Learning for Single-Shot Confidence Calibration in Deep Neural Networks through Stochastic Inferences

Seonguk Seo*[1] Paul Hongsuck Seo*[1,2] Bohyung Han[1]
[1]Computer Vision Lab., ECE & ASRI, Seoul National University, Korea
[2]Computer Vision Lab., POSTECH, Korea
{seonguk, bhhan}@snu.ac.kr hsseo@postech.ac.kr

Abstract

We propose a generic framework to calibrate accuracy and confidence of a prediction in deep neural networks through stochastic inferences. We interpret stochastic regularization using a Bayesian model, and analyze the relation between predictive uncertainty of networks and variance of the prediction scores obtained by stochastic inferences for a single example. Our empirical study shows that the accuracy and the score of a prediction are highly correlated with the variance of multiple stochastic inferences given by stochastic depth or dropout. Motivated by this observation, we design a novel variance-weighted confidence-integrated loss function that is composed of two cross-entropy loss terms with respect to ground-truth and uniform distribution, which are balanced by variance of stochastic prediction scores. The proposed loss function enables us to learn deep neural networks that predict confidence calibrated scores using a single inference. Our algorithm presents outstanding confidence calibration performance and improves classification accuracy when combined with two popular stochastic regularization techniques—stochastic depth and dropout—in multiple models and datasets; it alleviates overconfidence issue in deep neural networks significantly by training networks to achieve prediction accuracy proportional to confidence of prediction.

1. Introduction

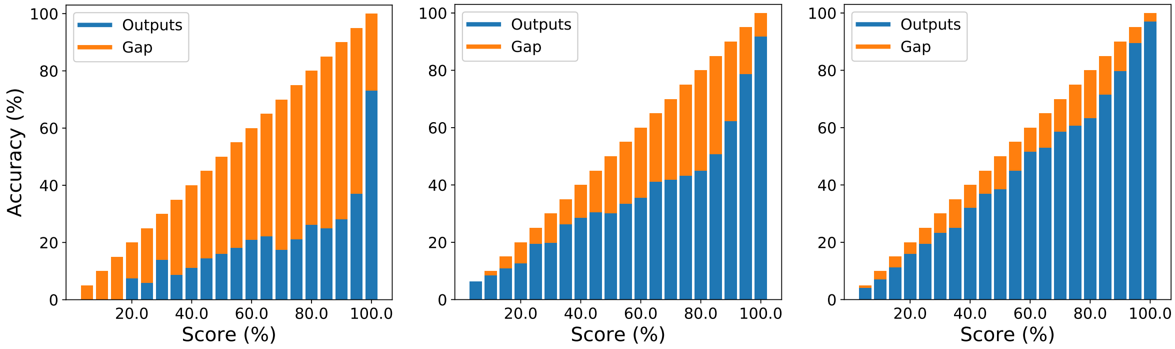
Deep neural networks have achieved remarkable performance in various tasks, but have critical limitations in reliability of their predictions. One example is that inference results are often overly confident even for unseen or ambiguous examples. Since many practical applications including medical diagnosis, autonomous driving, and machine inspection require accurate uncertainty estimation as well as high prediction score for each inference, such an overconfidence issue makes deep neural networks inappropriate to be deployed for real-world problems in spite of their impressive accuracy.

Regularization is a common technique in training deep neural networks to avoid overfitting problems and improve generalization performance [10, 11, 24]. Although regularization is effective to learn robust models, its objective is not directly related to generating score distributions aligned with uncertainty of predictions. Hence, existing deep neural networks are often poor at calibrating prediction accuracy and confidence.

Our goal is to learn deep neural networks that are able to estimate uncertainty of each prediction while maintaining accuracy. In other words, we propose a generic framework to calibrate prediction score (confidence) with accuracy in deep neural networks. The main idea of our algorithm starts with an observation that the variance of prediction scores measured from multiple stochastic inferences is highly correlated with the accuracy and confidence of the average prediction. We also show that a Bayesian interpretation of stochastic regularizations such as stochastic depth and dropout leads to the consistent conclusion with the observation. By using the empirical observation with the theoretical interpretation, we design a novel loss function to enable a deep neural network to predict confidence-calibrated scores based only on a single prediction, without multiple stochastic inferences. Our contribution is summarized as

- We provide a generic framework to estimate uncertainty of a prediction based on stochastic inferences in deep neural networks, which is supported by empirical observations and theoretical analysis.
- We propose a novel variance-weighted confidence-integrated loss function in a principled way, which enables networks to produce confidence-calibrated predictions even without performing stochastic inferences and introducing hyper-parameters.
- The proposed framework presents outstanding performance to reduce overconfidence issue and estimate ac-

