Spatial-Temporal-Fusion BNN: Variational Bayesian Feature Layer

Shiye Lei* Zhuozhuo Tu* Leszek Rutkowski† Feng Zhou‡ Li Shen§
Fengxiang He§ Dacheng Tao§

Abstract

Bayesian neural networks (BNNs) have become a principal approach to alleviate overconfident predictions in deep learning, but they often suffer from scaling issues due to a large number of distribution parameters. In this paper, we discover that the first layer of a deep network possesses multiple disparate optima when solely retrained. This indicates a large posterior variance when the first layer is altered by a Bayesian layer, which motivates us to design a spatial-temporal-fusion BNN (STF-BNN) for efficiently scaling BNNs to large models: (1) first normally train a neural network from scratch to realize fast training; and (2) the first layer is converted to Bayesian and inferred by employing stochastic variational inference, while other layers are fixed. Compared to vanilla BNNs, our approach can greatly reduce the training time and the number of parameters, which contributes to scale BNNs efficiently. We further provide theoretical guarantees on the generalizability and the capability of mitigating overconfidence of STF-BNN. Comprehensive experiments demonstrate that STF-BNN (1) achieves the state-of-the-art performance on prediction and uncertainty quantification; (2) significantly improves adversarial robustness and privacy preservation; and (3) considerably reduces training time and memory costs.

1 Introduction

Neural networks often have significant uncertainty on their parameters in optimization [52]. The point estimation of optimization to the parameters usually contributes to overconfident predictions [24], which considerably undermines the trust of deep learning, especially in security-critical application domains, like medical diagnosis [54] and autonomous driving [29]. Bayesian neural networks (BNNs) are the main approaches to mitigate overconfidence by applying interval estimation. However, due to the tremendous number of trainable distribution parameters in BNNs, they experience issues in scaling to large-scale models and data, including prohibitively high convergence time, memory cost, application inflexibility, and training instability.

In this work, we explore the properties of solely retraining a single layer of the pretrained network, and the layer stability is defined by calculating the cosine similarity between parameters of the layer in the retrained net and in the pretrained net. Empirical studies on FashionMNIST and CIFAR-10 show that parameters of the

*S. Lei and Z. Tu are associated with School of Computer Science, Faculty of Engineering, the University of Sydney, Darlington NSW 2008, Australia and JD Explore Academy, JD.com Inc., Beijing 100176, China. This work was completed when they were interns at JD Explore Academy. Email: slei5230@uni.sydney.edu.au and zhtu3055@uni.sydney.edu.au.

†L. Rutkowski is with the Department of Artificial Intelligence, University of Social Sciences, 90-113, Łódź, Poland, and also with the Systems Research Institute of the Polish Academy of Sciences, 01-447 Warsaw, Poland. Email: lrutkowski@san.edu.pl.

‡F. Zhou is associated with Department of Computer Science and Technology, BNRist Center, THU-Bosch Joint ML Center, Tsinghua University, Beijing 100086, China. Email: zhoufeng6288@tsinghua.edu.cn.

§L. Shen, F. He, and D. Tao are associated with JD Explore Academy, JD.com Inc., Beijing 100176, China. Email: shenli100@jd.com, hefengxiang@jd.com, and taodacheng@jd.com.

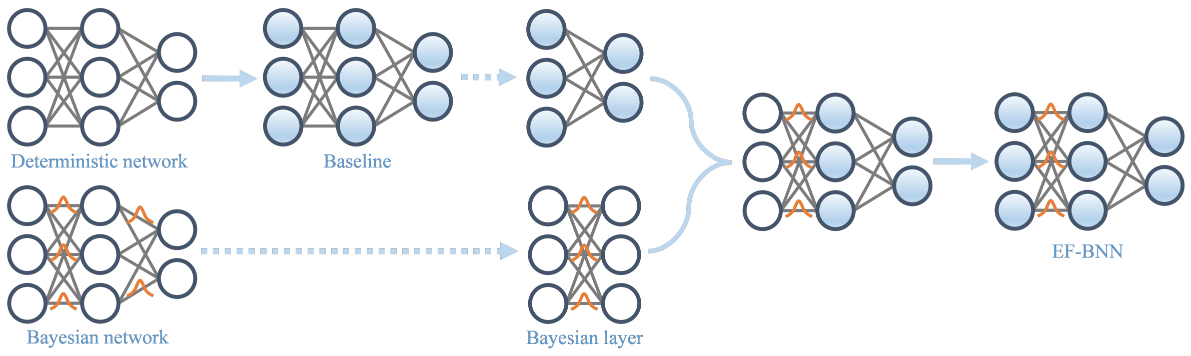


Figure 1: Spatial-temporal-fusion BNN architecture. The lines between circles denote the network parameters, and Bayesian layers are marked by orange bell curves. Hollow circles are colored by training, which is denoted by the arrows with blue solid lines.

last layer usually converge to similar results, while parameters of the first layer exhibit much lower stability, *i.e.*, more diverse convergence results after the retraining. The low layer stability in the first layer indicates that there exists multiple optimum with large discrepancy when the layer is retrained. Therefore, if the first layer with low stability is converted to Bayesian, the posterior distribution of the Bayesian layer tends to cover these dissimilar optimum, which induces a large posterior variance and helps alleviate overconfidence.

Inspired by the findings in the experiments *w.r.t.* layer stability, we propose the *spatial-temporal-fusion BNN* (STF-BNN) that realizes the fusion of optimization and Bayesian inference for training neural networks from both "spatial" and "temporal" aspects, and the schematic diagram of the STF-BNN is presented in Figure 1. In the first training phase, we employ optimization methods to train the whole neural network to reach fast and memory-efficient learning. During the second training phase, we employ Bayesian inference to train the first layer, while all other layers are frozen. Due to the tremendous decrease in the number of distribution parameters compared to vanilla BNNs, Bayesian inference achieves fast convergence in this phase. Then, we theoretically prove the generalization ability and the capability of mitigating overconfidence of our STF-BNN.

Comprehensive experiments of two standard neural network architectures, VGG-19 and Wide ResNet-28-10, are conducted on four benchmark datasets, CIFAR-10, CIFAR-100, CIFAR-10-C, and CIFAR-100-C, to evaluate STF-BNNs from three aspects: uncertainty quantification, privacy protection, and adversarial robustness. Empirical results show that STF-BNNs can efficiently improve performance in these aspects. Besides, the performance of STF-BNNs in uncertainty quantification is on-par with other state-of-the-art approaches, while significantly reducing training time and memory costs.

In sum, the main contributions are as follows:

- For the first time in literature we reveal the large disparity of layer stability between different layers in deep neural networks, which prompts us to design the STF-BNN;
- Based on spatial and temporal fusions, we propose the STF-BNN, which can be trained fast and also reduces the memory costs;
- We theoretically prove the generalization performance and the capability of mitigating overconfidence of STF-BNN;
- Comprehensive experiments are conducted on STF-BNNs and show its superior performance on uncertainty quantification, privacy preservation, and adversarial robustness.

2 Related Works

Bayesian neural networks. From the Bayesian perspective, the parameters of neural networks obey a specific distribution, which terms the posterior distribution and is approximated by Bayesian inference. A famous Bayesian inference approach is Markov chain Monte Carlo (MCMC) [15, 20, 50], which constructs a Markov chain that performs a Monte Carlo sampler to approximate the posterior. To leverage MCMC on training deep Bayesian neural networks, stochastic gradient MCMC (SGMCMC) employs stochastic gradients to speed up and scale MCMC [9], such as Stochastic Gradient Langevin Dynamic (SGLD) [60], Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) [6], Stochastic Gradient Fisher Scoring (SGFS) [2], Cyclical SGMCMC [70], and a summary [40]. Another major Bayesian inference method is variational inference (VI) [3], which casts inference as an optimization problem and aims to maximize the evidence lower bound (ELBO). Stochastic variational inference (SVI) [17, 27, 33] also adopts stochastic gradients to speed up and scale VI. Bayes by backprop [4] and Radial BNNs [13] provide a practical solution in implementations. Khan et al. [30] further speed up SVI based on Adam [31]. Krishnan et al. [34] and Deng et al. [8] also utilize the weights of pretrained neural networks as the initial posterior mean in training BNNs to improve the convergence performance.

Uncertainty quantification. Considering the overconfident predictions given by deep neural networks [18], quantifying the uncertainty of model predictions is crucial for improving the explainability and reliability of networks. Some metrics have been proposed for uncertainty quantification [19, 28], such as expected calibration error (ECE) that measures the difference between the prediction accuracy and the predictive confidence score produced by networks [18]. Many approaches are also proposed to better estimate the prediction uncertainty by designing novel architectures, loss functions, and training strategies based on deterministic networks. SDE-Net [32] is consisted of a drift net and a diffusion net to fit the training examples and produce the uncertainty, respectively. Van Amersfoort et al. [59] propose a novel loss function that inspects the centroid of data to improve the capability of uncertainty quantification. Liu et al. [38] also achieve this by adding a weight normalization step during training and replacing the output layer with a Gaussian Process. Except for the approaches that employ deterministic neural networks for estimating the prediction uncertainty via single forward propagation, there are many ensemble methods, such as deep ensemble [37] and Batch ensemble [61], which also designed for uncertainty quantification. Because their parameters are equipped with distributions, BNNs have gradually become dominant approaches in quantifying the uncertainty of neural networks [10, 14] in recent years. Kristiadi et al. [35] prove that equipping the last linear classifier with a Bayesian layer can mitigate the overconfidence of ReLU networks, while our approach introduces the Bayesian layer into the first non-linear layer with two-phase training for more efficient training.

