Simple and Effective Stochastic Neural Networks

Tianyuan Yu[1], **Yongxin Yang**[1], **Da Li**[2,3], **Timothy Hospedales**[2,3], **Tao Xiang**[1]
[1]Center for Vision, Speech and Signal Processing, University of Surrey
[2]School of Informatics, University of Edinburgh
[3]Samsung AI Centre, Cambridge
{tianyuan.yu, yongxin.yang, t.xiang}@surrey.ac.uk, da.li1@samsung.com, t.hospedales@ed.ac.uk

Abstract

Stochastic neural networks (SNNs) are currently topical, with several paradigms being actively investigated including dropout, Bayesian neural networks, variational information bottleneck (VIB) and noise regularized learning. These neural network variants impact several major considerations, including generalization, network compression, robustness against adversarial attack and label noise, and model calibration. However, many existing networks are complicated and expensive to train, and/or only address one or two of these practical considerations. In this paper we propose a simple and effective stochastic neural network (SE-SNN) architecture for discriminative learning by directly modeling activation uncertainty and encouraging high activation variability. Compared to existing SNNs, our SE-SNN is simpler to implement and faster to train, and produces state of the art results on network compression by pruning, adversarial defense, learning with label noise, and model calibration.

Introduction

Stochastic neural networks (SNNs) have a long history. Recently various stochastic neural network instantiations have been topical in their applications to reducing overfitting (Gal and Ghahramani 2016; Neelakantan et al. 2016) and training data requirements (Garnelo et al. 2018), providing confidence estimates on predictions (Gal and Ghahramani 2016), enabling network compression (Dai et al. 2018), improving robustness to adversarial attack (Alemi et al. 2017), improving optimization (Neelakantan et al. 2016), generative modeling (Kingma and Welling 2014), and inputting or producing probability distributions (de Bie, Peyré, and Cuturi 2019; Frogner, Farzaneh, and Solomon 2019).

One of the most theoretically appealing stochastic neural network formulations is Bayesian neural networks, which place a prior distribution on the network weights (Graves 2011; Blundell et al. 2015; Ritter, Botev, and Barber 2018). However this usually leads to more complex learning and inference procedures that rely on variational approximations or sampling. Another line of work instead focuses on modeling uncertainty in neural network activations. Notably the variational information bottleneck (VIB) (Alemi et al. 2017) is motivated by information theoretic considerations (Tishby, Pereira, and Bialek 1999) to learn a hidden representation that carries maximum information about the output and minimum information about the input. Evaluating the required mutual information terms requires modeling probability distributions over activations rather than weights. Deep VIB (Alemi et al. 2017) leads to improved generalization, adversarial robustness and model compression algorithms (Dai et al. 2018). Furthermore, modeling noisy stochastic activations is often practically useful for improving exploration and local minima escape during optimization, generalization and adversarial robustness (You et al. 2019; Noh et al. 2017; Bishop 1995; Gulcehre et al. 2016), and in some cases can be linked back to Bayesian neural nets (Noh et al. 2017; Gal and Ghahramani 2016) when the noise added at each activation can be considered as a result of sampling from the weight posterior.

In this paper we propose a simple and effective stochastic neural network (SE-SNN) that models activation uncertainty through predicting a Gaussian mean and variance at each layer, which is then sampled during the forward pass. This is similar to the strategy used to model activation distributions in VAE (Kingma and Welling 2014) and VIB (Alemi et al. 2017). Differently, we then place a non-informative prior on the activation distribution and derive an activation regularizer that encourages high activation variability via preferring high-entropy activations. In conjunction with a discriminative learning loss, this means that the network is optimized for activation patterns that have high uncertainty while simultaneously being predictive of the target label. The interplay between these two objectives leads to several appealing capabilities in pruning, adversarial defense, learning with label noise, and improving model calibration. **Pruning**: Pruning aims to reduce the number of parameters in neural networks (NNs) while maintaining accuracy. Simultaneously optimizing for high per-activation variability/uncertainty and predictive accuracy leads to the network packing more entropy into the least significant neurons – so that the most crucial neurons are free to operate unperturbed. This leads to a simple pruning criterion based on each neuron's entropy value. **Adversarial defense**: Adversarial defense methods aim to increase NNs resilience to adversarial attack. By optimizing for both per-activation uncertainty and the network's predictive accuracy, a representation-level data augmentation policy is trained that perturbs the internal features during training for increased robustness (Alemi et al. 2017; You et al. 2019). **Label noise**:

Label noise is common in real-world datasets and tends to reduce learning performance. With SE-SNN, per-activation uncertainty can be easily aggregated to produce per-instance uncertainty. By optimizing for per-instance uncertainty and predictive accuracy, the network allocates the uncertainty the representation of the hard-to-classify instances so as to down-weight their influence on parameter learning. **Calibration**: In real-world decision-making, trustworthiness in the form of correctly calibrated confidence estimates, is often more critical than high accuracy – e.g., if the network is likely to make a mistake on a given instance, the decision can be delegated to a human. During inference, per-instance uncertainty can be considered as an indicator of confidence, and provide improved temperature scaling (Guo et al. 2017) to reduce calibration error.

To summarize, our contributions are: (1) A new simple yet effective stochastic neural network formulation. (2) We show that our SE-SNN has connections to VIB (Alemi et al. 2017), Dropout (Srivastava et al. 2014) and non-informative activation priors while being simpler to implement and faster to train, as well as impactful on a wider range of practical problems. (3) Comprehensive evaluations show excellent performance on pruning-based model compression, adversarial defense, label noise robust learning, and model calibration.

