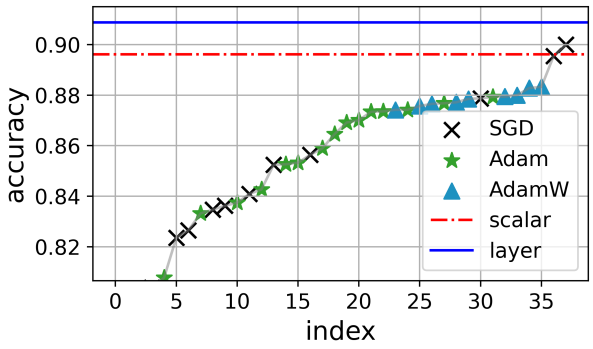
(a) VGG13



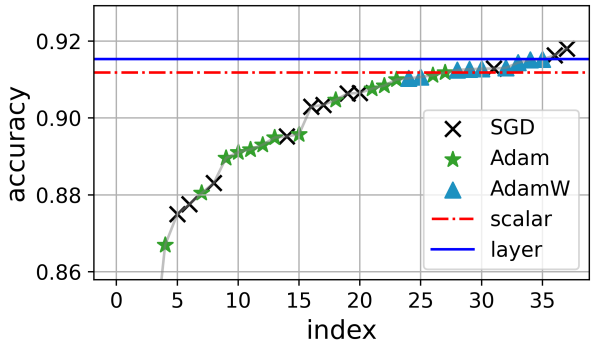
(b) VGG19



(c) ResNet18

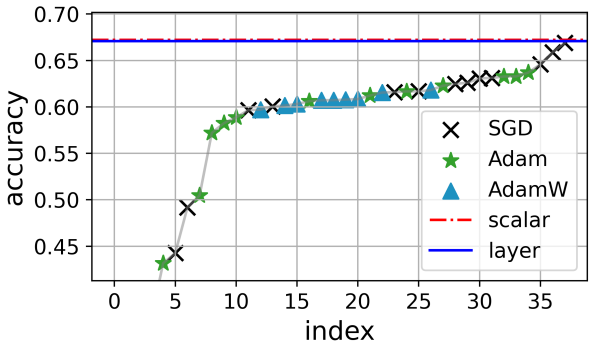


(d) ResNet34

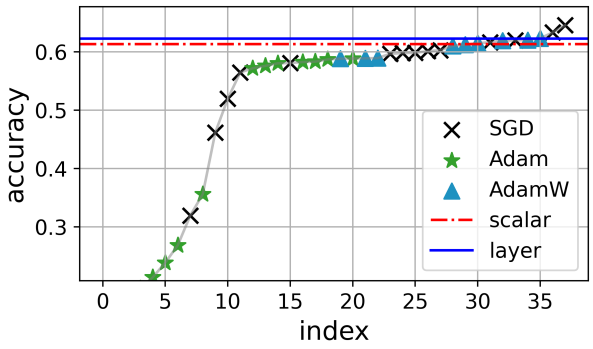


(e) Desnse121

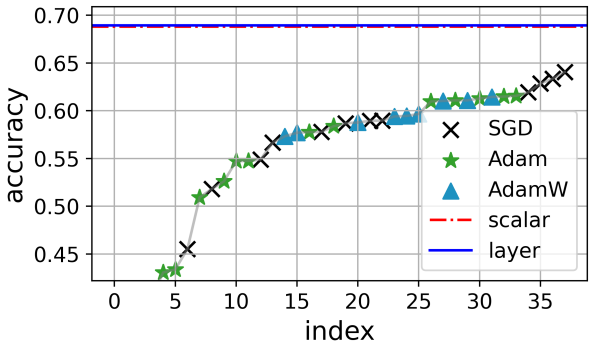
Figure 2: Sorted test accuracy of CIFAR10. The x-axis represents the experiment index.



