Towards Stochastic Neural Networks via Inductive Wasserstein Embeddings

Hao Yang [*1] **Yongxin Yang** [*2] **Da Li** [2 3] **Yun Zhou** [1] **Timothy Hospedales** [2 3]

Abstract

Stochastic Neural Networks (SNNs) have recently shown promising results in adversarial defense, label noise robustness, and model calibration. However, the current implementations are dominated by the Gaussian noise injection, which essentially models the activations and/or the model weights using Gaussian distributions. Despite its ease of use, the Gaussian distribution has its own limitation in being unimodal. In this work, we present inductive Wasserstein embeddings, which models the activations and weights using implicit density, i.e., we do not assign a specific form for the distribution such as Gaussian, instead we design the sampling process. Compared to the alternatives, our method achieves competitive results on various tasks.

1. Introduction

Stochastic neural networks (SNNs) have attracted great attentions from the machine learning community in various contexts, such as Bayesian deep learning (Blundell et al., 2015; Hernández-Lobato & Adams, 2015; Gal & Ghahramani, 2016; Huang et al., 2020), adversarial defense (Alemi et al., 2017; Hendrycks et al., 2018), and model calibration (Guo et al., 2017).

One representative stream of SNNs in deep learning is the deep Variational Information Bottleneck (VIB) method (Alemi et al., 2017; 2018), which is inspired by the information theory (Tishby et al., 1999). Though VIB has demonstrated exceptional performances in multiple tasks, such as standard supervised learning and adversarial defence, it still suffers from the same limitation as the Variational Autoencoder (VAE) in terms of using a standard Gaussian prior (Tomczak & Welling, 2018; Kim & Pavlovic, 2020).

Some researchers recently argued that the standard Gaussian prior hinders VIB capturing the task uncertainty and limits its performance, thus they proposed some variants with Gaussian prior replaced by others, e.g., non-informative prior in Yu et al. (2021). Those alternative choices still fall into Gaussian family, and so are easy to implement using the reparameterization trick. However, Gaussian distributions might be insufficient to model multi-modal cases, found in complex tasks.

In this paper, we relax the Gaussian distribution assumption, and propose to model both the feature and class prototypes using implicit density. More specifically, we do not assign a specific distribution function, instead, we design the sampling process. This can be thought of as the inductive version of Wasserstein embeddings (Frogner et al., 2019), where an object (e.g., a word or a node in a graph) is represented by a set of vectors (point-cloud), and the difference of two objects is measured by Wasserstein distance of two sets. This approach (Frogner et al., 2019) is transductive because the set of vectors corresponds to the trainable parameter, thus it is not applicable to new concepts.

In contrast, we derive an inductive variant for general supervised learning, where some data (e.g., feature vectors and class labels) are combined with random noise, and passed through a neural network, to get some samples conditioned on those data. The archtecuture is very close to the generator of Conditional GAN (Mirza & Osindero, 2014). The set of samples, as a whole, will be treated as the representation of the input feature/label, and the difference of them is calculated by the Sliced Wasserstein distance (Rabin et al., 2011).

The illustration of the proposed method can be found in Fig. 1. Because of the intrinsic stochasticity, this method is a realisation of a stochastic neural network. We evaluate the proposed SNNs on three tasks: adversarial defence, model calibration, and label noise robustness. Results show that our method achieves the state-of-the-art performance across various settings.

2. Preliminaries

2.1. Computation of logits

For classification using conventional neural networks, the final layer involves the computation of logits, which is typi-

[*]Equal contribution [1]National University of Defense Technology [2]University of Edinburgh [3]Samsung AI Center, Cambridge. Correspondence to: Yun Zhou <zhouyun@nudt.edu.cn>.

Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning.,

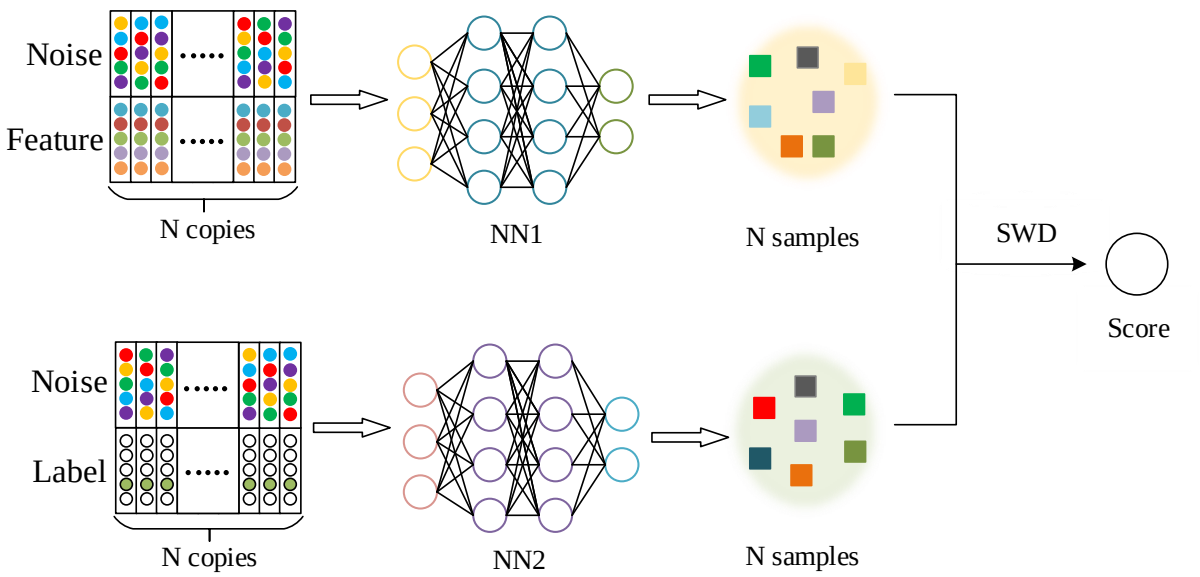

Figure 1. An illustration of our proposed method. Each feature and putative label (one-hot vector) are mapped to non-parametric probability distributions represented by sample sets. The associated classifier logit is computed using Sliced Wasserstein Distance between the feature and label distributions. Feature and label to distribution mapping is simply implemented by duplicating the feature and prototype N times, concatenating with N independent noise vectors, and passing through two separate neural networks, which finally leads to an N sample representation of each.

cally a dot product of feature vector x and classifier matrix W. Then, the probability of class i for the given input x is calculated by,

$$p(Y=i|x) = \frac{\exp\left(x \cdot W_{\cdot,i}\right)}{\sum_{j=1}^{C} \exp\left(x \cdot W_{\cdot,j}\right)} \tag{1}$$

where $W_{\cdot,j}$ stands for the jth column of matrix W. The dot-product is a kind of similarity measure (cosine similarity without normalisation), thus it can be replaced by a negative distance measure $d(\cdot,\cdot)$.

$$p(Y=i|x) = \frac{\exp\left(-d(x, W_{\cdot,i})\right)}{\sum_{j=1}^{C} \exp\left(-d(x, W_{\cdot,j})\right)} \tag{2}$$

2.2. Stochastic Neural Networks

Different from the conventional neural networks, the stochastic neural networks treat x and/or W as distribution instead of point-estimate, thus the vector-to-vector distance function $d(\cdot,\cdot)$ should be replaced with a divergence.

In practice, this is realised by injecting noises into model weights (Blundell et al., 2015) or activations (Alemi et al., 2017; Kingma & Welling, 2014). For example, SE-SNN (Yu et al., 2021) proposed to use

$$x \longleftarrow f_\mu(x) + f_\sigma(x) \odot \epsilon \tag{3}$$

where $f_\mu(x)$ is a neural network that produces the mean vector, $f_\sigma(x)$ is another neural network that produces the (square root of) variance vector, and ϵ is a random sample form standard Gaussian $\mathcal{N}(0, I)$. Equivalently, this indicates that x is modelled by a Gaussian parametrised by neural networks, i.e., $x := \mathcal{N}(f_\mu(x), \mathrm{diag}(f_\sigma^2(x)))$.