*Equal contribution



(a) Baseline (ECE = 0.346)

(b) CI[Oracle] (ECE = 0.122)

(c) VWCI (ECE = 0.053)

Figure 1. Reliability diagrams of VGG-16 models trained with baseline, CI (ours) and VWCI (ours) losses in Tiny ImageNet dataset. This diagram shows expected accuracy as a function of confidence, *i.e.*, classification score. ECE (Expected Calibration Error) denotes the average gap between confidence and expected accuracy. The proposed algorithm (VWCI) achieves well-calibrated results compared to the baseline and the best estimate by a simpler version of ours (CI).

curate uncertainty in various combinations of network architectures and datasets.

The rest of the paper is organized as follows. We review the prior research and describe the theoretical background in Section 2 and 3, respectively. Section 4 presents our confidence calibration algorithm through stochastic inferences, and Section 5 demonstrates experimental results.

2. Related Work

Uncertainty modeling and estimation in deep neural networks is a critical problem and receives growing attention from machine learning community. Bayesian approach is a common tool to provide a mathematical framework for uncertainty estimation. However, the exact Bayesian inference is not tractable in deep neural networks due to its high computational cost, and various approximate inference techniques—MCMC [17], Laplace approximation [14] and variational inference [1, 4, 8, 20]—have been proposed. Recently, a Bayesian interpretation of multiplicative noise is employed to estimate uncertainty in deep neural networks [3, 15]. Besides, there are several approaches outside Bayesian modeling, *e.g.*, post-processing [5, 18, 22, 28] and deep ensembles [12]. All the post-processing methods require a hold-out validation set to adjust prediction scores after training, and the ensemble-based technique employs multiple models to estimate uncertainty.

Stochastic regularization is a well-known technique to improve generalization performance by injecting random noise to deep neural networks. The most notable method is dropout [24], which rejects a subset of hidden units in a layer based on Bernoulli random noise. There exist several variants, for example, dropping weights [27] or skipping layers [10]. Most stochastic regularization methods perform stochastic inferences during training, but make deterministic predictions using the full network during testing. On the contrary, we also employ stochastic inferences to obtain diverse and reliable outputs during testing.

Although the following works do not address uncertainty estimation, their main idea is related to our objective more or less. Label smoothing [25] encourages models to be less confident, by preventing a network from assigning the full probability to a single class. A similar loss function is discussed to train confidence-calibrated classifiers in [13], but it focuses on how to discriminate in-distribution and out-of-distribution examples, rather than estimating uncertainty or alleviating miscalibration of in-distribution examples. On the other hand, [21] claims that blind label smoothing and penalizing entropy enhances accuracy by integrating loss functions with the same concept with [13, 25], but its improvement is marginal in practice.

3. Preliminaries

This section describes a Bayesian interpretation of stochastic regularization in deep neural networks, and discusses the relationship between stochastic regularization and uncertainty modeling.

3.1. Stochastic Methods for Regularizations

A popular class of regularization techniques is stochastic regularization, which introduces random noise for perturbing network structures. Our approach focuses on the multiplicative binary noise injection, where random binary noise is applied to the inputs or weights by elementwise multiplication, since such stochastic regularization techniques are widely used [10, 24, 27]. Note that input perturbation can be reformulated as weight perturbation. For example, dropout—binary noise injection to activations—is interpretable as weight perturbation that masks out all the weights associated with the dropped inputs. Therefore, if a classification network modeling $p(y|x,\theta)$ with parameters θ is trained with stochastic regularization methods by minimizing cross entropy, the loss function is defined by

$$\mathcal{L}_{\text{SR}}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log p\left(y_i|x_i, \hat{\omega}_i\right), \tag{1}$$

where $\hat{\omega}_i = \theta \odot \epsilon_i$ is a set of perturbed parameters by elementwise multiplication with random noise sample $\epsilon_i \sim p(\epsilon)$, and $(x_i, y_i) \in \mathcal{D}$ is a pair of input and output in training dataset $\mathcal{D}$.