Privacy protection and adversarial robustness. Privacy and security concerns in deep learning are rising as deep learning has been applied to increasingly sensitive domains [11, 12, 43]. Since neural networks easily "memorize" the training data, attackers can infer the sensitive information contained in the training data by inspecting the well-trained networks [55]. For example, a membership inference (MI) attack aims to infer whether an example is in the training data or not. Some works have also proposed to mitigate the privacy leakage in deep learning models through injecting noise into gradients during the training process [1, 23, 47, 63].

In addition to the risk of privacy leakage, neural networks have also been reported to be vulnerable to adversarial attacks [43, 66]: networks sometimes output completely different predictions when the input images are slightly modified. The adversarial attack is firstly probed by Goodfellow et al. [16], which propose the fast gradient sign method (FGSM) to generate adversarial examples by single gradient ascent. Madry et al. [43] leverage the projection gradient descent (PGD) on the adversarial attack. Compared to FGSM, PGD generates higher-quality adversarial examples via multiple iterations of gradient ascent. Carlini and Wagner

[5] develop three adversarial attacks, which make the perturbations quasi-imperceptible by restricting their l_0, l_2, and l_∞ norms. A common practice for improving adversarial robustness is adversarial training [53] which applies a minimax approach to robust neural networks. Some methods also modify optimization [64] and network architectures [39] to improve the robustness. However, Zhang et al. [68] show the trade-off between adversarial robustness and prediction accuracy and claim that adversarial training leads to underfitting on the normal sample and hurts model accuracy. Furthermore, they mitigate the trade-off between robustness and accuracy by disentangling the robust error into natural error and boundary error. Zhang et al. [69] also alleviate the trade-off by assigning small weights to non-important samples for saving model capacity and developing a reweighting-based framework that improves robustness while preserving accuracy.

3 Preliminaries

We denote the training set by $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^n$, n is the dimension of input data, $y_i \in \{1, \ldots, k\}$, k is the number of classes, and $m = |\mathcal{S}|$ is the training sample size. We assume that $(\mathbf{x}_i, y_i)$ are independent and identically distributed (i.i.d.) random variables drawn from the data generating distribution $\mathcal{D}$. Denote the ReLU neural network as $f_{\boldsymbol{\theta}}(\mathbf{x}) = W_d \phi\left(W_{d-1} \phi\left(\cdots \phi\left(W_1 \mathbf{x}\right)\right)\right) : \mathbb{R}^n \to \mathbb{R}^k$, which is a d layer feed-forward network parameterized by $\boldsymbol{\theta} = \operatorname{vec}(\{W_i\}_{i=1}^d)$, here ϕ denotes the ReLU activation function. The logit $f_{\boldsymbol{\theta}}(\mathbf{x})$ is normalized by the non-linear Softmax function $o(f_{\boldsymbol{\theta}}(\mathbf{x})) = \operatorname{softmax}(f_{\boldsymbol{\theta}}(\mathbf{x}))$, and $o(f_{\boldsymbol{\theta}}(\mathbf{x}))$ is assumed to be a discrete probability density function. Let $o^{(i)}(f_{\boldsymbol{\theta}}(\mathbf{x}))$ be the i-th component of $o(f_{\boldsymbol{\theta}}(\mathbf{x}))$, hence $\sum_{i=1}^k o^{(i)}(f_{\boldsymbol{\theta}}(\mathbf{x})) = 1$. We define $\hat{p}(\mathbf{x}) = o^{(y)}(f_{\boldsymbol{\theta}}(\mathbf{x}))$ and $\hat{y}(\mathbf{x}) = \arg\max_i o^{(i)}(f_{\boldsymbol{\theta}}(\mathbf{x}))$ to denote the confidence score and the predicted labels by the network $f_{\boldsymbol{\theta}}$ on $\mathbf{x}$, respectively.

For the classical setting that $f_{\boldsymbol{\theta}}$ is a deterministic neural network, a maximum likelihood estimation $\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \log \Pr\left[\hat{y}(\mathbf{x}_i) = y_i | (\mathbf{x}_i, y_i) \in \mathcal{S}\right]$ is returned by optimizing $f_{\boldsymbol{\theta}}$ on $\mathcal{S}$ in practice. However, due to the randomness of the learning algorithm, let $\mathbb{Q}(\boldsymbol{\theta})$ denote the posterior distribution returned by the training algorithm leveraged on $\mathcal{S}$, and $\boldsymbol{\theta}_{\text{MLE}} \sim \mathbb{Q}(\boldsymbol{\theta})$.

If $f_{\boldsymbol{\theta}}$ is a Bayesian neural network, the posterior $\mathbb{Q}(\boldsymbol{\theta})$ is often approximated by variational inference, which aims to maximize the evidence lower bound (ELBO):

$$\text{ELBO} = \mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{Q}(\boldsymbol{\theta})} \left(\log \mathbb{P}\left(\mathcal{S} | \boldsymbol{\theta}\right)\right) - \text{KL}\left(\mathbb{Q} \| \mathbb{P}\right), \tag{1}$$

where the likelihood $\log \mathbb{P}\left(\mathcal{S}|\boldsymbol{\theta}\right) \approx \sum_{i=1}^m \log o^{(y_i)}(\mathbf{x}_i)$ and $\text{KL}\left(\mathbb{Q}\|\mathbb{P}\right)$ is the KL-divergence between the approximate posterior $\mathbb{Q}(\boldsymbol{\theta})$ and the prior $\mathbb{P}(\boldsymbol{\theta})$:

$$\text{KL}(\mathbb{Q}\|\mathbb{P}) = \mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{Q}} \left(\log \frac{\mathbb{Q}(\boldsymbol{\theta})}{\mathbb{P}(\boldsymbol{\theta})}\right). \tag{2}$$

In this paper, we adopt Gaussian distribution as the approximate posterior, *i.e.*, $\mathbb{Q}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\Sigma}$ is the covariance matrix.

To estimate the probability of the output for a BNN by the posterior mean and variance of its parameters, we introduce Lemma 1 as follows.

Lemma 1 (MacKay [41])**.** *For a binary classification BNN $f_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}$, where $\boldsymbol{\theta}$ is the parameters of $f_{\boldsymbol{\theta}}$ and is sampled from the Gaussian distribution $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and $\mathbf{d}(\mathbf{x}) := \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})|_{\boldsymbol{\theta}=\boldsymbol{\mu}}$. The sigmoid function is defined as $\tau(z) = 1/(1 + \exp(-z))$ for $z \in \mathbb{R}$. Then we have that*

$$\Pr[y = 1|\mathbf{x}] \approx \tau(z(\mathbf{x})), \tag{3}$$

where

$$z(\mathbf{x}) := \frac{f_{\boldsymbol{\mu}}(\mathbf{x})}{\sqrt{1+\pi/8\mathbf{d}^\top\boldsymbol{\Sigma}\mathbf{d}}}. \tag{4}$$

The approximately equal in Eq. 3 comes from the first-order Taylor expansion of $f_{\boldsymbol{\theta}}$ at $\boldsymbol{\mu}$ [42]: $f_{\boldsymbol{\theta}}(\mathbf{x}) \approx f_{\boldsymbol{\mu}}(\mathbf{x}) + \mathbf{d}(\mathbf{x})^\top(\boldsymbol{\theta}-\boldsymbol{\mu})$. In Lemma 1, $z(\mathbf{x})$ can be regarded as the logit, and $\tau(z(\mathbf{x}))$ is the probability or the confidence score for the binary BNN.

4 Layer Stability

In this section, we explore the parameter stability of different layers in neural networks, which sheds light on designing following spatial-temporal-fusion BNN. In detail, a neural network f_{pre} is first trained on the training set $\mathcal{S}$. Then, the k-th layer of f_{pre} is initialized and retrained on $\mathcal{S}$ while the parameters of other layers are fixed until the training process has converged, so we get the retrained network $f_{re\text{-}k}$. With $f_{re\text{-}k}$, the stability of the k-th layer of f_{pre} can be defined as below.

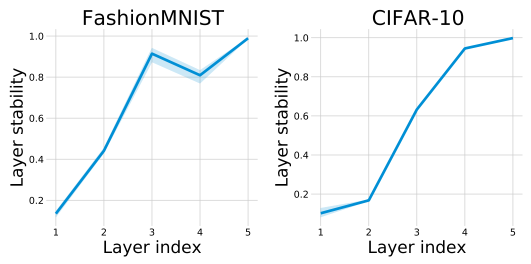


Figure 2: Layer stability as a function of layer index on FashionMNIST and CIFAR-10; the darker lines show the average over five seeds and the shaded area shows the standard deviations.

Definition 1 (Layer stability)**.** *Let $W, W' \in \mathbb{R}^{a\times b}$ denote the weight matrix of the k-th layer of the pretrained network f_{pre} and the retrained network $f_{re\text{-}k}$, respectively. a and b are the dimensions of the output and the input of the k-the layer. Then, the stability of the k-th layer of f_{pre} is defined as*

$$\frac{1}{a}\sum_{i=1}^{a}\frac{|\mathbf{w}_i^\top\mathbf{w}_i'|}{\|\mathbf{w}_i\|\|\mathbf{w}_i'\|}, \tag{5}$$

where $\mathbf{w}_i, \mathbf{w}_i' \in \mathbb{R}^b$ denote the i-th row vector of W_i and W_i', respectively, for $i \in [1, a]$.