Related Work

Connection to VIB and Sparse VD Though we derive our max-entropy regularizer from the perspective of a non-informative activation prior, our work is closely related to VIB (Alemi et al. 2017) and sparse variational dropout (VD) (Molchanov, Ashukha, and Vetrov 2017), despite their different perspectives. Specifically, if we replace our infinite-variance Gaussian with a standard Gaussian, it becomes VIB (see Eq. 17 in (Alemi et al. 2017)). The max-entropy regularizer is also linked to Eq. 14 in Sparse VD (Molchanov, Ashukha, and Vetrov 2017), which also encourages large variance/entropy (at a different rate). But Sparse VD (Molchanov, Ashukha, and Vetrov 2017) is derived with a completely different motivation: It has an intuitive explanation that the regularizer corresponds to a sparsity prior on the weights. We note that enforcing uncertainty on activations rather than weights has a number of advantages: (i) The weight prior is intractable analytically, which leads to the fact that Sparse VD regularization is itself an approximation. (ii) Deriving the regularizer from a weight prior is unnecessarily complicated for the purpose of sparsifying the model. In contrast, our approach sidesteps the need to sample weights and avoids keeping multiple copies of the network, which can potentially improve efficiency (e.g., in memory usage).

Stochastic layers have been used in several other works in order to achieve better classification or regression accuracy. (Kingma, Salimans, and Welling 2015) proposes a generalization of Gaussian dropout where the dropout rates are learned, leading to higher classification accuracy. Natural-parameter networks (NPN) (Wang, Xingjian, and Yeung 2016) is a class of probabilistic neural networks where the input, target output, weights, and neurons can all be modeled by arbitrary exponential-family distributions (e.g., Poisson distributions for word counts) instead of being limited to Gaussian distributions, in order to help classification, regression, and representation learning tasks. To reduce computational cost, (Postels et al. 2019) approximates uncertainty estimates using a sampling-free approach and obtains better results on classification and regression tasks. Comparing to these, SE-SNN is again simpler and has wider-reaching impact.

Network Compression Network compression based on pruning typically uses heuristics based on pruning low-importance weights or low-activation neurons (Molchanov et al. 2017; Wen et al. 2016), often assisted by sparsity-enhancing priors such as lasso (Wen et al. 2016), log-uniform (Neklyudov et al. 2017), Jefferys and horse-shoe (Louizos, Ullrich, and Welling 2017). ℓ_0 (Louizos, Welling, and Kingma 2018) and ℓ_1 (Liu et al. 2017) norm-based regularizers get similar pruning effects. We avoid the complication of Bayesian learning of weights by proposing a simpler and direct activation prior that predisposes neurons towards deactivation unless necessary to solve the supervised task.

Adversarial Defense Our method is related to existing randomization-based defense methods (**?**Xie et al. 2018; Alemi et al. 2017; Ye and Zhu 2018; Liu et al. 2019). However unlike these studies which use a fixed distribution for noise (**?**Alemi et al. 2017), a learned model distribution for effective randomization (Liu et al. 2019), image perturbations (Xie et al. 2018) or a learned adversarial data-generating distribution (Ye and Zhu 2018), our randomization-based defense is both *learned*, and *data-dependent* since the variance at each layer is generated based on the output of the previous layer. The most relevant work to ours is RSE (Liu et al. 2018), which adds Gaussian noise layers to the NN to thwart gradient-based attacks. However, RSE uses a fixed constant noise variance. We introduce *learned* and *data-dependent* noise, which leads to improved performance.

Label Noise Robustness A number of existing label noise-robust deep learning approaches require a subset of noisy data to be reliably re-annotated (cleaned) to verify which samples contain noise (Lee et al. 2017; Jiang et al. 2018). In contrast, some others, similar to our SE-SNN, do not rely on additional human noise annotation. These methods address label noise by either iterative label correction via bootstrapping (Reed et al. 2015), adding additional layers on top of a softmax classification layer to estimate the noise pattern (Sukhbaatar et al. 2015; Goldberger and Ben-Reuven 2017), or loss correlation (Patrini et al. 2017). By allocating large uncertainty to outlying samples, our SE-SNN offers a simple and effective solution to learning with label noise. This approach to label noise robustness is appealingly simple in that it requires neither explicit detection of noisy samples, nor additional annotation. It is worth noting that label noise robustness has been largely ignored by existing SNNs such as VIB (Alemi et al. 2017) and sparse VD (Molchanov, Ashukha, and Vetrov 2017).

Calibration Modern neural networks for classification often suffer from poor calibration (Guo et al. 2017). I.e., the probabilities assigned to outputs are not reflective of the actual accuracy. While regularizers such as label-smoothing

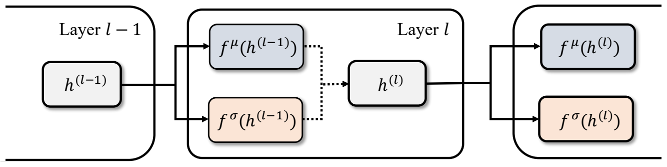


Figure 1: An illustration of the stochastic learning module in a SE-SNN. The output of layer l is sampled from the learned distribution defined by $f^{\mu}(h^{l-1})$ and $f^{\sigma}(h^{l-1})$.