At inference time, the network is parameterized by the expectation of the perturbed parameters, $\Theta = \mathbb{E}[\omega] = \theta \odot \mathbb{E}[\epsilon]$, to predict an output $\hat{y}$, which is given by

$$\hat{y} = \arg\max_{y} p\left(y|x, \Theta\right). \tag{2}$$

3.2. Bayesian Modeling

Given the dataset $\mathcal{D}$ with N examples, Bayesian objective is to estimate the posterior distribution of the model parameter, denoted by $p(\omega|\mathcal{D})$, to predict a label y for an input x, which is given by

$$p(y|x,\mathcal{D}) = \int_{\omega} p(y|x,\omega)p(\omega|\mathcal{D})d\omega. \tag{3}$$

A common technique for the posterior estimation is variational approximation, which introduces an approximate distribution $q_\theta(\omega)$ and minimizes Kullback-Leibler (KL) divergence with the true posterior $D_{\text{KL}}(q_\theta(\omega)||p(\omega|D))$ as follows:

$$\begin{aligned}\mathcal{L}_{\text{VA}}(\theta) = &-\sum_{i=1}^{N}\int_{\omega} q_\theta(\omega)\log p(y_i|x_i,\omega)d\omega \\ &+ D_{\text{KL}}(q_\theta(\omega)||p(\omega)).\end{aligned} \tag{4}$$

The intractable integration and summation over the entire dataset in Eq. (4) is approximated by Monte Carlo method and mini-batch optimization, resulting in

$$\begin{aligned}\hat{\mathcal{L}}_{\text{VA}}(\theta) = &-\frac{N}{MS}\sum_{i=1}^{M}\sum_{j=1}^{S}\log p\left(y_i|x_i,\hat{\omega}_{i,j}\right) \\ &+ D_{\text{KL}}\left(q_\theta(\omega)||p(\omega)\right),\end{aligned} \tag{5}$$

where $\hat{\omega}_{i,j} \sim q_\theta(\omega)$ is a sample from the approximate distribution, S is the number of samples, and M is the size of a mini-batch. Note that the first term is data likelihood and the second term is divergence of the approximate distribution with respect to the prior distribution.

3.3. Bayesian View of Stochastic Regularization

Suppose that we train a classifier with ℓ_2 regularization by a stochastic gradient descent method. Then, the loss function in Eq. (1) is rewritten as

$$\hat{\mathcal{L}}_{\text{SR}}(\theta) = -\frac{1}{M}\sum_{i=1}^{M}\log p\left(y_i|x_i,\hat{\omega}_i\right) + \lambda||\theta||_2^2, \tag{6}$$

where ℓ_2 regularization is applied to the deterministic parameters θ with weight λ. Optimizing this loss function is equivalent to optimizing Eq. (5) if there exists a proper prior $p(\omega)$ and $q_\theta(\omega)$ is approximated as a Gaussian mixture distribution [3]. Note that [3] casts dropout training as an approximate Bayesian inference. Thus, we can interpret training with stochastic depth [10] within the same framework by a simple modification. (See our supplementary document for the details.) Then, the predictive distribution of a model trained with stochastic regularization is approximately given by

$$\hat{p}(y|x,\mathcal{D}) = \int_{\omega} p(y|x,\omega)q_\theta(\omega)d\omega. \tag{7}$$

Following [3] and [26], we estimate the predictive mean and uncertainty using a Monte Carlo approximation by drawing parameter samples $\{\hat{\omega}_i\}_{i=1}^{T}$ as

$$\mathbb{E}_{\hat{p}}[y=c] \approx \frac{1}{T}\sum_{i=1}^{T}\hat{p}(y=c|x,\hat{\omega}_i), \tag{8}$$

$$\text{Cov}_{\hat{p}}[\mathbf{y}] \approx \mathbb{E}_{\hat{p}}[\mathbf{y}\mathbf{y}^\intercal] - \mathbb{E}_{\hat{p}}[\mathbf{y}]\mathbb{E}_{\hat{p}}[\mathbf{y}]^\intercal, \tag{9}$$