We discard the Euclidean distance $|\mathbf{w}_i - \mathbf{w}_i'|$ in the definition because ReLU networks possess the scale-invariant property, which states that the output of the ReLU network is invariant when the parameters of the layer are multiplied by a positive constant c and the logits are divided by c. Therefore, scaling parameters of a single layer has no effect on final predictions, and thus a series of scaled weights should be treated equally. The cosine similarity is employed to measure the layer stability because it is non-sensitive to this scaling. According to Definition 1, smaller layer stability indicates $\mathbf{w}$ and $\mathbf{w}'$ become more orthogonal, and thus they employ a more dissimilar ratio of the previous output to shape new features. It is worth noting that the definition of layer stability is different from the co-adaptation [7, 26], which measures the correlations between activations, while the layer stability focuses on the network parameters.

To measure the layer stability, we first trained 5 five-hidden-layer MLPs, *i.e.*, f_{pre} on FashionMNIST [62] and CIFAR-10 [36], respectively, and $f_{re\text{-}k}$ are subsequently obtained by retraining the specific layer of f_{pre} on the same training set. Then, according to Eq. 5, we can compute the k-th layer's stability by comparing the parameters of the k-th layer in f_{pre} and in $f_{re\text{-}k}$, and the stability of different layers is plotted in Figure 2. From the figure, we have two observations: (1) the first layer has the lowest stability compared to the subsequent layers; and (2) the stability of the last layer is close to one.

The experiment reveals the large disparity of layer stability among different layers in deep neural networks.

Moreover, low layer stability shows that there exists multiple optima with large discrepancies when the layer is retrained. Therefore, if we convert the layers with low layer stability into Bayesian layers, the posterior distribution of Bayesian layers tends to cover these dissimilar optima, which induces a large posterior variance and hence helps mitigate overconfidence.

5 Methods

Inspired by the finding that solely retraining the first layer of the pretrained networks induces low layer stability, we construct the spatial-temporal-fusion BNN by making the first layer as Bayesian to efficiently mitigate overconfidence with faster training and less memory cost.

5.1 Spatial-Temporal Fusion

Spatial-temporal-fusion BNNs (STF-BNNs) realize the fusion from both spatial and temporal aspects. From the spatial perspective, STF-BNNs fuse the conventional deterministic neural networks with the Bayesian layer, which allows the network to capture the parameter uncertainty and also substantially reduce the number of distribution parameters compared to vanilla BNNs. From the temporal perspective, the fusion of optimization and Bayesian inference makes the training of STF-BNNs more efficient. The training procedure of STF-BNNs can be divided into two phases as follows.

During the first training phase, STF-BNNs employ optimization methods, such as stochastic gradient descent (SGD), to train the whole network $f_{\boldsymbol{\theta}}$ to $f_{\boldsymbol{\theta}_{\mathrm{MLE}}}$. In this way, one can achieve fast training of the network.

During the second training phase, STF-BNNs adopt Bayesian inference to train the first layer of the network $f_{\boldsymbol{\theta}_{\mathrm{MLE}}}$, in order to quantify uncertainty for improving the reliability in a relatively cost. The parameter $\boldsymbol{\theta}_{\mathrm{MLE}}$ is partitioned into $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$: $\boldsymbol{\theta}_1$ denotes the parameters of the first layer, and $\boldsymbol{\theta}_2$ is the parameters of the subsequent layers. Then we parameterize the first layer with the posterior mean and variance of weights by $\boldsymbol{\theta}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, and Bayesian inference is employed to infer the approximate posterior $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ while $\boldsymbol{\theta}_2$ is fixed. In this training phase, *Bayes by backprop* [4] is employed for practical implementation. The entire process of training an STF-BNN is presented in Algorithm 1, and we employ the trainable parameter $\boldsymbol{\rho}$ to generate the covariance matrix $\boldsymbol{\Sigma}_1$ for guaranteeing the positive variance.

5.2 Generalization Bound of STF-BNN

In this section, we explore the generalizability [21, 48] for the STF-BNN based on the PAC-Bayes generalization bound [46].

The parameters $\boldsymbol{\theta}$ of the STF-BNN are partitioned into $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ according to whether they are updated in Bayesian inference or optimization, respectively. For $\boldsymbol{\theta}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, we assume the approximate posterior $\mathbb{Q}(\boldsymbol{\theta}_1) = \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ obeys mean-field assumption, which implies $\boldsymbol{\Sigma}_1$ is a diagonal matrix. On the other hand, $\boldsymbol{\theta}_2$ is optimized by SGD. Without loss of generality, we assume $\mathbb{E}_{\boldsymbol{\theta}_2 \sim \mathbb{Q}(\boldsymbol{\theta}_2)}[\boldsymbol{\theta}_2] = \mathbf{0}$. Then, $\boldsymbol{\theta}_2$ has an analytic stationary distribution according to Mandt et al. [44]:

$$\mathbb{Q}(\boldsymbol{\theta}_2) = \frac{1}{\sqrt{2\pi \det(\boldsymbol{\Sigma}_2)}} \exp\left\{ -\frac{1}{2} \boldsymbol{\theta}_2^\top \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\theta}_2 \right\}. \tag{10}$$

$\boldsymbol{\Sigma}_2 = \mathrm{cov}(\boldsymbol{\theta}_2, \boldsymbol{\theta}_2)$ is the covariance matrix of $\boldsymbol{\theta}_2$, and $\boldsymbol{\Sigma}_{12} = \mathrm{cov}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ denotes the covariance matrix between $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$.

```
Algorithm 1: Training procedure of the STF-BNN
Optimize a deterministic neural network $f_{\boldsymbol{\theta}_{\mathrm{MLE}}}$;
Partition $\boldsymbol{\theta}_{\mathrm{MLE}}$ into $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, and reinitialize $\boldsymbol{\theta}_1$ with variational parameters $\boldsymbol{\gamma} = (\boldsymbol{\mu}_1, \boldsymbol{\rho})$;
for epoch in num_epochs do
    Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ and $\boldsymbol{\theta}_1$:
        $\boldsymbol{\Sigma}_1 = \log(1 + \exp(\boldsymbol{\rho}))$
        $\boldsymbol{\theta}_1 = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_1 \circ \boldsymbol{\epsilon};$          (6)
    Compute $-$ ELBO $l(\boldsymbol{\theta}_1, \boldsymbol{\gamma})$ with batch size $M$:
        $l(\boldsymbol{\theta}_1, \boldsymbol{\gamma}) = \sum_{i=1}^{M} \mathcal{L}(f(\mathbf{x}_i|\boldsymbol{\theta}_1, \boldsymbol{\gamma}), y_i) + \mathrm{KL}(\mathbb{Q}\|\mathbb{P}),$          (7)
     where $\mathcal{L}$ is the cross-entropy loss, and $\mathrm{KL}(\mathbb{Q}\|\mathbb{P})$ is the KL-divergence between the approximate
     posterior $\mathbb{Q}(\boldsymbol{\theta}_1) = \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and the prior $\mathbb{P}(\boldsymbol{\theta}_1) = \mathcal{N}(0, I)$.
    Calculate gradients of variational parameters:
        $\nabla_{\boldsymbol{\mu}_1} l = \frac{\partial l(\boldsymbol{\theta}_1, \boldsymbol{\gamma})}{\partial \boldsymbol{\theta}_1} + \frac{\partial l(\boldsymbol{\theta}_1, \boldsymbol{\gamma})}{\partial \boldsymbol{\mu}_1},$
        $\nabla_{\boldsymbol{\rho}} l = \frac{\partial l(\boldsymbol{\theta}_1, \boldsymbol{\gamma})}{\partial \boldsymbol{\theta}_1} \frac{\boldsymbol{\epsilon}}{1 + \exp(-\boldsymbol{\rho})} + \frac{\partial l(\boldsymbol{\theta}_1, \boldsymbol{\gamma})}{\partial \boldsymbol{\rho}};$          (8)
    Update the variational parameters:
        $\boldsymbol{\mu}_1 \leftarrow \boldsymbol{\mu}_1 - \alpha \nabla_{\boldsymbol{\mu}_1} l$
        $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} - \alpha \nabla_{\boldsymbol{\rho}} l;$          (9)
end
```

$0 - 1$ loss is employed in the theoretical analysis, and the expected risk and the empirical risk of STF-BNN are defined as:

$$\mathcal{R}(\mathbb{Q}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \mathbb{E}_{(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)\sim\mathbb{Q}(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)} \left[\mathbf{1}\left(y \neq \hat{y}\left(\mathbf{x}\right)\right)\right] \tag{11}$$

$$\hat{\mathcal{R}}(\mathbb{Q}) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_{(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)\sim\mathbb{Q}(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)} \left[\mathbf{1}\left(y_i \neq \hat{y}\left(\mathbf{x}_i\right)\right)\right] \tag{12}$$

respectively, where $\mathbf{1}(\cdot)$ is the indicator function, $\mathbb{Q}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ is the joint posterior distribution for $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$. Then, we obtain a generalization bound for STF-BNNs as follows:

Theorem 1 (Generalization bound of STF-BNN)**.** *For any positive real $\delta \in (0, 1)$, with probability at least $1 - \delta$ over a training sample set of size m, we have the following inequality for the distribution of the output hypothesis function $\mathbb{Q}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$:*

$$\mathcal{R}(\mathbb{Q}) \leq \hat{\mathcal{R}}(\mathbb{Q}) + \sqrt{\frac{\Delta + 2\log\frac{1}{\delta} + 2\log m + 4}{4m - 2}}, \tag{13}$$

where

$$\begin{aligned}\Delta =& 2\mathrm{KL}(\mathbb{Q}(\boldsymbol{\theta}_1)\|\mathbb{P}(\boldsymbol{\theta}_1)) + \mathrm{tr}(\boldsymbol{\Sigma} - 2I) \\ &- \log(\det(\boldsymbol{\Sigma})) + \|\boldsymbol{\Sigma}_1\|^2 + \|\boldsymbol{\Sigma}_2\|^2,\end{aligned} \tag{14}$$

and $\boldsymbol{\Sigma}$ is the covariance matrix of $\boldsymbol{\theta}$ and is defined as

$$\boldsymbol{\Sigma} = \operatorname{cov}(\boldsymbol{\theta}, \boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^\top & \boldsymbol{\Sigma}_2 \end{bmatrix}. \tag{15}$$

The proof for this generalization bound has two parts: (1) disentangle the KL divergence *w.r.t.* $\boldsymbol{\theta}$ to the KL divergence *w.r.t.* $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, respectively; and (2) adopt the PAC-Bayesian framework to obtain the bound. A detailed proof is given in Section 7.1.

5.3 STF-BNN Mitigate Overconfidence

For the binary classification STF-BNN $f_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2} : \mathbb{R}^n \to \mathbb{R}$, we theoretically study its capability in tackling the overconfidence in this section. According to Lemma 1, by defining $\mathbf{d}_1(\mathbf{x}) := \nabla_{\boldsymbol{\theta}_1} f_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2}(\mathbf{x})|_{\boldsymbol{\theta}_1=\boldsymbol{\mu}_1}$ for the binary STF-BNN $f_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2}$, where $\boldsymbol{\theta}_1$ has the approximated posterior $\mathcal{N}(\boldsymbol{\theta}_1|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, we can estimate the logit of $f_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2}$ as follows:

$$z(\mathbf{x}) := \frac{f_{\boldsymbol{\mu}_1,\boldsymbol{\theta}_2}(\mathbf{x})}{\sqrt{1 + \pi/8\mathbf{d}_1^\top \boldsymbol{\Sigma}_1 \mathbf{d}_1}}. \tag{16}$$

Theorem 2 (STF-BNN mitigates overconfidence)**.** *Let $f_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2} : \mathbb{R}^n \to \mathbb{R}$ be a binary ReLU STF-BNN parameterized by $\boldsymbol{\theta}_1 \in \mathbb{R}^{r\times n}$ and $\boldsymbol{\theta}_2$, where r is the output dimension of the first Bayesian layer in the STF-BNN. Let $\mathcal{N}(\boldsymbol{\theta}_1|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ be the approximated posterior over $\boldsymbol{\theta}_1$, and $\boldsymbol{\theta}_2$ is obtained by the optimization. Then for any input $\mathbf{x} \in \mathbb{R}^n$,*

$$\lim_{\delta\to\infty} \tau(|z(\delta\mathbf{x})|) \le \tau\left(\frac{\|\mathbf{u}\|\|\mathbf{w}\|}{s_{\min}(\mathbf{u}^\top\mathbf{J})\sqrt{\pi/8\lambda_{\min}(\boldsymbol{\Sigma}_1)}}\right), \tag{17}$$

where $\mathbf{w} \in \mathbb{R}^{r\times n}$ and $\mathbf{u} \in \mathbb{R}^r$ depend only on $\boldsymbol{\mu}_1$ and $\boldsymbol{\theta}_2$, respectively, the matrix $\mathbf{J} := \left.\frac{\partial \mathbf{w}}{\partial \boldsymbol{\theta}_1}\right|_{\boldsymbol{\mu}_1}$ is the Jacobian of $\mathbf{w}$ w.r.t. $\boldsymbol{\theta}_1$ at $\boldsymbol{\mu}_1$, and functions $\lambda_{\min}(\cdot)$ and $s_{\min}(\cdot)$ return the minimum eigenvalue and singular value of their matrix argument, respectively.

The proof is given in Section 7.2. Because $\tau(z(\mathbf{x}))$ is the confidence score of $\mathbf{x}$, Theorem 2 states that the confidence score of $\delta\mathbf{x}$, which is an input far away from the training data when $\delta \to \infty$, can be bounded and will not be close to zero or one in our STF-BNN. Therefore, our approach theoretically limits the confidence score of inputs far away from the training data and mitigates the problem of overconfidence.

6 Experiments

In this section, we analyze the properties of STF-BNN through a comprehensive empirical study on three tasks: uncertainty quantification, privacy preservation, and adversarial robustness.

6.1 Setup

In our experiments, We utilize two popular network architectures of VGG-19 [56] and Wide ResNet-28-10 [67] on four datasets: CIFAR-10, CIFAR-100 [36], and their corresponding corruptions CIFAR-10-C and CIFAR-100-C [25]. The preprocessing of a dataset and data augmentation of training images follow Zagoruyko and Komodakis [67].

We optimize deterministic models for 200 epochs in the first training phase. The learning rate is initialed as 0.1 and is decayed by 0.2 at the $[60, 120, 160]$ epoch. Weight decay factor is set as $5e-4$. The networks only undergo the first training phase are termed as *baselines*. In the second training phase, we train our STF-BNNs for 30 epochs by variational inference. Learning rate is initialed as 0.1 and is decayed by 0.2 at the $[10, 20, 25]$ epoch. Weight decay is $5e-4$ and is only applied on $\boldsymbol{\mu}_1$ of the Bayesian layer. The variational parameters $(\boldsymbol{\mu}_1, \boldsymbol{\rho}_1)$ in the Bayesian layer are initialized by $\boldsymbol{\mu}_1 \sim \mathcal{N}(0, 0.1)$ and $\boldsymbol{\rho} \sim \mathcal{N}(-2.25, 0.1)$, and are updated every one sample for faster convergence. Batch size is set to 128, and SGD with momentum $= 0.9$ is utilized for both training phases.

6.2 Evaluation Metrics

Expected calibration error (ECE) [18] is a widely-used metric for measuring the calibration of networks: predictions are divided into M intervals bins of equal size according to their probability confidence $\hat{p}$. B_m denotes the set of samples whose probability confidence falls into the interval $I_m = (\frac{m-1}{M}, \frac{m}{M}]$ for $m \in \{1, \ldots, M\}$. The average accuracy and confidence of B_m are defined as

$$\operatorname{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}[\hat{y}_i = y_i] \tag{18}$$

$$\operatorname{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i, \tag{19}$$

respectively. Then ECE is defined as:

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{M} \mid \operatorname{acc}(B_m) - \operatorname{conf}(B_m) \mid. \tag{20}$$

Lower ECE corresponds to better calibration performance: the predictive confidence score reflects the prediction accuracy more faithfully, *i.e.*, possesses lower predictive uncertainty.

Membership inference attack [55] is often employed for evaluating the capability of privacy preservation in machine learning models, and we utilize a threshold-based version of it [65] to evaluate our methods in protecting privacy. Given a dataset $\mathcal{S} = (\mathcal{S}_{\text{train}}, \mathcal{S}_{\text{test}})$, where $\mathcal{S}_{\text{train}}$ and $\mathcal{S}_{\text{test}}$ are training set and test set, respectively.

Suppose the "to-be-inferenced" example $(\mathbf{x}, y)$ comes from $\mathcal{S}$, then the accuracy of membership inference attack with a threshold ζ is calculated as follow,

$$\operatorname{Acc}(\zeta) = \frac{1}{2} \times \left(\frac{\sum_{(\mathbf{x},y) \in \mathcal{S}_{\text{train}}} \mathbf{1}[\hat{p}(\mathbf{x}) \geq \zeta]}{|\mathcal{S}_{\text{train}}|} + \frac{\sum_{(x,y) \in \mathcal{S}_{\text{test}}} \mathbf{1}[\hat{p}(\mathbf{x}) < \zeta]}{|\mathcal{S}_{\text{test}}|} \right). \tag{21}$$

Then, we can find the optimal threshold ζ_{optim} with maximizing the attack accuracy, *i.e.*,

$$\zeta_{\text{optim}} = \arg\max_{\zeta} \operatorname{Acc}(\zeta), \tag{22}$$

and $\operatorname{Acc}(\zeta_{\text{optim}})$ is the final attack accuracy for $f_{\boldsymbol{\theta}}$. Lower $\operatorname{Acc}(\zeta_{\text{optim}})$ means the attacker can hardly judge whether an example $\mathbf{x} \in \mathcal{S}$ belongs to S_{train} or S_{test}, *i.e.*, the attacker infers less information from the model.