Similarly, $W_{\cdot,j}$ can be modelled as a Gaussian distribution, and $d(x, W_{\cdot,i})$ can be any Gaussian-to-Gaussian divergence. Alternatively, one can use sample(s) from the distribution, and calculate the vector-to-vector distance as usual.

To prevent the distribution from collapsing, the SNNs usually have the regularisation term that promotes non-zero variance. SNNs showed some nice properties for the cases of adversarial attack, label noise robustness, etc.

3. Inductive Wasserstein Embeddings

Previous work is dominated by the use of Gaussian distribution, or a specific kind of distribution which we know how to do reparameterization. In this work, we aim to relax this assumption to deal with arbitrarily distributions – that support multi-modality for example. In fact, we do not have to pre-define the distribution function, all we need is a mechanism to sample from the distribution conditioned on

x or $W_{\cdot,j}$. A simple model for x is

$$x := f_\theta(x, \epsilon) = \text{MLP}_\theta([[x; x; \ldots; x]; \epsilon]]) \quad (4)$$

where $[x; x; \ldots; x]$ is N copies of feature vector x, and ϵ is a $N \times D$ matrix of pure noises (from the uniform/normal distribution). D is a hyper-parameter that controls the amount of samples. $[a; b]$ stands for row-wise concatenation of a and b. MLP_θ is a neural network parametrised by θ, and it produces N output vectors for the N input vectors. These N outputs are considered as N independent samples for the distribution conditioned on x.

Similarly, we can model $W_{\cdot,j}$ by

$$W_{\cdot,j} := g_\phi(c_j, \epsilon) = \text{MLP}_\phi([[c_j; c_j; \ldots; c_j]; \epsilon]]) \quad (5)$$

where c_j is the one-hot vector with the jth position being one. Alternatively, c_j could be a dense vector representing the prototype of the jth class.

Now $d(x, W_{\cdot,i}))$ becomes a metric over set of samples, and each set represents a distribution. There exit several choices for $d(\cdot, \cdot)$, and the desired properties are (i) efficient computation and (ii) differentiability. Thus we choose to use the Sliced Wasserstein Distance (SWD) (Rabin et al., 2011), as it has a lower computational cost than the Sinkhorn iterative method, and it is fully differentiable.

Finally, we need a mechanism that makes sure the distribution does not collapse into a single point. A simple solution is to maximise the sample variance, i.e.,

$$\max_\theta \text{Var}(f_\theta(x, \epsilon)) \quad (6)$$

In practice, it is better to use a margin loss so that the variance would not go overly large.

$$\min_\theta \max(0, b - \text{Var}(f_\theta(x, \epsilon))) \quad (7)$$

Thus the loss is zero when the variance is larger than a certain threshold b. The similar regularization term is also defined for g_ϕ.

To demonstrate the proposed method, we train a classification model for MNIST with distribution dimension being 2, thus it is possible to visualize the embeddings directly in Fig. 2, where we can see both instance x and classifier $W_{\cdot,j}$ are in the form of distributions (point clouds).

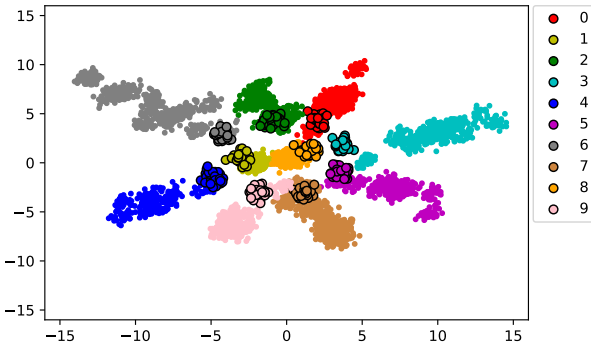

Figure 2. Visualizing high-dimensional MNIST embeddings. Each color corresponds to one class. The dots with black border are samples from the class distribution, and the borderless dots are samples from (multiple) instance distributions.

4. Experiments

We evaluate our proposed method in tasks: adversarial defense, model calibration, and label noise robustness.