(Müller, Kornblith, and Hinton 2019) improve calibration, they do not outperform calibrating the temperature of the output softmax based on a validation set (Guo et al. 2017). Dropout has also been used to model (Gal and Ghahramani 2016) uncertainty and improve predictive log-likelihood and RMSE. We show that the SE-SNN's uncertainty estimate can be leveraged to improve existing model calibration techniques in (Guo et al. 2017; Gal and Ghahramani 2016) .

Methodology

Stochastic Layers We consider a neural network discriminatively trained for a predictive task such as object recognition. Instead of computing fixed point estimates of feature vectors, we propose to use *stochastic* neurons. More specifically, for an input h, a layer will output a series of univariate distributions. By sampling from those distributions independently, we get a random output z. Finally we apply the nonlinear activation function $\psi(\cdot)$ to z and get the input for the next layer. In this study, we choose to use Gaussian distribution with parameterized mean and variance, which has been popularized by VAE (Kingma and Welling 2014) and VIB (Alemi et al. 2017) due to the ease of reparameterization. Formally, for the l-th layer, this forward-pass process can be written as (omitting neuron index for notation simplicity),

$$\begin{aligned}\mu^{(l)} &= f^{\mu}(h^{(l-1)}),\\ \sigma^{(l)} &= f^{\sigma}(h^{(l-1)}),\\ z^{(l)} &\sim \mathcal{N}(\mu^{(l)}, \sigma^{2(l)}),\\ h^{(l)} &= \psi(z^{(l)}).\end{aligned} \tag{1}$$

The above process is illustrated in Figure 1, where each standard deviation predictor f^{σ} uses softplus activation $f(x) = \log(1 + \exp(x))$ to ensure positivity.

Supervised Learning Loss We can choose to replace some or all intermediate layers of a vanilla neural network with such stochastic layers. For the final layer (i.e., the classifier) we opt for a standard linear layer, and the classification loss L_{cl} is the same as that of a vanilla neural network, e.g., cross-entropy. Since a Gaussian distribution is fully reparameterizable, the network can be trained end-to-end as long as the sampling process $z \sim \mathcal{N}(\mu, \sigma^2)$ is realized by $z = \mu + \epsilon \cdot \sigma$ where $\epsilon \sim \mathcal{N}(0, 1)$.

Max-entropy Regularization We place a non-informative prior on the produced Gaussian (denoted as $\mathcal{N}(\mu_2, \sigma_2^2)$). The non-informative prior is a Gaussian with arbitrary mean (μ_2) and infinite variance (σ_2^2). This reflects the prior that none of neurons is meaningful for predictive purposes. We minimize the KL divergence of the produced Gaussian from the prior Gaussian,

$$\begin{aligned}&\min_{\mu_1,\sigma_1}\,(\lim_{\sigma_2\to\infty}\mathrm{KL}(\mathcal{N}(\mu_1,\sigma_1^2)||\mathcal{N}(\mu_2,\sigma_2^2)))\\ \Rightarrow\ &\min_{\mu_1,\sigma_1}\,(\lim_{\sigma_2\to\infty}(\log\frac{\sigma_2}{\sigma_1}+\frac{\sigma_1^2+(\mu_1-\mu_2)^2}{2\sigma_2^2}-\frac{1}{2}))\\ \Rightarrow\ &\min_{\sigma_1}\,(\lim_{\sigma_2\to\infty}(\log\frac{\sigma_2}{\sigma_1}))\ \Rightarrow\ \min_{\sigma_1}\,(-\log\sigma_1).\end{aligned} \tag{2}$$

Eq. 2 suggests that we simply need to maximize the predicted standard deviation, or equivalently the entropy of the predicted Gaussian. Thus we call it a *max-entropy regularizer* Ω. It can be easily used in any existing neural network architecture:

$$\min_{\theta}\ -\log(\sigma(h|\theta)), \tag{3}$$

where $\sigma(h|\theta)$ denotes the predicted standard deviation of hidden unit h given the neuron uncertainty prediction parameter θ. For numerical safety, we introduce a margin b in the loss,

$$\min_{\theta}\ (b - \log(\sigma(h|\theta))^{+}. \tag{4}$$

This means that the regularization does not punish the model as long as the entropy is larger than a threshold b. Note that, this loss design is not absolutely necessary as the increment for σ shrinks (since the gradient is $-\frac{1}{\sigma}$) during training, and thus never reaches infinity with a finite number of updates. However, one can think of it as an early-stopping mechanism for this regularization term.

So far we have introduced the stochasticity to the smallest unit of a network, i.e., a single neuron. To compute the value of the regularizer over a mini-batch consisting of N training samples, we need to aggregate the entropy of multiple neurons and set the margin b on the aggregation. How to aggregate exactly is task-dependent, and we next provide some suggestions for the four tasks to be discussed next, including network pruning, adversarial defense, label noise defense, and model calibration.

Pruning For network pruning, we aggregate entropy over samples for each neuron, and then penalize if that neuron's aggregated entropy is low. To this end, the regularizer is formulated as:

$$\Omega(\theta) = \frac{1}{K}\sum_{j=1}^{K}(b - \frac{1}{N}\sum_{i=1}^{N}\log(\sigma_{i,j}))^{+}, \tag{5}$$

where $i = 1 \ldots N$ indexes the training samples, and $j = 1 \ldots K$ neurons in a layer (i.e., feature channels). This regularizer aims to make neurons very stochastic, to the point of compromising their reliability for computing a supervised learning task. Thus only those neurons that are most useful for the task get their entropy lowered and thus pay the regularization cost. Less useful neurons get their entropy maximized, allowing them to be detected and pruned after training.