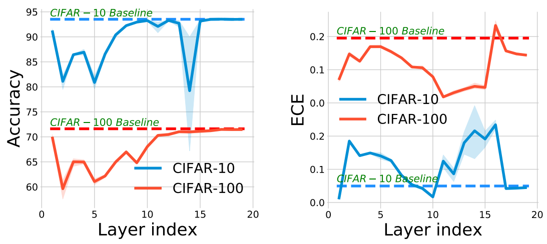
Adversarial training is a popular strategy to enhance the adversarial robustness of neural networks against adversarial examples, which is generated through projected gradient descent (PGD) [43] in our empirical experiments. More specifically, adversarial training can be formulated as solving the minimax-loss problem as follow:

$$\min_{\boldsymbol{\theta}} \frac{1}{m} \sum_{i=1}^{m} \max_{\|\mathbf{x}_i' - \mathbf{x}_i\| \le \xi} \ell\left(f_{\boldsymbol{\theta}}\left(\mathbf{x}_i'\right), y_i\right), \tag{23}$$

where ξ is the *radius* to limit the distance between adversarial examples and original examples. Adversarial examples are also fed to well-trained models for adversarial attack, and a model is adversarial robust if it has a high accuracy under adversarial attack.

6.3 Ablation Studies

To explore how the location of Bayesian layers affects the performance of STF-BNNs, we conduct a layer-wise ablation study by specifying different layers as the Bayesian layer retrained in the second training phase. Then the prediction accuracy and ECE performance are measured on these STF-BNNs with Bayesian layers in different layer indexes, as shown in Figure 3(a) and Figure 3(b). From the figures, we have two observations: (1) there is a good prediction accuracy and a low ECE when the first convolutional layer is converted to the Bayesain layer compared to other layers; (2) according to Figure 3(a), STF-BNNs can restore predictive ability, which was lost in the initialization of the Bayesian layer, by Bayesian inference in the second training phase; and (3) according to Figure 3(b), there is a stable drop in ECE in STF-BNNs compared to the baseline, which no longer undergoes the second training phase.



(a) ACC for different layers (b) ECE for different layers

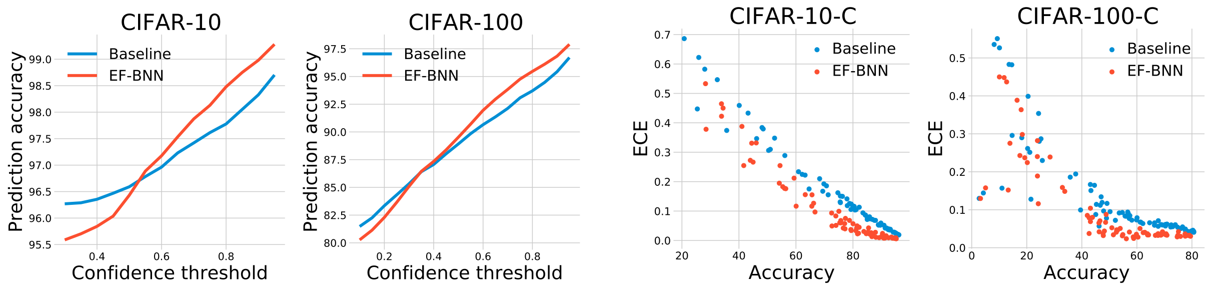
Figure 3: (a) Plot of prediction accuracy formed by varying the layer index of Bayesian layers. (b) Plot of ECE formed by varying the layer index of Bayesian layers. The network architecture is VGG-19, and the models are trained 5 times with different random seeds.

The layer-wise ablation study verifies the superiority of specifying the first layer as the Bayesian layer, and also shows the potential of STF-BNNs in alleviating overconfidence. Therefore, we adopt the STF-BNN whose first layer is Bayesian, in following experiments.

6.4 Evaluation on Uncertainty Quantification

Compared with baseline. We first evaluate the performance under high condidence score with STF-BNNs and baselines on CIFAR-10 and CIFAR-100. To achieve the comparison, we allow models to refuse to give a prediction if the maximum confidence score $\max_i o^{(i)}(f_{\boldsymbol{\theta}}(\mathbf{x}))$ is below a certain confidence threshold, and pictures in Figure 4(a) are plotted by varying the confidence threshold. From Figure 4(a), we observe that when the confidence threshold is large, STF-BNNs attain higher prediction accuracy. In other words, our method is more reliable in the case of a high predictive confidence score compared with the baseline.

We also compare STF-BNNs and the baselines in prediction accuracy and ECE on the corrupted data sets CIFAR-10-C and CIFAR-100-C, each of which contains 75 sub-datasets with different corruption types and intensities. The comparison result is shown in Figure 4(b), and each point in the pictures denotes a sub-dataset of the corruption set. From pictures in Figure 4(b), we have an observation that STF-BNN has lower ECE on the corruption sets than the baseline.



(a) Confidence threshold vs. Accuracy (b) Accuracy vs. ECE

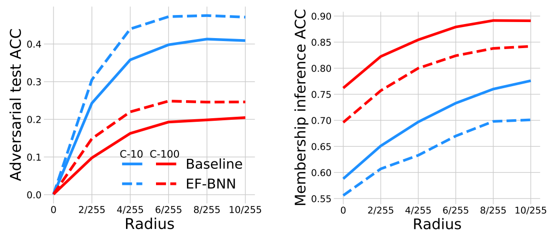
Figure 4: (a) Test accuracy as a function of confidence threshold on CIFAR-10 (left panel) and CIFAR-100 (right panel). (b) Scatter plots between accuracy and ECE formed by employing STF-BNNs and baseline models on different sub data sets in the corruption data set CIFAR-10 (left panel) and CIFAR-100 (right panel). The network architecture is Wide ResNet-28-10.

Compared with other methods. We compared STF-BNNs to other common BNNs or ensembled methods on uncertainty quantification. The result has been shown in Figure 5, and cACC and cECE in the figure denote test accuracy and ECE on CIFAR-10-C and CIFAR-100-C, respectively. From the pictures we obtain the following observations: (1) STF-BNNs (purple bar) have on-par performance of ECE and cECE with the state-of-the-art approaches; (2) training an STF-BNN is time-efficient compared with other methods[1]; (3) MC dropout (yellow bar) also has an short training time, but its performance is significant lower than our method in most cases; and (4) MFVI BNN (green bar), which is the vanilla BNN that only contains random weights and is trained from scratch, has relatively poor performance on both prediction accuracy and ECE, while it needs much longer training time compared to our approach. Therefore, the failure of MFVI BNN confirms the effectiveness of our spacial-temporal fusion from the opposite side.

6.5 Evaluation on Robustness and Privacy

In this section, we evaluate the performance of STF-BNNs on adversarial robustness and privacy preservation through measuring the adversarial attack accuracy and membership inference attack accuracy.

To explore the impact of STF-BNNs on robustness, we deploy the STF-BNN to six well-trained models, which are adversarially trained with six different adversarial radius ξ of $[0, 2/255, 4/255, 6/255, 8/2555, 10/255]$, respectively. Then, we measure the adversarial attack accuracy by employing the adversarial examples with the radius of $10/255$ to attack the adversarial-trained STF-BNNs and corresponding baselines, as shown in Figure 6(a). From the figure, we observe that our method achieves higher accuracy, *i.e.*, preserve better robustness, under adversarial attacks compared to the baselines.



(a) Adversarial attack (b) Member inference attack

Figure 6: (a) Test accuracy under adversarial attack are function of adversarial radius on CIFAR-10 (C-10) and CIFAR-100 (C-100). (b) Membership inference attack accuracy as a function of adversarial radius. (a) and (b) share the same legend. The radius of test adversarial examples in (a) is set to $10/255$. The network architecture is Wide ResNet-28-10.

As for the aspect of privacy protection, it is critical for

[1]The STF-BNN is trained on 1 Tesla V100 GPU, and others are trained on 8 TPUv2 cores according to uncertainty-baselines. 8 TPUv2 cores have stronger performance on training neural networks compare with 1 Tesla V100 GPU.

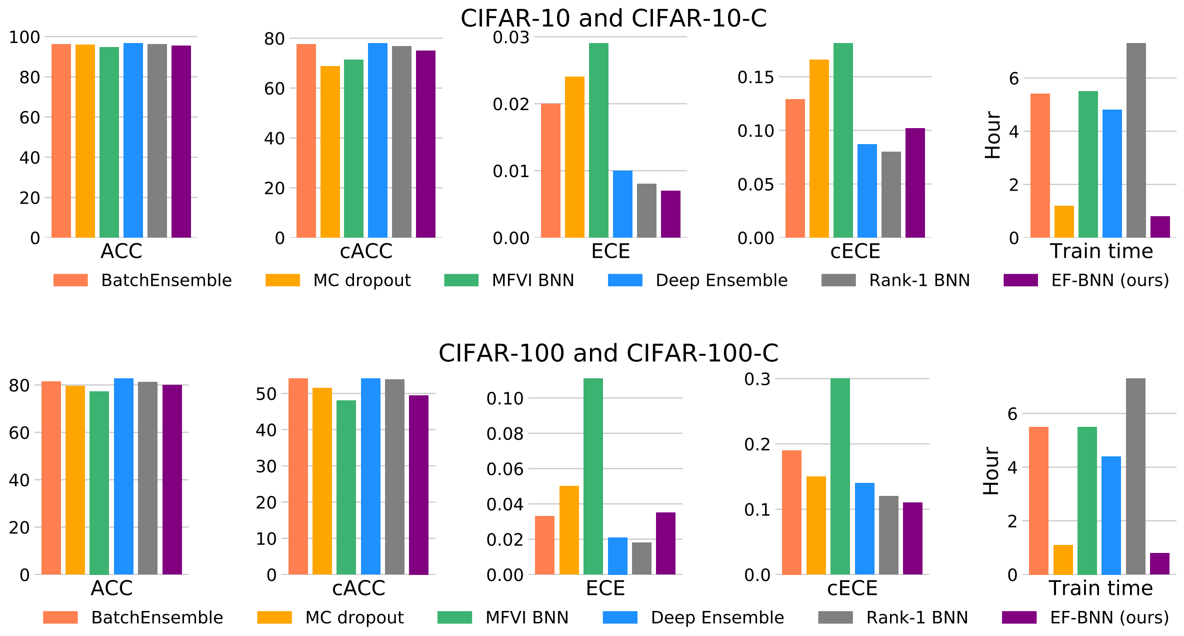


Figure 5: Plots of comparison between different methods on CIFAR-10 and CIFAR-10-C (top panel), CIFAR-100 and CIFAR-100-C (bottom panel), respectively. cACC and cECE denote accuracy and ECE on CIFAR-10/100-C, respectively. The purple bar represents STF-BNN, which has a decent performance on ECE and cECE and are advantageous on training time. All the methods are based on Wide ResNet-28-10 and averaged over 10 seeds.