4.1. Adversarial Defense

Setting Following Dong et al. (2018) and Madry et al. (2017), we focus on two types of adversarial attacks: Fast Gradient Sign Method (FGSM) and Project Gradient Descent (PGD) attack (Madry et al., 2017). A popular architecture we used is ResNet18 (V2), which is different from classical ResNet in that we use 1x1 CNN layers instead of average pooling for down-sampling. We compare the original undefended network (denoted **No defense**), adversarial training (**adversarial**), parametric noise injection (**PNI**, (He et al., 2019)) and **SE-SNN** (Yu et al., 2021).

We train different models on CIFAR-10 and then use FGSM and Project Gradient Descent (PGD) attacks to evaluate the adversarial robustness of different methods. For the FGSM and PGD attack, we keep a consistent settings as PNI and data is normalized to 0-1. In PGD attack, we set the iteration times as 7 for each data and α is 0.01.

Results As can be seen from the comparative results in Table 2, our proposed method outperforms the competing defense methods on FGSM and PGD attacks. Different from SE-SNN and PNI, which assume Gaussian stochastic layers, our proposed method relaxes this assumption to a more flexible distribution.

4.2. Model Calibration

Setting We evaluate our proposed method following the protocol in Guo et al. (2017) except that we set the kernel size of the first conv layer to 7. We follow the architecture used in Yu et al. (2021) and add a stochastic layer before the classification layer in ResNet-18, ResNet-50, and ResNext101-32-8d respectively with datasets CIFAR-10 and CIFAR-100. In this experiment, we choose a widely-used evaluation matrix called Expected Calibration Error (ECE) (Naeini et al., 2015; Guo et al., 2017) to measure the calibration error. ECE approximates the difference in expection between confidence and accuracy by partioning predictions

Table 1. ECE (%, lower is better) with 15 bins on ResNet18, ResNet50 and ResNext101-32x8d (WRN). Comparison methods are the basic ResNet network and their SE-SNN version. Results after temperature scaling are shown in brackets.

Dataset	CIFAR-10			CIFAR-100		
	ResNet18	ResNet50	WRN	ResNet18	ResNet50	WRN
Baseline	11.2(14.1)	13.1(17.8)	8.3(**1.2**)	39.2(19.1)	41.4(36.9)	42.3(34.8)
SE-SNN	**9.7(2.8)**	10.7(5.8)	9.4(3.9)	38.2(12.0)	37.1(34.7)	37.0(22.8)
Proposed	10.0(7.7)	**4.9(1.8)**	**7.1**(1.9)	**20.5(11.2)**	**13.4(6.9)**	**2.0(3.2)**

Table 2. Comparison of defense methods against FGSM and PGD attacks on CIFAR-10 with a ResNet-18 (V2) backbone (Acc, %).

Method	FGSM	PGD
No defense	35.38	0.01
Adversarial Training	52.94	45.24
SeSNN	57.98	48.70
PNI	58.06	49.42
Proposed	**58.25**	**50.22**

into M equally-spaced bins and taking a weighted average of the bins' difference between accuracy and confidence. Specifically, ECE is calculated as follows:

$$ECE = \sum_{k=1}^{K} \frac{|B_k|}{n} \left|\text{acc}\left(B_k\right) - \text{conf}\left(B_k\right)\right| \qquad (8)$$

where n is the number of samples, B_k denotes the set of samples in bin k. acc and $conf$ denote the average accuracy and confidence of items in the specified bin respectively. The K is set to 15 in this experiment. Following Guo et al. (2017) we report results with and without additional temperature scaling.

Results We compared our proposed method with vanilla neural networks and SE-SNN model for two datasets: CIFAR-10 and CIFAR-100. Table 1 shows the ECE (with $K = 15$ bins) result, the bracketed numbers indicate the results after temperature scaling skills. It can be seen that the proposed method outperforms baseline and SE-SNN. Temperature scaling benefits all methods but our method still performs best overall.