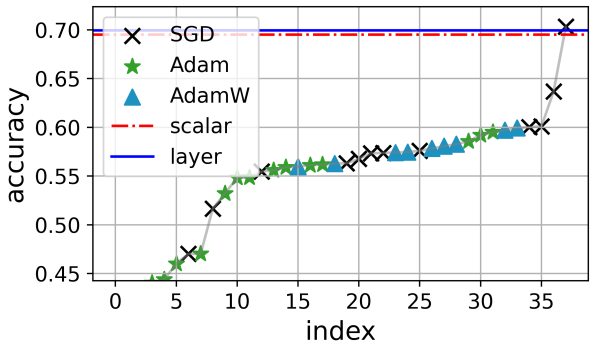
(a) VGG13



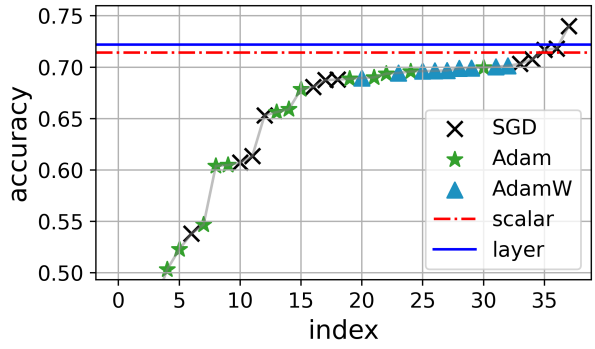
(b) VGG19



(c) ResNet18

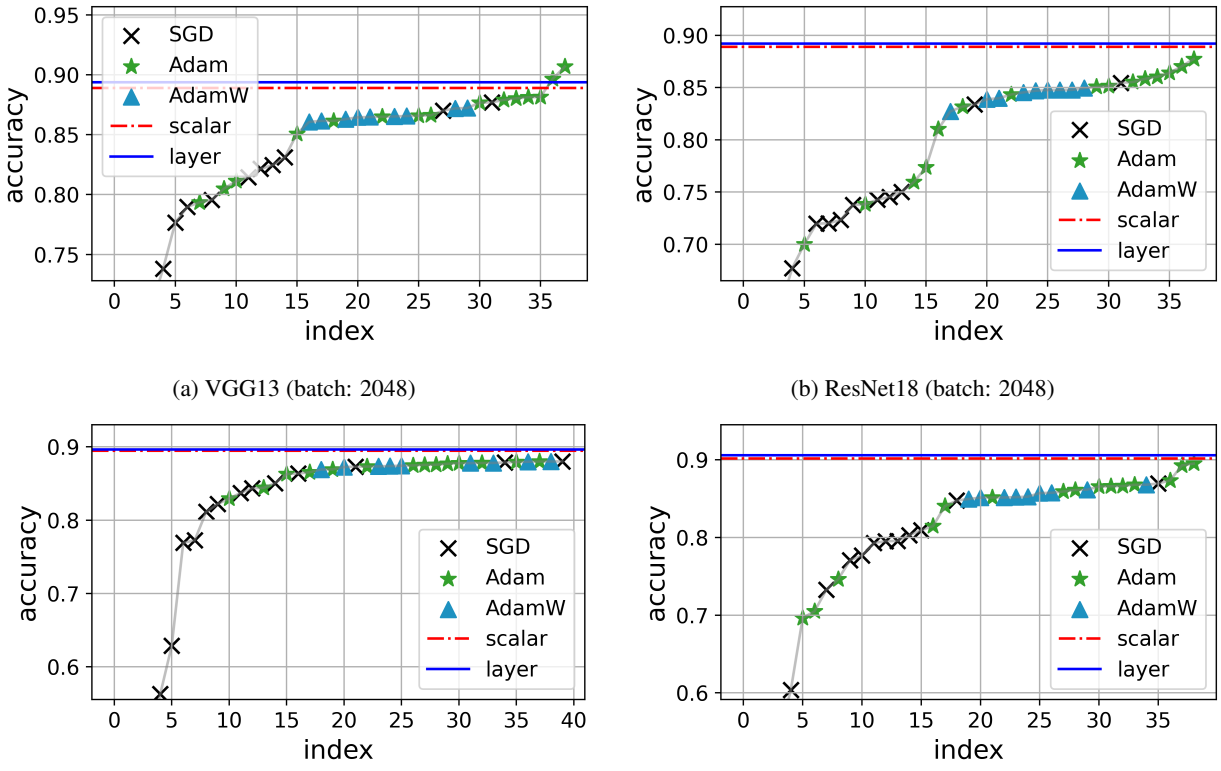


(d) ResNet34



(e) Desnse121

Figure 3: Sorted test accuracy of CIFAR100. The x-axis represents the experiment index.