widespread use of machine learning models, but usually exhibits a trade-off with adversarial robustness in adversarial training, see Song et al. [57]. To determine the capability of privacy preservation of our method, we employ a membership inference attack on the above adversarial-trained STF-BNNs and corresponding baselines. From the result shown in Figure 6(b), we obtain an observation that STF-BNNs decrease the membership inference attack accuracy, *i.e.*, improve the privacy preserving ability on networks with diverse degrees of robustness. The results suggest that our method helps to eliminate the privacy-preserving compromise due to adversarial training.

7 Proofs

The section collects detailed proofs of the results that are omitted in Section 5. To avoid technicalities, the measurability/integrability issues are ignored throughout this paper. Moreover, Fubini's theorem is assumed to be applicable for any integration *w.r.t.* multiple variables. In other words, the order of integrations is exchangeable. Besides, the norm $\|\cdot\|$ denotes the l_2 norm in the following.

7.1 Proof of generalization bound on STF-BNN

In this section, we present the detailed proof of Theorem 1 based on the PAC-Bayesian framework [22, 45, 46]. From the PAC-Bayesian view, a posterior distribution $\mathbb{Q}(\boldsymbol{\theta})$, other than the parameters $\boldsymbol{\theta}_{\mathrm{MLE}}$, is returned by a stochastic learning algorithm. Then, a classic result uniformly bounding the expected risk $\mathcal{R}(\mathbb{Q})$ in terms of the empirical risk $\hat{\mathcal{R}}(\mathbb{Q})$ and the KL divergence $\mathrm{KL}(\mathbb{Q}\|\mathbb{P})$ is as follows.

Lemma 2 ([45], Theorem 2)**.** *For any positive real* $\delta \in (0, 1)$*, with probability at least* $1 - \delta$ *over the sample* *of size m, we have the following inequality for all distribution $\mathbb{Q}$:*

$$\mathcal{R}(\mathbb{Q}) \leq \hat{\mathcal{R}}(\mathbb{Q}) + \sqrt{\frac{\mathrm{KL}(\mathbb{Q}\|\mathbb{P}) + \log\frac{1}{\delta} + \log m + 2}{2m-1}}, \tag{24}$$

where $\mathrm{KL}(\mathbb{Q}\|\mathbb{P})$ *is the KL divergence between the distributions* $\mathbb{Q}$ *and* $\mathbb{P}$ *and is defined as,*

$$\mathrm{KL}(\mathbb{Q}\|\mathbb{P}) = \mathbb{E}_{\boldsymbol{\theta}\sim\mathbb{Q}}\left(\log\frac{\mathbb{Q}(\boldsymbol{\theta})}{\mathbb{P}(\boldsymbol{\theta})}\right). \tag{25}$$

With the above lemma, we can derive the generalization bound of our STF-BNN.

Proof of Theorem 1. For the STF-BNN, the parameters $\boldsymbol{\theta}$ are partitioned into $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ according to whether they are updated in Bayesian inference or optimization, respectively. We use the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, I)$ as the priors of $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, which are denoted by $\mathbb{P}(\boldsymbol{\theta}_1)$ and $\mathbb{P}(\boldsymbol{\theta}_2)$. For $\boldsymbol{\theta}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, we assume the approximate posterior $\mathbb{Q}(\boldsymbol{\theta}_1) = \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ obeys mean-field assumption, which implies $\boldsymbol{\Sigma}_1$ is a diagonal matrix. On the other hand, $\boldsymbol{\theta}_2$ is optimized by SGD, which is well-known as the Ornstein-Uhlenbeck process [58], and thus $\boldsymbol{\theta}_2$ has an analytic stationary distribution according to [44]:

$$\mathbb{Q}(\boldsymbol{\theta}_2) = \frac{1}{\sqrt{2\pi\det(\boldsymbol{\Sigma}_2)}}\exp\left\{-\frac{1}{2}\boldsymbol{\theta}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\theta}_2\right\}. \tag{26}$$

For simplify, we denote $\mathbb{Q}(\boldsymbol{\theta}_1), \mathbb{P}(\boldsymbol{\theta}_1), \mathbb{Q}(\boldsymbol{\theta}_2), \mathbb{P}(\boldsymbol{\theta}_2)$ as $\mathbb{Q}_1, \mathbb{P}_1, \mathbb{Q}_2, \mathbb{P}_2$, respectively. Recall $\mathrm{KL}(\mathbb{Q}\|\mathbb{P})$ in Eq. 25, we have

$$\begin{aligned}
&\log\frac{\mathbb{Q}(\boldsymbol{\theta})}{\mathbb{P}(\boldsymbol{\theta})}\\
=&\log\frac{\mathbb{Q}(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)}{\mathbb{P}(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)}\\
=&\log\frac{\mathbb{Q}(\boldsymbol{\theta}_1)\mathbb{Q}(\boldsymbol{\theta}_2)}{\mathbb{P}(\boldsymbol{\theta}_1)\mathbb{P}(\boldsymbol{\theta}_2)} + \log\frac{\mathbb{Q}(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)}{\mathbb{Q}(\boldsymbol{\theta}_1)\mathbb{Q}(\boldsymbol{\theta}_2)}\\
=&\log\frac{\mathbb{Q}(\boldsymbol{\theta}_1)}{\mathbb{P}(\boldsymbol{\theta}_1)} + \log\frac{\mathbb{Q}(\boldsymbol{\theta}_2)}{\mathbb{P}(\boldsymbol{\theta}_2)} + \log\frac{\mathbb{Q}(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)}{\mathbb{Q}(\boldsymbol{\theta}_1)\mathbb{Q}(\boldsymbol{\theta}_2)}.
\end{aligned} \tag{27}$$

Hence, $\mathrm{KL}(\mathbb{Q}\|\mathbb{P})$ can be written as:

$$\mathrm{KL}(\mathbb{Q}_1\|\mathbb{P}_1) + \mathbb{E}_{\boldsymbol{\theta}\sim\mathbb{Q}}\left(\log\frac{\mathbb{Q}(\boldsymbol{\theta}_2)}{\mathbb{P}(\boldsymbol{\theta}_2)} + \log\frac{\mathbb{Q}(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)}{\mathbb{Q}(\boldsymbol{\theta}_1)\mathbb{Q}(\boldsymbol{\theta}_2)}\right). \tag{28}$$

By plugging $\mathbb{P}(\boldsymbol{\theta}_2)$ and $\mathbb{Q}(\boldsymbol{\theta}_2)$ into $\log\frac{\mathbb{Q}(\boldsymbol{\theta}_2)}{\mathbb{P}(\boldsymbol{\theta}_2)}$,

$$\begin{aligned}
&\log\left(\frac{\mathbb{Q}(\boldsymbol{\theta}_2)}{\mathbb{P}(\boldsymbol{\theta}_2)}\right)\\
=&\log\left(\frac{\sqrt{2\pi\det(I)}}{\sqrt{2\pi\det(\boldsymbol{\Sigma}_2)}}\exp\left\{\frac{1}{2}\boldsymbol{\theta}_2^\top I\boldsymbol{\theta}_2 - \frac{1}{2}\boldsymbol{\theta}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\theta}_2\right\}\right)\\
=&\frac{1}{2}\log\left(\frac{1}{\det(\boldsymbol{\Sigma}_2)}\right) + \frac{1}{2}\left(\boldsymbol{\theta}_2^\top I\boldsymbol{\theta}_2 - \boldsymbol{\theta}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\theta}_2\right).
\end{aligned} \tag{29}$$

For the joint posterior distribution $\mathbb{Q}(\boldsymbol{\theta}) = \mathbb{Q}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$, its covariance matrix $\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^\top & \boldsymbol{\Sigma}_2 \end{bmatrix}$. However, for $\mathbb{Q}(\boldsymbol{\theta}_1)\mathbb{Q}(\boldsymbol{\theta}_2)$, because $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are independent in this distribution, $\boldsymbol{\Sigma}_{12} = \mathrm{cov}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \mathbf{0}$ and its covariance matrix $\boldsymbol{\Sigma}' = \begin{bmatrix} \boldsymbol{\Sigma}_1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_2 \end{bmatrix}$. Therefore, $\log \frac{\mathbb{Q}(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)}{\mathbb{Q}(\boldsymbol{\theta}_1)\mathbb{Q}(\boldsymbol{\theta}_2)}$ can be written as

$$\begin{aligned}
&\log \frac{\mathbb{Q}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)}{\mathbb{Q}(\boldsymbol{\theta}_1)\mathbb{Q}(\boldsymbol{\theta}_2)} \\
=&\log \left(\frac{\sqrt{2\pi \det(\boldsymbol{\Sigma}')}}{\sqrt{2\pi \det(\boldsymbol{\Sigma})}} \exp\left\{ \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{\Sigma}' \boldsymbol{\theta} - \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\theta} \right\} \right) \\
=&\frac{1}{2}\log\left(\frac{\det(\boldsymbol{\Sigma}')}{\det(\boldsymbol{\Sigma})}\right) + \frac{1}{2}\left(\boldsymbol{\theta}^\top\boldsymbol{\Sigma}'\boldsymbol{\theta} - \boldsymbol{\theta}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\theta}\right).
\end{aligned} \tag{30}$$