Label Noise Different from pruning, we aim to identify uncertain samples. Therefore, we aggregate entropy over neurons for each sample, and prefer high sample-wise entropy, leading to

$$\Omega(\theta) = \frac{1}{N}\sum_{i=1}^{N}(b - \frac{1}{K}\sum_{j=1}^{K}\log(\sigma_{i,j}))^{+}. \quad (6)$$

With this regularizer, inlier samples pay the entropy cost in order to produce a clean feature for classification to satisfy the supervised learning loss. Outlier samples, caused by either label noise or being out-of-distribution, are anyway hard to classify; the regularizer naturally inflates their entropy since doing so does not impact the supervised learning loss. This high-variance representation in turn reduces their (negative) impact on network fitting.

Adversarial Defense To defend against adversarial samples, we aim to inflate entropy over both the neuron- and sample-axes to produce a highly stochastic model:

$$\Omega(\theta) = \frac{1}{N}\frac{1}{K}\sum_{i=1}^{N}\sum_{j=1}^{K}(b - \log(\sigma_{i,j}))^{+}. \quad (7)$$

This looks similar to VIB's adversarial defense (Alemi et al. 2017), but with the vital difference that our entropy regularizer is not restricted to the $\mathcal{N}(0,1)$ prior used in VIB. It can be seen as learning a layer-wise data augmentation policy, which turns out to be very useful for adversarial defense in practice.

Calibration For training an easy-to-calibrate model, we use the instance-wise entropy regularizer in Eq. 6, which we expect to assign more variance to instances that are hard to classify. Recall that SE-SNN produces data-dependent noise variance $\sigma_{i,j}$ for the i-th sample and j-th neuron, which can be considered an indicator of confidence. We now link this to temperature calibration, which is a simple and widely used method to calibrating modern neural networks (Guo et al. 2017). Temperature-scaling based calibration uses a scalar parameter T for all classes to raise the output entropy. We further exploit our variance estimate to produce better temperature calibrated confidences p_i from logits l_i as

$$p_i = \text{softmax}(l_i/(T\sum_{j=1}^{K}\sigma_{i,j})). \quad (8)$$

As per standard temperature calibration (Guo et al. 2017), T is optimized with respect to validation negative log likelihood. Neither standard temperature calibration, nor our generalization above change the accuracy, since the maximum output of the softmax is unchanged in both cases. However, SE-SNN enables better calibration by allowing the temperature scaling to depend on the per-instance variance, so hard-to-classify high-variance instances are scaled more strongly.

Training SE-SNN We train SE-SNN using a combination of the standard cross-entropy L_{cl} loss and our max-entropy regularizer Ω as

$$L = L_{cl} + \omega\Omega(\theta), \quad (9)$$

where ω balances the classification loss and max-entropy regularizer and Ω is defined according to the different applications as discussed above.

Experiments

Experiments are carried out to evaluate the efficacy of the proposed framework in four applications: neural network pruning, adversarial attack defense, learning with label noise, and model calibration.

Neural Network Pruning

Competitors We follow the architecture/dataset combinations used in most recent neural network pruning studies, including LeNet-5-Caffe[1] network on MNIST (LeCun et al. 1998), VGG-16 (Simonyan and Zisserman 2015) on CIFAR10 (Krizhevsky and Hinton 2009) and a variant of VGG–16 on CIFAR100. Under these settings, the proposed method is compared with the following contemporary state-of-the-art methods including Generalized Dropout (GD) (Srinivas and Babu 2016), Group Lasso (GL) (Wen et al. 2016), Sparse Variational Dropout (VD) (Molchanov, Ashukha, and Vetrov 2017), Structured Bayesian Pruning (SBP) (Neklyudov et al. 2017), Bayesian Compression with Group Normal Jeffreys Prior (BC-GNJ) and Group Horseshoe Prior (BC-GHS) (Louizos, Ullrich, and Welling 2017), Sparse L0 Regularization (L0) and L0 with separate λ for each layer (L0-sep) (Louizos, Welling, and Kingma 2018), Variational Information Bottleneck (VIBNet) (Dai et al. 2018), and Network Slimming (NS) (Liu et al. 2017).

Evaluation Metrics Following the majority of the existing evaluations, we monitor the test error while focusing on the following three compression/complexity metrics: (a) Model size: The ratio of nonzero weights in the compressed network versus the original model. (b) FLOPs: The number of floating point operations required to predict a label from an input image during test[2]. (c) Run-time memory footprint: The ratio of the space for storing hidden feature maps during run-time in the pruned network versus the original network.