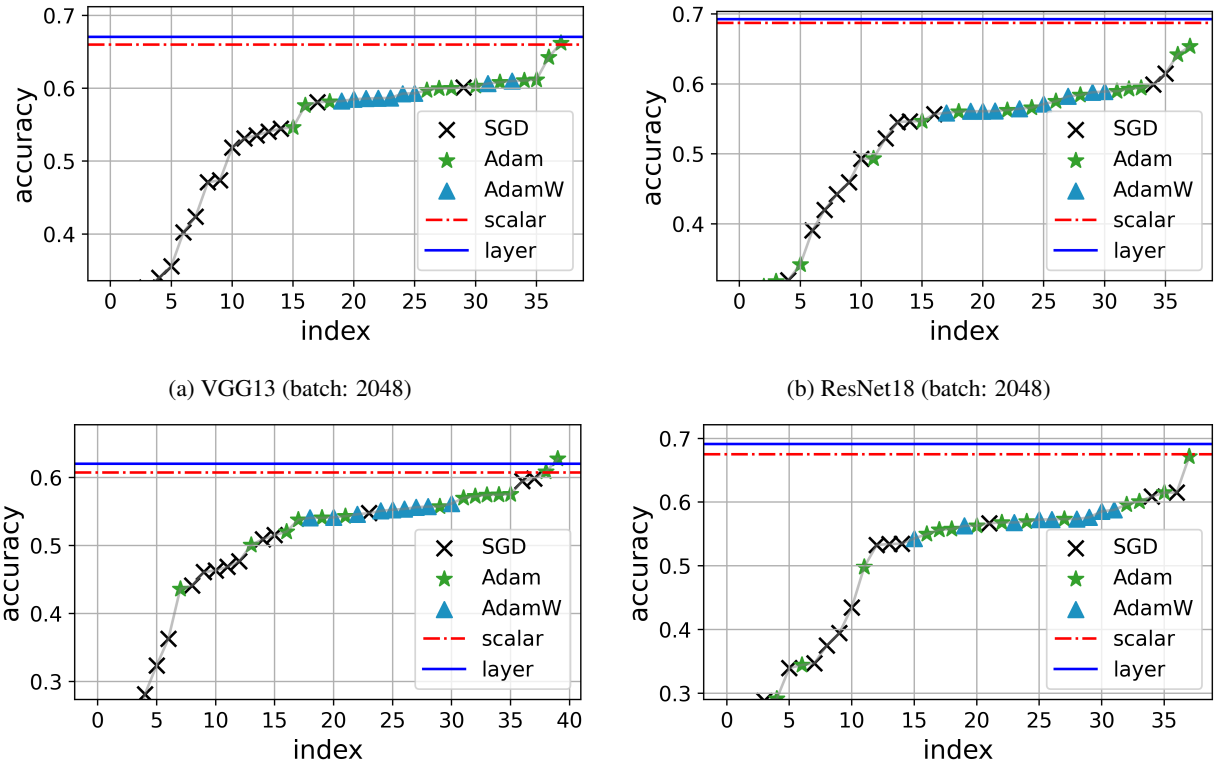
(a) VGG13 (batch: 2048)

(b) ResNet18 (batch: 2048)

(c) VGG19 (batch: 1280)

(d) ResNet34 (batch: 1280)

Figure 4: Sorted test accuracy of CIFAR10 with large batch sizes. The x-axis represents the experiment index.