By plugging Eq. 29 and 30 into Eq. 28, we have

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\theta}\sim\mathbb{Q}}\left(\log\frac{\mathbb{Q}(\boldsymbol{\theta}_2)}{\mathbb{P}(\boldsymbol{\theta}_2)} + \log\frac{\mathbb{Q}(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)}{\mathbb{Q}(\boldsymbol{\theta}_1)\mathbb{Q}(\boldsymbol{\theta}_2)}\right) \\
=&\int_{\boldsymbol{\theta}\in\Theta}\left[\frac{1}{2}\log\left(\frac{1}{\det(\boldsymbol{\Sigma}_2)}\right) + \frac{1}{2}\left(\boldsymbol{\theta}_2^\top \boldsymbol{I}\boldsymbol{\theta}_2 - \boldsymbol{\theta}_2^\top\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\theta}_2\right)\right. \\
&\left. + \frac{1}{2}\log\left(\frac{\det(\boldsymbol{\Sigma}')}{\det(\boldsymbol{\Sigma})}\right) + \frac{1}{2}\left(\boldsymbol{\theta}^\top\boldsymbol{\Sigma}'\boldsymbol{\theta} - \boldsymbol{\theta}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\theta}\right)\right] q(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta} \\
=&\frac{1}{2}\log\left(\frac{\det(\boldsymbol{\Sigma}_1)\det(\boldsymbol{\Sigma}_2)}{\det(\boldsymbol{\Sigma})\det(\boldsymbol{\Sigma}_2)}\right) + \frac{1}{2}\mathbb{E}_{\boldsymbol{\theta}_2}\left[\boldsymbol{\theta}_2^\top\boldsymbol{I}\boldsymbol{\theta}_2\right] - \\
&\frac{1}{2}\mathbb{E}_{\boldsymbol{\theta}}\left[\boldsymbol{\theta}_2^\top\boldsymbol{\Sigma}_2\boldsymbol{\theta}_2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{\theta}}\left[\boldsymbol{\theta}^\top\boldsymbol{\Sigma}'\boldsymbol{\theta}\right] - \frac{1}{2}\mathbb{E}_{\boldsymbol{\theta}}\left[\boldsymbol{\theta}^\top\boldsymbol{I}\boldsymbol{\theta}\right] \\
=&\frac{1}{2}\log\left(\frac{\det(\boldsymbol{\Sigma}_1)}{\det(\boldsymbol{\Sigma})}\right) + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_2 - I) + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}'\boldsymbol{\Sigma} - I) \\
\leq&\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_1 - I) - \frac{1}{2}\log\det(\boldsymbol{\Sigma}) + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_2 - I) + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}'\boldsymbol{\Sigma} - I) \\
=&-\frac{1}{2}\log\det(\boldsymbol{\Sigma}) + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma} - I) + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}'\boldsymbol{\Sigma} - I) \\
=&-\frac{1}{2}\log\det(\boldsymbol{\Sigma}) + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma} - 2I) + \frac{1}{2}(\|\boldsymbol{\Sigma}_1\|^2 + \|\boldsymbol{\Sigma}_2\|^2)
\end{aligned} \tag{31}$$

Then plugging the above equation and Eq. 28 into Lemma 2 yields the desired inequality and the proof is finished. □

7.2 Proof of the capability of Mitigating Overconfidence about STF-BNN

In this section, we present the detailed proof of Theorem 2. For a ReLU network, its input space is partitioned into linear regions by the nonlinearities at the activation [49, 51]. The output-input mapping induced by a ReLU network is linear with respect to the input data within linear regions and nonlinear and non-smooth in the boundaries between linear regions. Then, with the notion of linear regions, we introduced Lemma 3 as follows.

Lemma 3 ([24], Lemma 3.1)**.** *Let $\{Q_i\}_{l-1}^{R}$ be the set of linear regions associated to the ReLU network $f : \mathbb{R}^n \to \mathbb{R}^k$. For any $x \in \mathbb{R}^n$ there exists an $\alpha > 0$ and $t \in \{1, ..., R\}$ such that $\delta x \in Q_t$ for all $\delta \geq \alpha$.*

Furthermore, the restriction of f to Q_t can be written as an affine function $\mathbf{U}x + \mathbf{c}$ for some suitable $\mathbf{U} \in \mathbb{R}^{k\times n}$ and $\mathbf{c} \in \mathbb{R}^k$.

With the above lemma, we can prove the capability of mitigating overconfidence of our STF-BNN.

Proof of Theorem 2. The binary STF-BNN $f_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2} : \mathbb{R}^n \to \mathbb{R}$ consists of $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$. According to Lemma 3 there exists an $\alpha > 0$ and a linear region R, along with $\mathbf{u}^\top \in \mathbb{R}^r$, $\mathbf{w} \in \mathbb{R}^{r\times n}$, $\mathbf{b} \in \mathbb{R}^r$, and $c \in \mathbb{R}$, such that for any $\delta \geq \alpha$, we have that $\delta\mathbf{x} \in R$ and restriction $f_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2|R}$ can be written as $\mathbf{u}^\top(\mathbf{wx} + \mathbf{b}) + c$, where r is the output dimension of the first Bayesian layer in the STF-BNN. Therefore for any such δ, we can write the gradient $\mathbf{d}_1(\delta\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}_1}\left(\mathbf{u}^\top(\mathbf{w}\delta\mathbf{x} + \mathbf{b}) + c\right)\big|_{\boldsymbol{\mu}_1}$ as follows:

$$\begin{aligned}
\mathbf{d}(\delta\mathbf{x}) &= \left.\frac{\partial\left(\delta\mathbf{u}^\top\mathbf{wx}\right)}{\partial\boldsymbol{\theta}_1}\right|_{\boldsymbol{\mu}_1} + \left.\frac{\partial\mathbf{u}^\top\mathbf{b}}{\partial\boldsymbol{\theta}_1}\right|_{\boldsymbol{\mu}_1} + \left.\frac{\partial c}{\partial\boldsymbol{\theta}_1}\right|_{\boldsymbol{\mu}_1} \\
&= \delta\left.\frac{\partial\mathbf{u}^\top\mathbf{wx}}{\partial\boldsymbol{\theta}_1}\right|_{\boldsymbol{\mu}_1}\mathbf{x} + \left.\frac{\partial\mathbf{u}^\top\mathbf{b}}{\partial\boldsymbol{\theta}_1}\right|_{\boldsymbol{\mu}_1} \\
&= \delta\left(\mathbf{u}^\top\mathbf{Jx} + \frac{1}{\delta}\nabla_{\boldsymbol{\theta}_1}\mathbf{u}^\top\mathbf{b}|_{\boldsymbol{\mu}_1}\right),
\end{aligned} \tag{32}$$

where $\mathbf{J} = \left.\frac{\partial\mathbf{w}}{\partial\boldsymbol{\theta}_1}\right|_{\boldsymbol{\mu}_1}$. Then, by Eq. 16 we have

$$\begin{aligned}
|z(\delta\mathbf{x})| =& \frac{\left|\delta\mathbf{u}^\top\mathbf{wx} + \mathbf{u}^\top\mathbf{b} + c\right|}{\sqrt{1 + \pi/8\mathbf{d}_1(\delta\mathbf{x})^\top\boldsymbol{\Sigma}_1\mathbf{d}_1(\delta\mathbf{x})}} \\
=& \frac{\left|\delta\left(\mathbf{u}^\top\mathbf{wx} + \frac{1}{\delta}\mathbf{u}^\top\mathbf{b} + \frac{1}{\delta}\mathbf{c}\right)\right|}{\sqrt{1 + \pi/8\delta^2\boldsymbol{\Omega}^\top\boldsymbol{\Sigma}_1\boldsymbol{\Omega}}} \\
=& \frac{\left|\left(\mathbf{u}^\top\mathbf{wx} + \frac{1}{\delta}\mathbf{u}^\top\mathbf{b} + \frac{1}{\delta}\mathbf{c}\right)\right|}{\sqrt{\frac{1}{\delta^2} + \pi/8\boldsymbol{\Omega}^\top\boldsymbol{\Sigma}_1\boldsymbol{\Omega}}},
\end{aligned} \tag{33}$$

where $\boldsymbol{\Omega} = \mathbf{u}^\top\mathbf{Jx} + \frac{1}{\delta}\nabla_{\boldsymbol{\theta}_1}\mathbf{u}^\top\mathbf{b}|_{\boldsymbol{\mu}_1}$. Notice that as $\delta \to \infty$, $1/\delta$ and $1/\delta^2$ go to 0. Hence, in the limit, we have that

$$\lim_{\delta\to\infty}|z(\delta\mathbf{x})| = \frac{\left|\mathbf{u}^\top\mathbf{wx}\right|}{\sqrt{\pi/8\left(\mathbf{u}^\top\mathbf{Jx}\right)^\top\boldsymbol{\Sigma}_1\left(\mathbf{u}^\top\mathbf{Jx}\right)}}. \tag{34}$$

To get the desired result, the following lemmas in terms of matrix operations are needed.