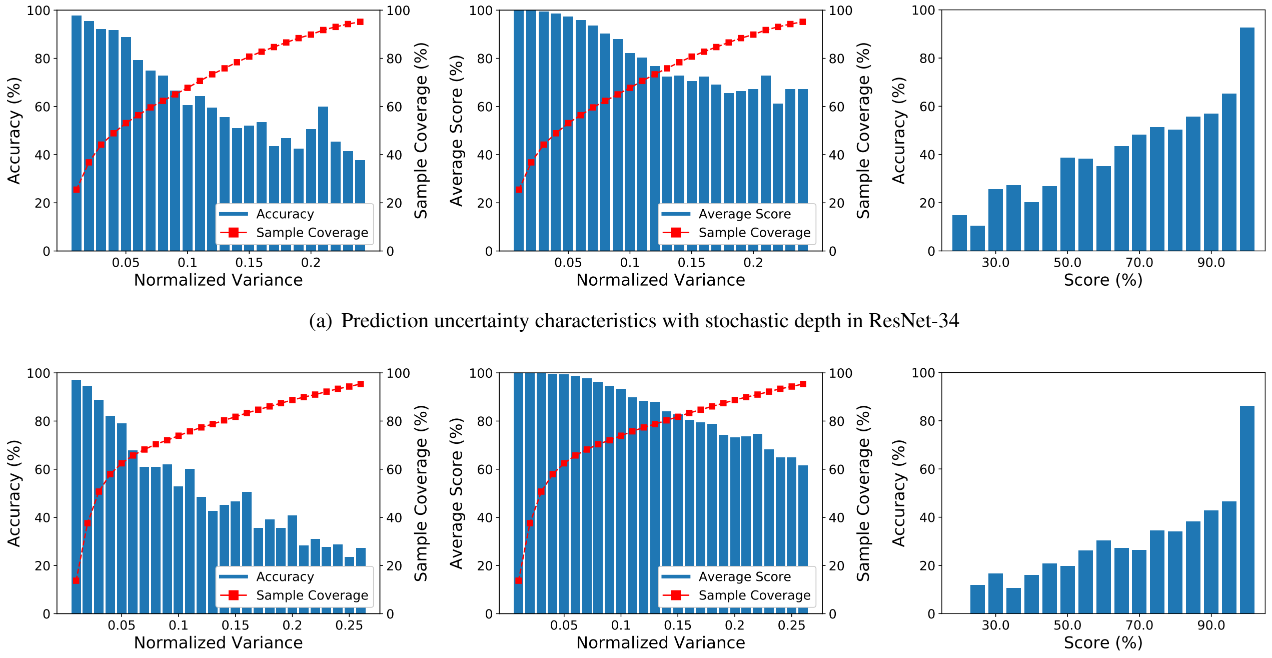
where $\mathbf{y} = (y_1, \ldots, y_C)^\intercal$ denotes a score vector of C class labels. Eq. (8) and Eq. (9) mean that the average prediction and its predictive uncertainty can be estimated from multiple stochastic inferences.

4. Methods

We present a novel confidence calibration technique for prediction in deep neural networks, which is given by a variance-weighted confidence-integrated loss function. We present our observation that variance of multiple stochastic inferences is closely related to accuracy and confidence of predictions, and provide an end-to-end training framework for confidence self-calibration. Then, we show that the prediction accuracy and uncertainty are directly accessible from a predicted score from a single forward pass.

4.1. Empirical Observations

Eq. (9) implies that the variation of models results in the variance of multiple stochastic predictions for a single example. Figure 2 presents how the variance of multiple stochastic inferences given by stochastic depth or dropout is related to the accuracy and confidence of the corresponding average prediction, where the confidence is measured by the maximum score of the average prediction. In the figure, the accuracy and the score of each bin are computed with



(a) Prediction uncertainty characteristics with stochastic depth in ResNet-34

(b) Prediction uncertainty characteristics with dropout in VGGNet with 16 layers

Figure 2. Uncertainty observed from multiple stochastic inferences with two stochastic regularization methods, (a) stochastic depth and (b) dropout. We present (left, middle) tendency of accuracy and score of the average prediction with respect to normalized variance of stochastic inferences and (right) relation between score and accuracy. In regularization methods, average accuracy and score drop gradually as normalized variance increases. The red lines indicate coverage (cumulative ratio) of examples. We present results from CIFAR-100.

the examples belonging to the corresponding bin of the normalized variance. We present results from CIFAR-100 with ResNet-34 and VGGNet with 16 layers. The histograms illustrate the strong correlation between the predicted variance and the reliability—accuracy and confidence—of a prediction; we can estimate accuracy and uncertainty of an example effectively based on its prediction variances given by multiple stochastic inferences.

4.2. Variance-Weighted Confidence-Integrated Loss

The strong correlation of accuracy and confidence with the predicted variance observed in Figure 2 shows great potential to make confidence-calibrated predictions through stochastic inferences. However, variance computation involves multiple stochastic inferences by executing multiple forward passes. Note that this property incurs additional computational cost and may produce inconsistent results.