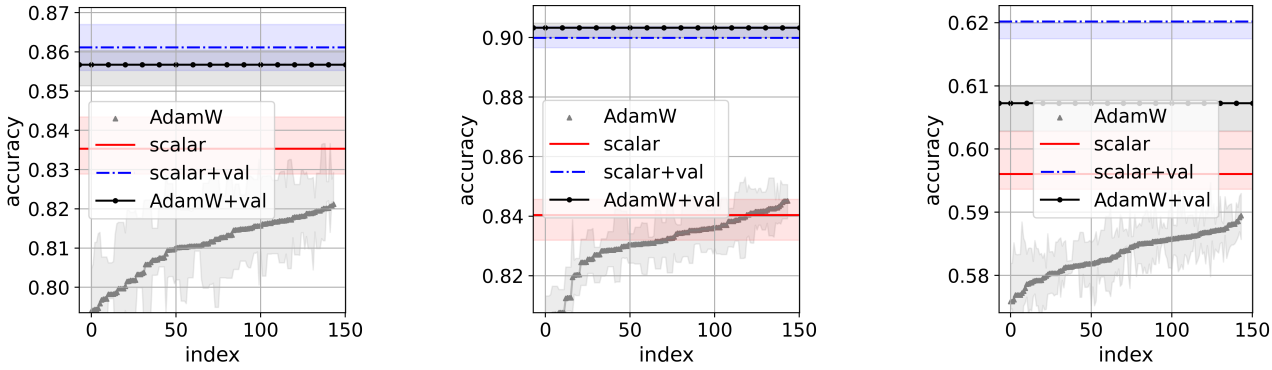
(a) VGG13 (batch: 2048)

(b) ResNet18 (batch: 2048)

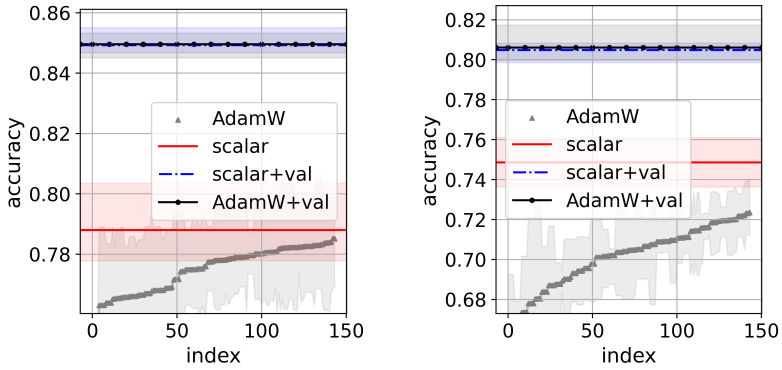
(c) VGG19 (batch: 1280)

(d) ResNet34 (batch: 1280)

Figure 5: Sorted test accuracy of CIFAR100 with large batch sizes. The x-axis represents the experiment index.