Training While many existing studies remove redundant *weights*, we remove redundant *neurons* during compression. Therefore, we can use the pipeline proposed in (Molchanov et al. 2017). Specifically, after training the neural network, an initial batch pruning stage is followed by a loop of removing the least important neuron and fine-tuning. Since SE-SNN is designed to discount unimportant neurons through inflating their *pre-activation* variance, we find that the network achieves this by simultaneously assigning high-variance and negative mean. As a result, a large portion of redundant neurons never activate their RELU non-linearity. In the initial batch pruning stage, neuron inactivity thus provides a single-step pruning criterion before the iterative pruning begins (and one that is guaranteed not to affect the test accuracy since these neurons propagate no information). The pruning then enters the second stage where the least important neuron removal + fine-tuning loop continues until reaching the target trade-off between accuracy and model compression

[1]https://github.com/BVLC/caffe/tree/master/examples/mnist

[2]Following the setting in (Dai et al. 2018), we count each multiplication as a single FLOP and exclude additions, which is also consistent with most prior work

Methods	Model size (%)	FLOPs (Mil)	Memory (%)	Error (%)
GD (Srinivas and Babu 2016)	1.38	0.250	32.00	1.1
GL (Wen et al. 2016)	23.69	0.201	19.35	1.0
VD (Molchanov, Ashukha, and Vetrov 2017)	9.29	0.660	60.78	1.0
SBP (Neklyudov et al. 2017)	19.66	0.213	21.15	**0.9**
BC-GNJ (Louizos, Ullrich, and Welling 2017)	0.95	0.283	35.03	1.0
BC-GHS (Louizos, Ullrich, and Welling 2017)	**0.64**	0.153	22.80	1.0
L0 (Louizos, Welling, and Kingma 2018)	8.92	1.113	85.82	**0.9**
L0-sep (Louizos, Welling, and Kingma 2018)	1.08	0.389	40.36	1.0
VIBNet (Dai et al. 2018)	0.83	0.094	15.55	1.0
SE-SNN	2.35	**0.061**	**11.08**	**0.9**

Table 1: Compression results on MNIST using LeNet-5-Caffe

Methods	Model size (%)	FLOPs (Mil)	Memory (%)	Error (%)
BC-GNJ (Louizos, Ullrich, and Welling 2017)	6.57	141.50	81.68	8.6
BC-GHS (Louizos, Ullrich, and Welling 2017)	5.40	121.90	74.82	9.0
VIBNet (Dai et al. 2018)	5.30	70.63	49.57	8.8(8.5)
SE-SNN	**2.57**	**53.61**	**49.41**	**8.0**

Table 2: Compression results on CIFAR10 using VGG16

objectives. Since one neuron/channel is removed at each iteration, the accuracy never drops sharply. We set the regularizer weight ω (Eq. 9) and margin b (Eq. 5) as 0.01 and 4 in all experiments, respectively.

Testing During testing, only the mean of the learned distribution is passed between layers. So there are no additional parameters and inference cost compared to the network's deterministic counterpart. The distribution generation branches are only used during training to identify redundant neurons.

Results on MNIST The most commonly used benchmark and architecture is MNIST with LeNet-5-Caffe. We follow the standard training and testing protocols. The results are shown in Table 1. It is clear that SE-SNN achieves the best performance on FLOPs, run-time memory footprint, and test error. In terms of model size, our model is only comparable with the state of art. This is because it does not prune the linear layer as much as other methods. Instead, it focuses on pruning the convolutional filters, hence the excellent performance on FLOPS and memory footprint.

Results on CIFAR10 For CIFAR10, several VGG16 variants and training protocols have been proposed in different works. We use the standard VGG16 architecture but change the dimension of linear layers from 4096 to 512, as in (Louizos, Ullrich, and Welling 2017; Dai et al. 2018). The results in Table 2 compare our method with (Louizos, Ullrich, and Welling 2017; Dai et al. 2018). The error rate for VIBNet in parentheses was obtained by further fine-tuning the pruned architecture. Our model achieves the best performance across all the evaluation metrics and error rates.

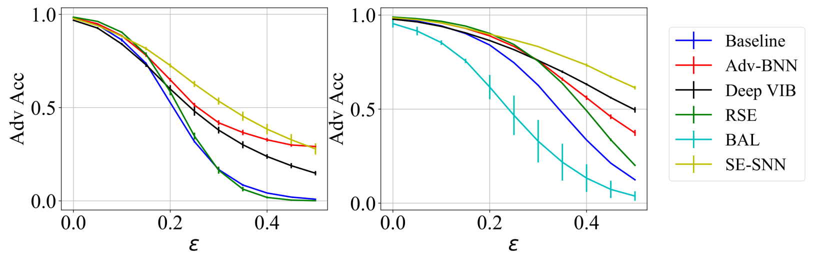
Results on CIFAR100 We compare with the study in (Liu et al. 2017), which uses the same VGG16 variant that replaces two fully connected layers with three convolutional layers. This architecture improves accuracy at the expense of FLOPs and memory. The results in Table 3 show that our model produces the best compression result while maintaining comparable accuracy. Note that the 26.2% error rate achieved by our model is identical to that of the original network. So if the accuracy drop is used as a compression metric, our model is as good as any competitor.