4.3. Label Noise Robustness

Setting The label noise robustness of our proposed method is evaluated on MNIST (LeCun et al., 1998). We consider patterned label noise which is more common in practice. Specifically, one class label will be flipped to a different one at a strength p. The flipping pattern is fixed within each run but varies across different runs following (Hendrycks et al., 2018). For the network architecture, we follow Patrini et al. (2017) to train a neural network with

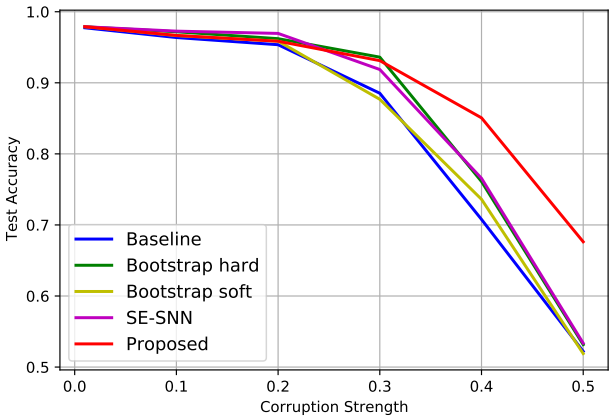

Figure 3. Learning with label-noise on MNIST.

three FC layers of dimension 128, 128 and the output layer, where the second FC layer is implemented as stochastic layer. Due to the randomness of noisy samples in selection and label reassignment, multiple runs of experiments are conducted.

For baseline methods, we choose bootstrap-hard and bootstrap-soft (Reed et al., 2015), and both of them use the predicted labels to refine the original labels that are potentially corrupted by noise. The difference lies on whether the updated label is binary or continuous. We also compare SE-SNN (Yu et al., 2021) which adds Gaussian stochastic layer on the second FC layers.

Results The results are shown in Figure. 3. We can see that our proposed method is competitive over listed methods. Especially when the corruption strength is larger than 0.3, we outperform the other methods.

5. Conclusion

In this paper, we propose a novel stochastic neural network model. Instead of the Gaussian distribution, we use implicit density for the distributions of both features and class prototype. The experiments show the proposed method has advantages for adversarial defense, model calibration, and label noise robustness.

References

Alemi, A. A., Fischer, I., Dillon, J., and Murphy, K. Deep variational information bottleneck. In *ICLR*, 2017.

Alemi, A. A., Fischer, I., and Dillon, J. V. Uncertainty in the variational information bottleneck. *CoRR:abs/1807.00906*, 2018.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In *ICML*, 2015.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., and Li, J. Boosting adversarial attacks with momentum. In *CVPR*, 2018.

Frogner, C., Mirzazadeh, F., and Solomon, J. Learning embeddings into entropic wasserstein spaces. In *ICLR*, 2019.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *ICML*, 2017.

He, Z., Rakin, A. S., and Fan, D. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In *CVPR*, 2019.

Hendrycks, D., Mazeika, M., Wilson, D., and Gimpel, K. Using trusted data to train deep networks on labels corrupted by severe noise. In *NIPS*, 2018.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *ICML*, 2015.

Huang, C.-W., Touati, A., Vincent, P., Dziugaite, G. K., Lacoste, A., and Courville, A. Stochastic neural network with kronecker flow. In *AISTATS*, 2020.

Kim, M. and Pavlovic, V. Recursive inference for variational autoencoders. In *NeurIPS*, 2020.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *ICLR*, 2014.

LeCun, Y., Léon Bottou, Y. B., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Madry, A., Schmidt, A. M. L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *CVPR*, 2017.

Mirza, M. and Osindero, S. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

Naeini, M. P., Cooper, G., and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, 2015.

Patrini, G., Rozza, A., Menon, A. K., Nock, R., and Qu, L. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*, 2017.

Rabin, J., Peyré, G., Delon, J., and Bernot, M. Wasserstein barycenter and its application to texture mixing. In *Scale Space and Variational Methods in Computer Vision*, 2011.

Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. Training deep neural networks on noisy labels with bootstrapping. In *ICLR Workshop*, 2015.

Tishby, N., Pereira, F. C., and Bialek, W. The information bottleneck method. *Allerton*, 1999.

Tomczak, J. and Welling, M. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pp. 1214–1223. PMLR, 2018.

Yu, T., Yang, Y., Li, D., Hospedales, T., and Xiang, T. Simple and effective stochastic neural networks. In *AAAI*, 2021.