To alleviate these limitations, we propose a generic framework for training accuracy-score calibrated networks whose prediction score from a single forward pass directly provides the confidence of a prediction. This objective achieved by designing a new loss function, which augments a confidence-calibration term to the standard cross-entropy loss, while the two terms are balanced by the variance measured by multiple stochastic inferences. Specifically, our variance-weighted confidence-integrated loss $\mathcal{L}_{\text{VWCI}}(\cdot)$ for the whole training data $(x_i, y_i) \in \mathcal{D}$ is defined by a linear combination of the standard cross-entropy loss with the ground-truth $\mathcal{L}_{\text{GT}}(\cdot)$ and the cross-entropy with a uniform distribution $\mathcal{L}_{\text{U}}(\cdot)$, which is formally given by

$$\begin{aligned}\mathcal{L}_{\text{VWCI}}(\theta) &= \sum_{i=1}^{N}(1-\alpha_i)\mathcal{L}_{\text{GT}}^{(i)}(\theta) + \alpha_i\mathcal{L}_{\text{U}}^{(i)}(\theta) \\ &= \frac{1}{T}\sum_{i=1}^{N}\sum_{j=1}^{T} -(1-\alpha_i)\log p(y_i|x_i,\hat{\omega}_{i,j}) \\ &\quad + \alpha_i D_{\text{KL}}(\mathcal{U}(y)||p(y|x_i,\hat{\omega}_{i,j})) + \xi_i \qquad (10)\end{aligned}$$

where $\alpha_i \in [0,1]$ is a normalized variance, $\hat{\omega}_{i,j}(=\theta \odot \epsilon_{i,j})$ is a sampled model parameter with binary noise for stochastic prediction, T is the number of stochastic inferences, and ξ_i is a constant.

The two terms in our variance-weighted confidence-integrated loss pushes the network toward the opposite directions; the first term encourages the network to fit the ground-truth label while the second term forces the network to make a prediction close to the uniform distribution. These terms are linearly interpolated by an instance-specific balancing coefficient α_i, which is given by normalizing the prediction variance of an example obtained from multiple stochastic inferences. Note that the normalized variance α_i is distinct for each training example and is used to measure model uncertainty. Therefore, the optimization of our loss function produces gradient signals, which lead the predictions toward a uniform distribution for the examples with high uncertainty derived by high variances while increasing the prediction scores of the examples with low variances.

By training deep neural networks using the proposed loss function, we estimate the uncertainty of each testing example with a single forward pass. Unlike the ordinary models, a prediction score of our model is well-calibrated and represents confidence of a prediction, which means that we can rely more on the predictions with higher scores.

4.3. Confidence-Integrated Loss

Our claim is that an adaptive combination of the cross-entropy losses with respect to the ground-truth and a uniform distribution is a reasonable choice to learn uncertainty. As a special case of the proposed loss, we also present a blind version of the combination, which can be used as a baseline uncertainty estimation technique. This baseline loss function is referred to as the confidence-integrated loss, which is given by

$$\begin{aligned}\mathcal{L}_{\text{CI}}(\theta) &= \mathcal{L}_{\text{GT}}(\theta) + \beta\mathcal{L}_{\text{U}}(\theta) \\ &= \sum_{i=1}^{N} -\log p(y_i|x_i,\theta) \\ &\quad + \beta D_{\text{KL}}(\mathcal{U}(y)||p(y|x_i,\theta)) + \xi, \qquad (11)\end{aligned}$$

where $p(y|x_i,\theta)$ is the predicted distribution with model parameter θ and ξ is a constant. The main idea of this loss function is to regularize with a uniform distribution by expecting the score distributions of uncertain examples to be flattened first while the distributions of confident ones remain intact, where the impact of the confidence-integrated loss term is controlled by a global hyper-parameter β.

The proposed loss function is also employed in [21] to regularize deep neural networks and improve classification accuracy. However, [21] does not discuss confidence calibration issues while presenting marginal accuracy improvement. On the other hand, [13] discusses a similar loss function but focuses on differentiating between in-distribution and out-of-distribution examples by measuring the loss of each example using only one of the two loss terms depending on its origin.

Contrary to the existing approaches, we employ the loss function in Eq. (11) to estimate prediction confidence in deep neural networks. Although the confidence-integrated loss makes sense intuitively, such blind selection of a hyper-parameter β limits its generality compared to our variance-weighted confidence-integrated loss.