Adversarial Defense

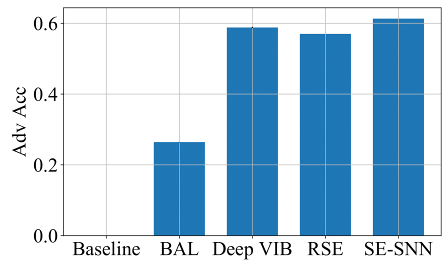
Experimental Setting Following (Alemi et al. 2017), we focus on two types of adversarial attacks: Fast Gradient Sign (FGS) (Goodfellow, Shlens, and Szegedy 2015) and an optimization-based attack CW-L2 (Carlini and Wagner 2017; Athalye, Carlini, and Wagner 2018).We evaluate untargeted FGS attacks with attack magnitude ϵ ranging from 0.0 to 0.5 and untargeted CW-L2 attack on models trained on MNIST. We use a popular architecture including three FC layers with 1024, 1024 and 256 output neurons respectively. The third FC layer is implemented as a stochastic layer. We set the regularizer weight and margin as 0.1 and 16, respectively. Results averaged over 20 runs are reported. We compare against the original (undefended) network, termed as 'Baseline', Deep VIB (Alemi et al. 2017) using the variational information bottleneck, Bayesian Adversarial Learning (BAL) (Ye and Zhu 2018) putting a distribution on the adversarial data-generating process, Adv-BNN (Liu et al. 2019) learning a BNN to incorporate the effective randomness and using adversarial training to seek the best model distribution, and RSE (Liu et al. 2018) adding random noise layer in the neural network and using the ensemble prediction over random noises to stabilise the performance. Since our model is stochastic, we follow the best practice recommended in (Athalye, Carlini, and Wagner 2018) for attacking stochastic

Methods	Model size (%)	FLOPs (Mil)	Memory (%)	Error (%)
NS-Single (Liu et al. 2017)	24.90	250.50	-	26.5
NS-Best (Liu et al. 2017)	20.80	214.80	-	26.0
VIBNet (Dai et al. 2018)	15.08	203.10	73.80	25.9(**25.7**)
SE-SNN	**14.93**	**181.31**	**70.16**	26.2

Table 3: Compression results on CIFAR100 using a VGG16 variant



(a) Untargeted FGS (L: normal training, R: adversarial training)



(b) Untargeted CW-L2

Figure 2: Adversarial defense accuracy (mean + standard deviation) under untargeted FGS and CW attacks on MNIST

models: We compute the expected gradient over multiple stochastic samples for each input when constructing attacks. This is because using the expected gradient over multiple posterior samples produces a better gradient estimator for the attacker, allowing it to generate samples that are much harder to defend against (Athalye, Carlini, and Wagner 2018). For the FGS attack, we evaluate two settings, namely normal training and adversarial training, the latter of which generates and uses FGS attack samples during training to increase adversarial robustness.

Results From the comparative results shown in Figure 2(a), we can see that our SE-SNN outperforms the competing defense methods over a range of FGS attack strengths when trained with adversarial attack samples (Figure 2(a)(R)) or with only normal samples (Figure 2(a)(L)). The advantage of SE-SNN is particularly pronounced when the attack magnitude is large. We can also see from Figure 2(b) that under the stronger CW attacks the Baseline now fails completely. In this case SE-SNN provides the most effective defense. It is worth pointing out that, unlike BAL (Ye and Zhu 2018), which learns their models with explicit adversarial sampling, our SE-SNN can also work with training with only normal samples - Figure 2(a) suggests that our SE-SNN beats BAL even without being trained with adversarial attack samples (comparing SE-SNN in Figure 2(a)(L) to BAL in Figure 2(a)(R)). It is noteworthy that though RSE provides strong defense against CW attacks, it performs very poorly on FGS. We attribute this to the fact that their fixed variance parameter is hand picked for CW. We tried re-tuning their σ for FGS, but failed. In contrast, our SE-SNN's ability to learn the variance σ leads to good performance against both attacks, and our regularizer hyperparameters ω, β are fixed across all adversarial experiments, without the need for re-tuning for different attacks.

Robustness against Label Noise

Experimental Setting The label noise robustness of our SE-SNN is evaluated on digit recognition using MNIST. 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 for 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 two FC layers of dimension 128, where both FC layers are implemented as stochastic layers. Due to the randomness of noisy samples in selection and label reassignment, multiple runs of experiments are conducted.

Competitors *Baseline:* This is the original network without adding our stochastic layers. *Bootstrap_hard* and *Bootstrap_soft* (Reed et al. 2015): Both iteratively use the model-predicted labels to refine the original labels that are potentially corrupted by noise. They differ in whether the updated label is binary or continuous. *Forward Correction* (Patrini et al. 2017): This model predicts the label corruption matrix (capturing the label flipping pattern) by first training a classifier on the noisy labels followed by corruption matrix estimation using the resulting softmax probabilities. The estimated matrix is then employed to regularize the retraining of the model. We stick to the original setting, which uses the $argmax$ at the 97th percentile of softmax probabilities for label noise detection. Note that unlike the competitors, our SE-SNN does not have any additional steps to refine the label or prune the training samples. Having said that, it can be easily combined with a label refinement procedure such as the corruption matrix based one in Forward Correction (Patrini et al. 2017).