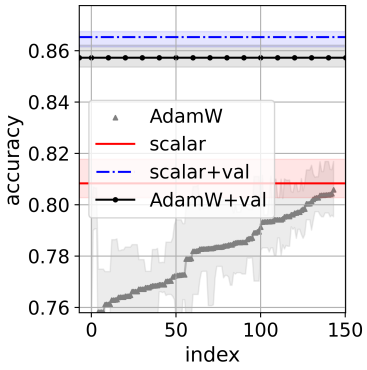


(a) CoraML (b) Citeseer (c) CoraFull

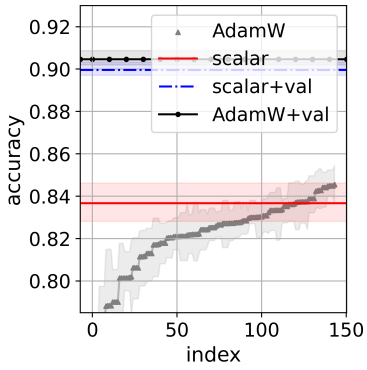


(d) DBLP (e) DBLP

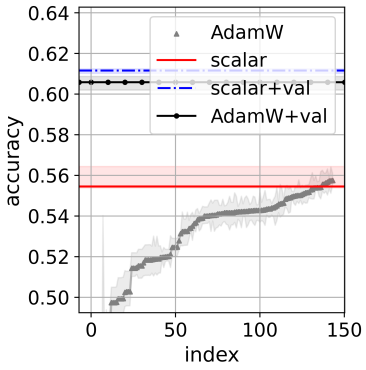
Figure 6: Test accuracy of GCN. The first and third quartiles construct the interval over the ten random splits. {+val} denotes the performance with both training and validation datasets for training.



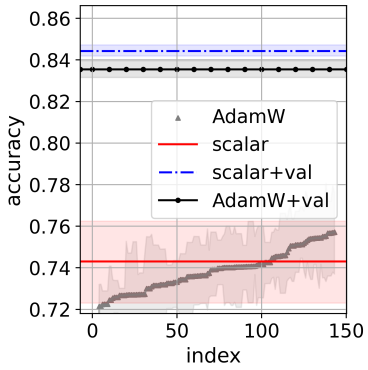
(a) CoraML



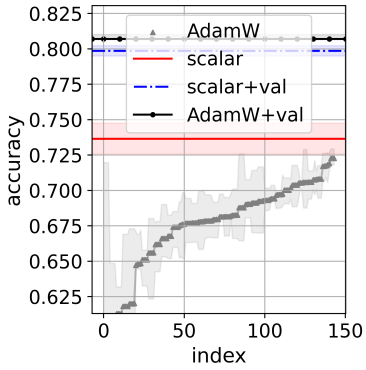
(b) Citeseer



(c) CoraFull

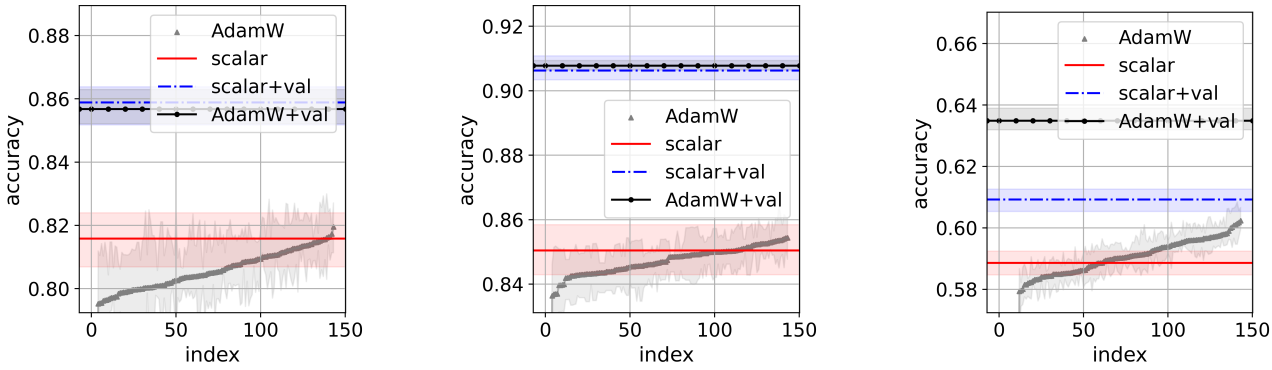


(d) DBLP

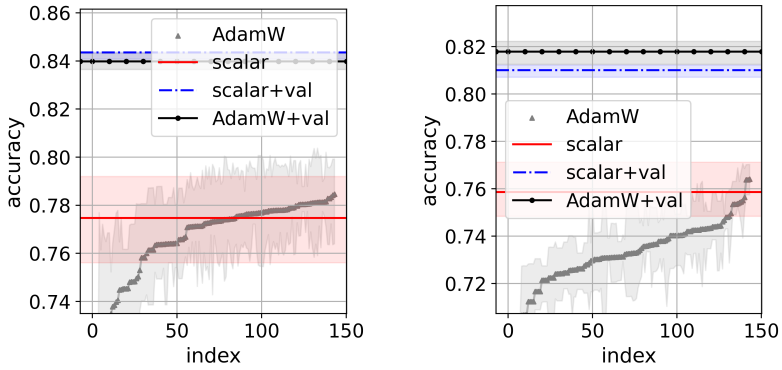


(e) DBLP

Figure 7: 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.