4.4. Relation to Other Calibration Approaches

There are several score calibration techniques [5, 16, 18, 29] by adjusting confidence scores through post-processing, among which [5] presents a method to calibrate confidence of predictions by scaling logits of a network using a global temperature τ. The scaling is performed before applying the softmax function, and τ is trained with a validation dataset. As discussed in [5], this simple technique is equivalent to maximize entropy of the output distribution $p(y_i|x_i)$. It is also identical to minimize KL-divergence $D_{\text{KL}}(p(y_i|x_i)||\,\mathcal{U}(y))$ because

$$\begin{aligned}&D_{\text{KL}}(p(y_i|x_i)||\,\mathcal{U}(y)) \\ &\quad = \sum_{c\in C} p(y_i^c|x_i)\log p(y_i^c|x_i) - p(y_i^c|x_i)\log\mathcal{U}(y^c) \\ &\quad = -H(p(y_i|x_i)) + \xi', \qquad (12)\end{aligned}$$

where C is a class set and ξ' is a constant. We can formulate another confidence-integrated loss with the entropy as

$$\mathcal{L}'_{\text{CI}}(\theta) = \sum_{i=1}^{N} -\log p(y_i|x_i,\theta) - \gamma H(p(y_i|x_i,\theta)), \quad (13)$$

where γ is a constant. Eq. (13) implies that temperature scaling in [5] is closely related to our framework.

5. Experiments

5.1. Experimental Settings

We select four most widely used deep neural network architectures to test the proposed algorithm: ResNet [7], VGGNet [23], WideResNet [30] and DenseNet [9].

We employ stochastic depth in ResNet as proposed in [7] while employing dropouts [24] before every fc layer except for the classification layer in other architectures. Note that, as discussed in Section 3.3, both stochastic depth and dropout inject multiplicative binary noise to within-layer activations or residual blocks, they are equivalent to noise injection into network weights. Hence, training with ℓ_2 regularization term enables us to interpret stochastic depth and dropout by Bayesian models.

We evaluate the proposed framework on two benchmarks, Tiny ImageNet and CIFAR-100, which contain 64×64 images in 200 object classes and 32×32 images in 100 object classes, respectively. There are 500 training images per class in both datasets. For testing, we use the validation set of Tiny ImageNet and the test set of CIFAR-100, which have 50 and 100 images per class, respectively. To test the two benchmarks with the same architecture, we resize images in Tiny ImageNet to 32×32.

All networks are trained by a stochastic gradient decent method with the momentum 0.9 for 300 epochs. We set the initial learning rate to 0.1 with the exponential decay in a factor of 0.2 at epoch 60, 120, 160, 200 and 250. Each batch consists of 64 training examples for ResNet, WideResNet and DenseNet and 256 for VGGNet. To train networks with

Table 1. Classification accuracy and calibration scores for several combinations of network architectures and datasets. We compare models trained with baseline, CI and VWCI losses. Since CI loss involves a hyper-parameter β, we present mean and standard deviation of results from models with five different β's. In addition, we also show results from the oracle CI loss, CI[Oracle], which are the most optimistic values out of results from all β's in individual columns. Note that the numbers corresponding to CI[Oracle] may come from different β's. Refer to the supplementary document for the full results.

Dataset	Architecture	Method	Accuracy [%]	ECE	MCE	NLL	Brier Score
Tiny ImageNet	ResNet-34	Baseline	50.82	0.067	0.147	2.050	0.628
		CI	50.09 ± 1.08	0.134 ± 0.079	0.257 ± 0.098	2.270 ± 0.212	0.665 ± 0.037
		VWCI	**52.80**	**0.027**	**0.076**	**1.949**	**0.605**
		CI[Oracle]	51.45	0.035	0.171	2.030	0.620
	VGG-16	Baseline	46.58	0.346	0.595	4.220	0.844
		CI	46.82 ± 0.81	0.226 ± 0.095	0.435 ± 0.107	3.224 ± 0.468	0.761 ± 0.054
		VWCI	**48.03**	**0.053**	**0.142**	**2.373**	**0.659**
		CI[Oracle]	47.39	0.122	0.320	2.812	0.701
	WideResNet-16-8	Baseline	55.92	0.132	0.237	1.974	0.593
		CI	55.80 ± 0.44	0.115 ± 0.040	0.288 ± 0.100	1.980 ± 0.114	0.594 ± 0.017
		VWCI	**56.66**	**0.046**	**0.136**	**1.866**	**0.569**
		CI[Oracle]	56.38	0.050	0.208	1.851	0.572
	DenseNet-40-12	Baseline	42.50	**0.020**	0.154	2.423	0.716
		CI	40.18 ± 1.68	0.059 ± 0.061	0.152 ± 0.082	2.606 ± 0.208	0.748 ± 0.035
		VWCI	**43.25**	0.025	**0.089**	**2.410**	**0.712**
		CI[Oracle]	41.21	0.025	0.094	2.489	0.726
CIFAR-100	ResNet-34	Baseline	77.19	0.109	0.304	1.020	0.345
		CI	77.56 ± 0.60	0.134 ± 0.131	0.251 ± 0.128	1.064 ± 0.217	0.360 ± 0.057
		VWCI	**78.64**	**0.034**	**0.089**	**0.908**	**0.310**
		CI[Oracle]	78.54	0.029	0.087	0.921	0.321
	VGG-16	Baseline	73.78	0.187	0.486	1.667	0.437
		CI	73.75 ± 0.35	0.183 ± 0.079	0.489 ± 0.214	1.526 ± 0.175	0.436 ± 0.034
		VWCI	**73.87**	**0.098**	**0.309**	**1.277**	**0.391**
		CI[Oracle]	73.78	0.083	0.285	1.289	0.396
	WideResNet-16-8	Baseline	77.52	0.103	0.278	0.984	0.336
		CI	77.35 ± 0.21	0.133 ± 0.091	0.297 ± 0.108	1.062 ± 0.180	0.356 ± 0.044
		VWCI	**77.74**	**0.038**	**0.101**	**0.891**	**0.314**
		CI[Oracle]	77.53	0.074	0.211	0.931	0.327
	DenseNet-40-12	Baseline	65.91	0.074	0.134	1.238	0.463
		CI	64.72 ± 1.46	0.070 ± 0.040	0.138 ± 0.055	1.312 ± 0.125	0.482 ± 0.028
		VWCI	**67.45**	**0.026**	**0.094**	**1.161**	**0.439**
		CI[Oracle]	66.20	0.019	0.053	1.206	0.456