Results The results are shown in Figure 3. We can see that even without any label correction/refinement procedure, our

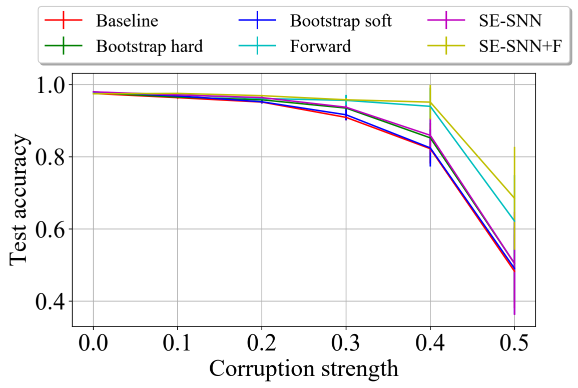


Figure 3: Label noise results on MNIST

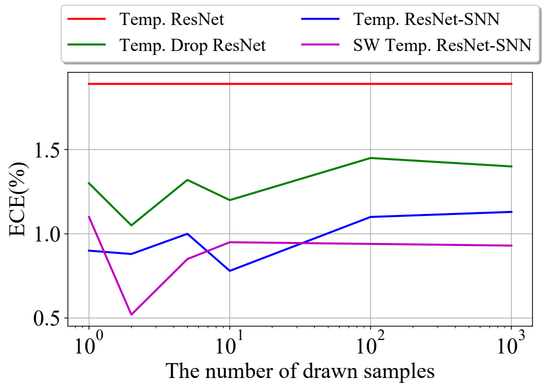


Figure 4: Calibration Results with ResNet on CIFAR-100

Dataset	CIFAR-10				CIFAR-100			
Model	ResNet	Res-SNN	WRN	WRN-SNN	ResNet	Res-SNN	WRN	WRN-SNN
Temp.	1.39	1.07	0.90	0.79	1.89	1.43	2.49	1.86
Temp. Drop	1.09	-	0.72	-	1.13	-	2.03	-
SW Temp.	-	**0.90**	-	**0.39**	-	**0.93**	-	**1.64**

Table 4: ECE (%) with 15 bins on ResNet-110 (ResNet), Wide-ResNet-16-2(WRN), and their corresponding SE-SNN version (e.g., Res-SNN for ResNet) with temperature scaling (Temp.), temperature scaling of dropout (Temp. Drop) in (Gal and Ghahramani 2016), and sample-wise temperature scaling (SW Temp.) using the per-sample variance learned by SE-SNN.

SE-SNN is already very competitive – it is only beaten by Forward Correction when the corruption strength (percentage of samples with label noise) is large (> 0.2). Once our SE-SNN is combined with the same procedure adopted by Forward Correction (SE-SNN+Forward), we achieve the best result.

Calibration

Experimental Setting Following the architecture and dataset used in (Guo et al. 2017), we train ResNet-110 and Wide-ResNet-16-2 on CIFAR-10 and CIFAR-100. For SE-SNN, a stochastic layer is added before the classification layer in these two neural networks respectively. Expected Calibration Error (ECE) (Naeini, Cooper, and Hauskrecht 2015) is the widely-used evaluation matrix to measure the calibration error. ECE approximates the difference in expectation between confidence and accuracy by partitioning predictions into K equally-spaced bins and taking a weighted average of the bins' difference between accuracy and confidence. Specifically, ECE computes:

$$ECE = \sum_{k=1}^{K} \frac{|B_k|}{n} |acc(B_k) - conf(B_k)|, \qquad (10)$$

where n is the number of samples, B_k denotes the set of samples in bin k. acc and $conf$ respectively denote the average accuracy and confidence (e.g., Eq. 8) of items in the specified bin.

Results Using SE-SNN we can draw multiple samples from the distribution corresponding to a single input, and use these to produce an average softmax result for prediction. The ability to average over multiple samples should lead to an improved calibration (as per other approaches to Bayesian neural networks (Gal and Ghahramani 2016), which is denoted as 'Temp. Drop'.), even before using our novel sample-wise temperature scaling strategy in Eq 8. Fig. 4 shows the ECE result for ResNet-110 on CIFAR100 as a function of the number of samples drawn. It can be seen that, as a function of the number of samples drawn to estimate probability posterior p_i, performance stabilizes after 10 samples. Table 4 summarizes the results for CIFAR-10/100 using ResNet-110 and WideResNet WRN-16-2 and 100 samples. We can see that: (1) Temperature scaled ResNet-SNN improves performance compared to temperature scaled vanilla ResNet. (2) Our sample-wise temperature scaling ('SW Temp') further improves performance compared to vanilla temperature scaling, reducing the calibration error by 12-50% relatively to vanilla temperature scaling. (3) Our method is clearly more effective than the dropout temperature scaling based one in (Gal and Ghahramani 2016) ('SW Temp' vs. 'Temp. Drop').

Conclusion

We proposed a simple and effective stochastic neural network framework. Our model is related to VIB and variational dropout, but provides a simpler and more direct realization via neuron regularization by a non-informative activation prior. Our extensive experiments show that this simple framework has diverse benefits for network pruning, adversarial defense, label noise robust learning, and model calibration.

References

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

Athalye, A.; Carlini, N.; and Wagner, D. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *ICML*, 274–283.

Bishop, C. M. 1995. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation* 7(1): 108–116.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight Uncertainty in Neural Networks. In *ICML*, 1613–1622.

Carlini, N.; and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *IEEESSP*, 39–57.

Dai, B.; Zhu, C.; Guo, B.; and Wipf, D. 2018. Compressing Neural Networks using the Variational Information Bottleneck. In *ICML*, 1135–1144.

de Bie, G.; Peyré, G.; and Cuturi, M. 2019. Stochastic Deep Networks. In *ICML*, 1556–1565.

Frogner, C.; Farzaneh, M.; and Solomon, J. 2019. Learning Embeddings into Entropic Wasserstein Spaces. In *ICLR*.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *ICML*, 1050–1059.