Lemma 4 ([35], Lemma A.2)**.** *Let $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{n\times n}$ be an SPD matrix, then $\mathbf{x}^\top\mathbf{Ax} \geq \lambda_{\min}(\mathbf{A})\|\mathbf{x}\|^2$.*

Lemma 5 ([35], Lemma A.3)**.** *Let $\mathbf{A} \in \mathbb{R}^{r\times n}$ and $\mathbf{z} \in \mathbb{R}^n$ with $r \geq n$, then $\|\mathbf{Az}\|^2 \geq s^2_{\min}(\mathbf{A})\|\mathbf{z}\|^2$.*

Then, by using the Cauchy-Schwarz inequality, Lemma 4, and Lemma 5 sequentially, we can upper-bound this limit:

$$\begin{aligned}\lim_{\delta\to\infty}|z(\delta\mathbf{x})| &= \frac{\left|\mathbf{u}^\top\mathbf{w}\mathbf{x}\right|}{\sqrt{\pi/8\left(\mathbf{u}^\top\mathbf{J}\mathbf{x}\right)^\top\mathbf{\Sigma}_1\left(\mathbf{u}^\top\mathbf{J}\mathbf{x}\right)}}\\ &\leq \frac{\|\mathbf{u}^\top\|\|\mathbf{w}\|\|\mathbf{x}\|}{\sqrt{\pi/8\lambda_{\min}(\mathbf{\Sigma}_1)\|\mathbf{u}^\top\mathbf{J}\mathbf{x}\|^2}}\\ &\leq \frac{\|\mathbf{u}^\top\|\|\mathbf{w}\|}{s_{\min}(\mathbf{u}^\top\mathbf{J})\sqrt{\pi/8\lambda_{\min}(\mathbf{\Sigma}_1)}},\end{aligned} \tag{35}$$

With $\|\mathbf{u}^\top\| = \|\mathbf{u}\|$, thus the proof is complete. $\square$

8 Conclusion

This paper designs the spatial-temporal-fusion BNN, which fuses the optimization and Bayesian inference from both spatial and temporal aspects to efficiently improve the reliability of neural networks. This fusion is based on an empirical finding that the first layer of the network has lower stability when it is retrained. Theoretical analysis has been provided on the generalization ability and the capability of mitigating overconfidence of our STF-BNN. Sufficient empirical studies present that STF-BNNs (1) achieve on-par performance on prediction and uncertainty quantification with the state-of-the-art methods; (2) have a tremendous decrease in training time and required memory; and (3) significantly improve adversarial robustness and privacy preservation.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] Sungjin Ahn, Anoop Korattikara Balan, and Max Welling. Bayesian Posterior Sampling via Stochastic Gradient Fisher Scoring. In *Proceedings of the 29th International Conference on Machine Learning*, page 230. icml.cc / Omnipress, 2012.

[3] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[4] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/blundell15.html`.

[5] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[6] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691, 2014.

[7] Michael Cogswell, Faruk Ahmed, Ross Girshick, Larry Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*, 2015.

[8] Zhijie Deng, Xiao Yang, Hao Zhang, Yinpeng Dong, and Jun Zhu. Bayesadapter: Being bayesian, inexpensively and robustly, via bayeisan fine-tuning. *arXiv preprint arXiv:2010.01979*, 2020.

[9] Chao Du, Jun Zhu, and Bo Zhang. Learning deep generative models with doubly stochastic gradient mcmc. *IEEE Transactions on Neural Networks and Learning Systems*, 29(7):3084–3096, 2018. doi: 10.1109/TNNLS.2017.2688499.

[10] Michael Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable Bayesian neural nets with rank-1 factors. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2782–2792. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/dusenberry20a.html`.

[11] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[12] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[13] Sebastian Farquhar, Michael A Osborne, and Yarin Gal. Radial bayesian neural networks: Beyond discrete support in large-scale bayesian deep learning. *stat*, 1050:7, 2020.

[14] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[15] Dani Gamerman and Hedibert F Lopes. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. CRC Press, 2006.

[16] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL `http://arxiv.org/abs/1412.6572`.

[17] Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24:2348–2356, 2011.

[18] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/guo17a.html`.

[19] Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schon. Evaluating scalable bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 318–319, 2020.

[20] WK Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97, 1970.

[21] Fengxiang He and Dacheng Tao. Recent advances in deep learning theory. *arXiv preprint arXiv:2012.10931*, 2020.

[22] Fengxiang He, Tongliang Liu, and Dacheng Tao. Control batch size and learning rate to generalize well: Theoretical and empirical evidence. In *Advances in Neural Information Processing Systems*, pages 1143–1152, 2019.

[23] Fengxiang He, Bohan Wang, and Dacheng Tao. Tighter generalization bounds for iterative differentially private learning algorithms. In *Conference on Uncertainty in Artificial Intelligence*, 2021.

[24] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019.

[25] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[26] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[27] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[28] Xiaoyang Huang, Jiancheng Yang, Linguo Li, Haoran Deng, Bingbing Ni, and Yi Xu. Evaluating and boosting uncertainty quantification in classification. *arXiv preprint arXiv:1909.06030*, 2019.

[29] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, et al. An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*, 2015.

[30] Mohammad Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable Bayesian deep learning by weight-perturbation in Adam. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2611–2620. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/khan18a.html`.

[31] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[32] Lingkai Kong, Jimeng Sun, and Chao Zhang. SDE-net: Equipping deep neural networks with uncertainty estimates. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5405–5415. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/kong20b.html`.

[33] Takuya Konishi, Takatomi Kubo, Kazuho Watanabe, and Kazushi Ikeda. Variational bayesian inference algorithms for infinite relational model of network data. *IEEE Transactions on Neural Networks and Learning Systems*, 26(9):2176–2181, 2015. doi: 10.1109/TNNLS.2014.2362012.

[34] Ranganath Krishnan, Mahesh Subedar, and Omesh Tickoo. Specifying weight priors in bayesian deep neural networks with empirical bayes. In *AAAI*, pages 4477–4484, 2020.

[35] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks. In *International Conference on Machine Learning*, pages 5436–5446. PMLR, 2020.

[36] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[37] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

[38] Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via

distance awareness. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7498–7512. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/543e83748234f7cbab21aa0ade66565f-Paper.pdf`.

[39] Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. Adv-BNN: Improved adversarial defense through robust bayesian neural network. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rk4Qso0cKm`.

[40] Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient mcmc. *Advances in neural information processing systems*, 28:2917–2925, 2015.

[41] David JC MacKay. The evidence framework applied to classification networks. *Neural computation*, 4 (5):720–736, 1992.

[42] David JC MacKay. Probable networks and plausible predictions-a review of practical bayesian methods for supervised neural networks. *Network: computation in neural systems*, 6(3):469, 1995.

[43] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[44] Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35, 2017.

[45] David A McAllester. Pac-bayesian model averaging. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 164–170, 1999.

[46] David A McAllester. Some pac-bayesian theorems. *Machine Learning*, 37(3):355–363, 1999.

[47] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

[48] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[49] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper/2014/file/109d2dd3608f669ca17920c511c2a41e-Paper.pdf`.

[50] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2 (11):2, 2011.

[51] Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*, 2013.

[52] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[53] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *Advances in Neural Information Processing Systems*, pages 3358–3369, 2019.

[54] Afshar Shamsi, Hamzeh Asgharnezhad, Shirin Shamsi Jokandan, Abbas Khosravi, Parham M. Kebria, Darius Nahavandi, Saeid Nahavandi, and Dipti Srinivasan. An uncertainty-aware transfer learning-based framework for covid-19 diagnosis. *IEEE Transactions on Neural Networks and Learning Systems*, 32(4): 1408–1417, 2021. doi: 10.1109/TNNLS.2021.3054306.

[55] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[56] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[57] Liwei Song, Reza Shokri, and Prateek Mittal. Privacy risks of securing machine learning models against adversarial examples. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 241–257, 2019.

[58] George E Uhlenbeck and Leonard S Ornstein. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

[59] Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *International Conference on Machine Learning*, pages 9690–9700. PMLR, 2020.

[60] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

[61] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=Sklf1yrYDr`.

[62] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[63] Yi Yang, Shuai Huang, Wei Huang, and Xiangyu Chang. Privacy-preserving cost-sensitive learning. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):2105–2116, 2021. doi: 10.1109/TNNLS.2020.2996972.

[64] Nanyang Ye and Zhanxing Zhu. Bayesian adversarial learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6892–6901, 2018.

[65] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282. IEEE, 2018.

[66] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2805–2824, 2019. doi: 10.1109/TNNLS.2018.2886017.

[67] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016. ISBN 1-901725-59-6. doi: 10.5244/C.30.87. URL `https://dx.doi.org/10.5244/C.30.87`.

[68] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan.

Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482. PMLR, 2019.

[69] Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. Geometry-aware instance-reweighted adversarial training. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=iAX0l6Cz8ub`.

[70] Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for bayesian deep learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rkeS1RVtPS`.