the proposed variance-weighted confidence-integrated loss, we draw T samples with network parameters ω_i for each input image, and compute the normalized variance α based on T forward passes. The normalized variance is given by the mean of the Bhattacharyya coefficients between individual predictions and the average prediction, and, consequently, in the range of $[0.1]$.

5.2. Evaluation Metric

We measure classification accuracy and calibration scores—expected calibration error (ECE), maximum calibration error (MCE), negative log likelihood (NLL) and Brier score—of the trained models.

Let B_m be a set of indices of test examples whose prediction scores for the ground-truth labels fall into interval $(\frac{m-1}{M}, \frac{m}{M}]$, where $M(=20)$ is the number of bins. ECE and MCE are formally defined by

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N'} \left|\text{acc}(B_m) - \text{conf}(B_m)\right|,$$

$$\text{MCE} = \max_{m \in \{1,\dots,M\}} \left|\text{acc}(B_m) - \text{conf}(B_m)\right|,$$

where N' is the number of the test samples. Also, accuracy and confidence of each bin are given by

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{1}(\hat{y}_i = y_i),$$

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} p_i,$$

where $\mathbb{1}$ is an indicator function, $\hat{y}_i$ and y_i are predicted and true label of the i^{th} example and p_i is its predicted confi-

Table 2. Comparison between VWCI and TS of multiple datasets and architectures.

Dataset	Architecture	Method	Accuracy [%]	ECE	MCE	NLL	Brier Score
Tiny ImageNet	ResNet-34	TS (case 1)	50.82	0.162	0.272	2.241	0.660
		TS (case 2)	47.20	**0.021**	0.080	2.159	0.661
		VWCI	**52.80**	0.027	**0.076**	**1.949**	**0.605**
	VGG-16	TS (case 1)	46.58	0.358	0.604	4.425	0.855
		TS (case 2)	46.53	**0.028**	**0.067**	**2.361**	0.671
		VWCI	**48.03**	0.053	0.142	2.373	**0.659**
	WideResNet-16-8	TS (case 1)	55.92	0.200	0.335	2.259	0.627
		TS (case 2)	53.95	**0.027**	0.224	1.925	0.595
		VWCI	**56.66**	0.046	**0.136**	**1.866**	**0.569**
	DenseNet-40-12	TS (case 1)	42.50	0.037	0.456	2.436	0.717
		TS (case 2)	41.63	**0.024**	0.109	2.483	0.728
		VWCI	**43.25**	0.025	**0.089**	**2.410**	**0.712**
CIFAR-100	ResNet-34	TS (case 1)	77.67	0.133	0.356	1.162	0.354
		TS (case 2)	77.40	0.036	0.165	**0.886**	0.323
		VWCI	**78.64**	**0.034**	**0.089**	0.908	**0.310**
	VGG-16	TS (case 1)	73.66	0.197	0.499	1.770	0.445
		TS (case 2)	72.69	**0.031**	**0.074**	**1.193**	**0.389**
		VWCI	**73.87**	0.098	0.309	1.277	0.391
	WideResNet-16-8	TS (case 1)	77.52	0.144	0.400	1.285	0.361
		TS (case 2)	76.42	**0.028**	**0.101**	**0.891**	0.332
		VWCI	**77.74**	0.038	**0.101**	**0.891**	**0.314**
	DenseNet-40-12	TS (case 1)	65.91	0.095	0.165	1.274	0.468
		TS (case 2)	64.96	0.082	0.163	1.306	0.481
		VWCI	**67.45**	**0.026**	**0.094**	**1.161**	**0.439**