Garnelo, M.; Rosenbaum, D.; Maddison, C.; Ramalho, T.; Saxton, D.; Shanahan, M.; Teh, Y. W.; Rezende, D.; and Eslami, S. M. A. 2018. Conditional Neural Processes. In *ICML*, 1704–1713.

Goldberger, J.; and Ben-Reuven, E. 2017. Training deep neural-networks using a noise adaptation layer. In *ICLR*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *ICLR*.

Graves, A. 2011. Practical Variational Inference for Neural Networks. In *NIPS*, 2348–2356.

Gulcehre, C.; Moczulski, M.; Denil, M.; and Bengio, Y. 2016. Noisy Activation Functions. In *ICML*, 3059–3068.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On calibration of modern neural networks. In *ICML*, 1321–1330.

Hendrycks, D.; Mazeika, M.; Wilson, D.; and Gimpel, K. 2018. Using trusted data to train deep networks on labels corrupted by severe noise. In *NIPS*, 10477–10486.

Jiang, L.; Zhou, Z.; Leung, T.; Li, L.-J.; and Fei-Fei, L. 2018. MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels. In *ICML*, 2304–2313.

Kingma, D. P.; Salimans, T.; and Welling, M. 2015. Variational dropout and the local reparameterization trick. In *NIPS*, 2575–2583.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In *ICLR*.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *IEEE* 86(11): 2278–2324.

Lee, K.-H.; He, X.; Zhang, L.; and Yang, L. 2017. CleanNet: Transfer Learning for Scalable Image Classifier Training with Label Noise. In *ICCV*.

Liu, X.; Cheng, M.; Zhang, H.; and Hsieh, C.-J. 2018. Towards robust neural networks via random self-ensemble. In *ECCV*, 369–385.

Liu, X.; Li, Y.; Wu, C.; and Hsieh, C.-J. 2019. Adv-BNN: Improved Adversarial Defense through Robust Bayesian Neural Network. In *ICLR*.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning efficient convolutional networks through network slimming. In *ICCV*, 2736–2744.

Louizos, C.; Ullrich, K.; and Welling, M. 2017. Bayesian compression for deep learning. In *NIPS*, 3290–3300.

Louizos, C.; Welling, M.; and Kingma, D. P. 2018. Learning Sparse Neural Networks through L_0 Regularization. In *ICLR*.

Molchanov, D.; Ashukha, A.; and Vetrov, D. 2017. Variational dropout sparsifies deep neural networks. In *ICML*, 2498–2507.

Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; and Kautz, J. 2017. Pruning convolutional neural networks for resource efficient inference. In *ICLR*.

Müller, R.; Kornblith, S.; and Hinton, G. E. 2019. When does label smoothing help? In *NIPS*, 4696–4705.

Naeini, M. P.; Cooper, G.; and Hauskrecht, M. 2015. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, 2901–2907.

Neelakantan, A.; Vilnis, L.; Le, Q. V.; Kaiser, L.; Kurach, K.; Sutskever, I.; and Martens, J. 2016. Adding Gradient Noise Improves Learning for Very Deep Networks. In *ICLR Workshop*.

Neklyudov, K.; Molchanov, D.; Ashukha, A.; and Vetrov, D. P. 2017. Structured bayesian pruning via log-normal multiplicative noise. In *NIPS*, 6778–6787.

Noh, H.; You, T.; Mun, J.; and Han, B. 2017. Regularizing Deep Neural Networks by Noise: Its Interpretation and Optimization. In *NIPS*, 5115–5124.

Patrini, G.; Rozza, A.; Krishna Menon, A.; Nock, R.; and Qu, L. 2017. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*, 1944–1952.

Postels, J.; Ferroni, F.; Coskun, H.; Navab, N.; and Tombari, F. 2019. Sampling-free Epistemic Uncertainty Estimation Using Approximated Variance Propagation. In *ICCV*, 2931–2940.

Reed, S. E.; Lee, H.; Anguelov, D.; Szegedy, C.; Erhan, D.; and Rabinovich, A. 2015. Training Deep Neural Networks on Noisy Labels with Bootstrapping. In *ICLR Workshop*.

Ritter, H.; Botev, A.; and Barber, D. 2018. A Scalable Laplace Approximation for Neural Networks. In *ICLR*.

Simonyan, K.; and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Srinivas, S.; and Babu, R. V. 2016. Generalized dropout. *CoRR* abs/1611.06791.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *JMLR* 15: 1929–1958.

Sukhbaatar, S.; Bruna, J.; Paluri, M.; Bourdev, L.; and Fergus, R. 2015. Training convolutional networks with noisy labels. In *ICLR*.

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

Wang, H.; Xingjian, S.; and Yeung, D.-Y. 2016. Natural-parameter networks: A class of probabilistic neural networks. In *NIPS*, 118–126.

Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016. Learning structured sparsity in deep neural networks. In *NIPS*, 2082–2090.

Xie, C.; Wang, J.; Zhang, Z.; Ren, Z.; and Yuille, A. L. 2018. Mitigating adversarial effects through randomization. In *ICLR*.

Ye, N.; and Zhu, Z. 2018. Bayesian Adversarial Learning. In *NIPS*, 6892–6901.

You, Z.; Ye, J.; Li, K.; and Wang, P. 2019. Adversarial Noise Layer: Regularize Neural Network By Adding Noise. In *ICIP*, 909–913.