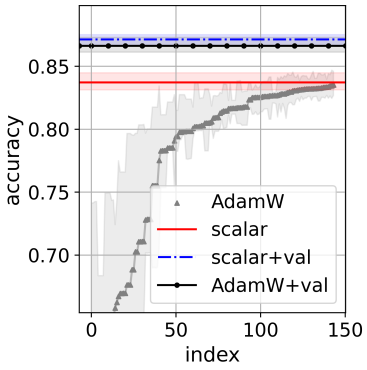


(a) CoraML (b) Citeseer (c) CoraFull

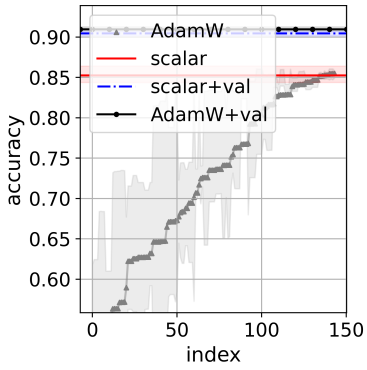


(d) DBLP (e) DBLP

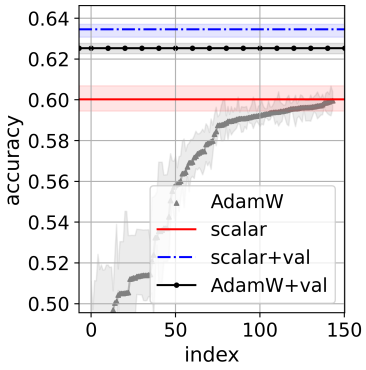
Figure 8: Test accuracy of GAT. The first and third quartiles construct the interval over the ten random splits. {+val} denotes the performance with both training and validation datasets for training.



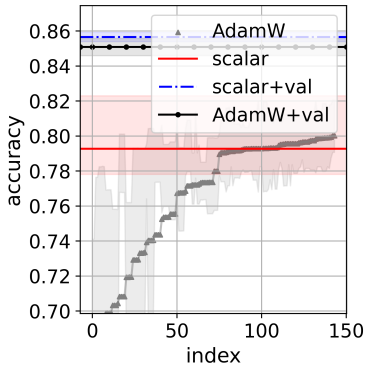
(a) CoraML



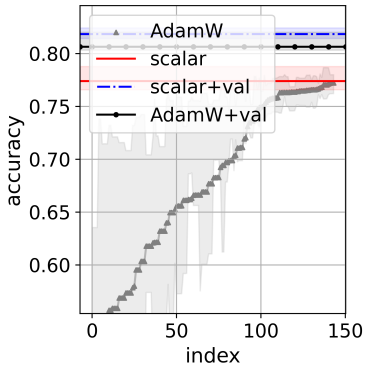
(b) Citeseer



(c) CoraFull

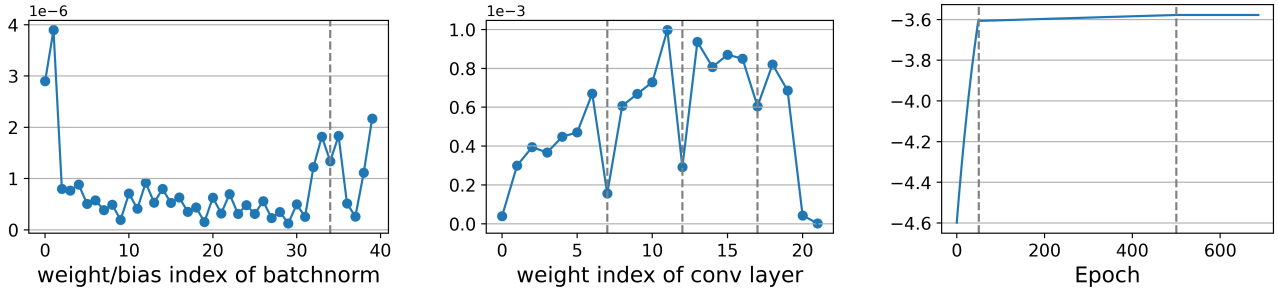


(d) DBLP

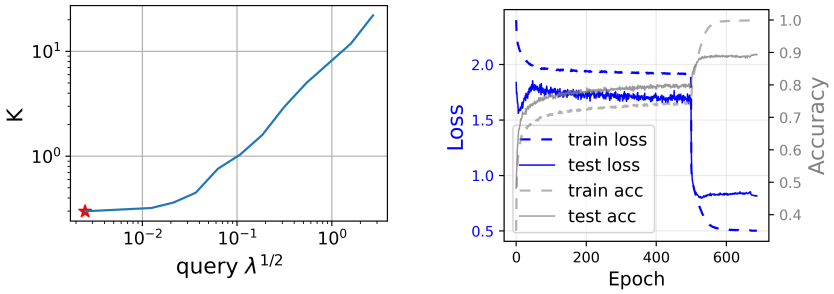


(e) DBLP

Figure 9: 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.

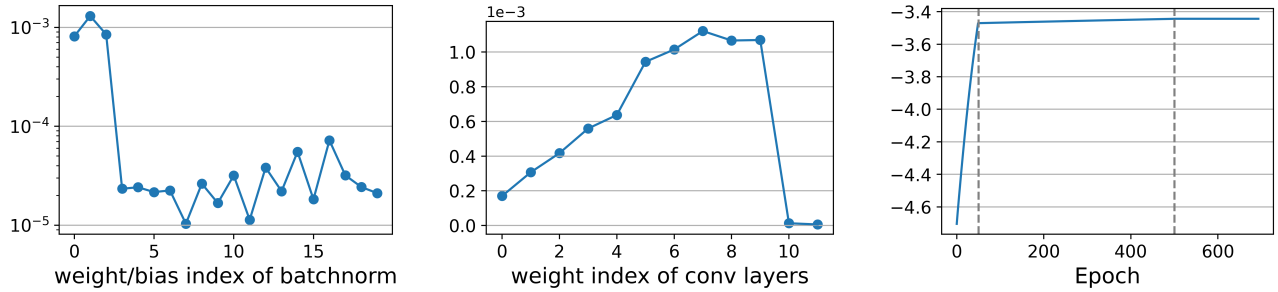


(a) mean($\boldsymbol{\sigma}$) of batch-norm layers. (b) mean($\boldsymbol{\sigma}$) of convolution layers. (c) mean($\boldsymbol{\sigma}$) in training.

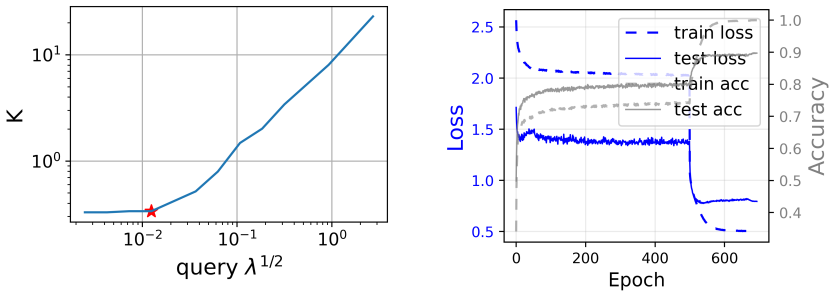


(d) Function $\tilde{K}(\bar{\boldsymbol{\lambda}})$. (e) Training and testing process.

Figure 10: 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 11: Training details of VGG13 on CIFAR10. The red star denotes the final K.