dence. NLL and Brier score are another ways to measure the calibration [2, 5, 6], which are defined as

$$\mathrm{NLL} = -\sum_{i=1}^{N'} \log p(y_i|x_i,\theta),$$

$$\mathrm{Brier} = \sum_{i=1}^{N'} \sum_{j=1}^{C} (p(\hat{y}_i = j|x_i,\theta) - \mathbb{1}(y_i = j))^2.$$

We note that low values for all these calibration scores means that the network is well-calibrated.

5.3. Results

Table 1 presents accuracy and calibration scores for several combinations of network architectures and benchmark datasets. The models trained with VWCI loss consistently outperform the models with CI loss, which is a special case of VWCI, and the baseline on both classification accuracy and confidence calibration performance. We believe that the accuracy gain is partly by virtue of the stochastic regularization with multiple samples [19]. Performance of CI is given by the average and variance from 5 different cases of $\beta (= 1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4})$[1] and CI[Oracle] denotes the most optimistic value among the 5 cases in each column. Note that VWCI presents outstanding results in most cases even when compared with CI[Oracle] and that performance of CI is sensitive to the choice of β's. These results imply that the proposed loss function balances two conflicting loss terms effectively using the variance of multiple stochastic inferences while performance of CI varies depending on hyper-parameter setting in each dataset.

We also compare the proposed framework with the state-of-the-art post-processing method, temperature scaling (TS) [5]. The main distinction between post-processing methods and our work is the need for held-out dataset; our method allows to calibrate scores during training without additional data while [5] requires held-out validation sets to calibrate scores. To illustrate the effectiveness of our framework, we compare our approach with TS in the following two scenarios: 1) using the entire training set for both training and calibration and 2) using 90% of training set for training and the remaining 10% for calibration. Table 2 presents that case 1 suffers from poor calibration performance and case 2 loses accuracy substantially due to training data reduction although it shows comparable calibration scores to VWCI. Note that TS may also suffer from the binning artifacts of histograms although we do not investigate this limitation in our work.

5.4. Discussion

To show the effectiveness of the propose framework, we analyze the proposed algorithm with ablative experiments.

[1] These 5 values of β are selected favorably to CI based on our preliminary experiment.

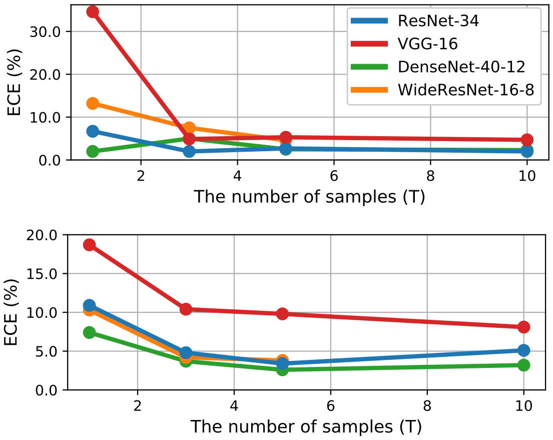


Figure 3. ECE of VWCI loss with respect to the number of samples (T) on Tiny ImageNet (top) and CIFAR-100 (bottom) dataset.

Table 3. Comparisons between the models based on the VWCI losses, trained from scratch and the uncalibrated pretrained networks (denoted by VWCI*).

	Architecture	Method	Acc. [%]	ECE	MCE	NLL	Brier
Tiny ImageNet	ResNet-34	Baseline	77.19	0.109	0.304	1.020	0.345
		VWCI	**78.64**	0.034	0.089	**0.908**	**0.310**
		VWCI*	77.87	**0.026**	**0.069**	1.013	0.346
	VGG-16	Baseline	73.78	0.187	0.486	1.667	0.437
		VWCI	73.87	0.098	0.309	1.277	0.391
		VWCI*	**74.17**	**0.074**	**0.243**	**1.227**	**0.385**
CIFAR-100	ResNet-34	Baseline	50.82	0.067	0.147	2.050	0.628
		VWCI	**52.80**	**0.027**	**0.076**	**1.949**	**0.605**
		VWCI*	52.77	0.034	0.099	1.965	**0.605**
	VGG-16	Baseline	46.58	0.346	0.595	4.220	0.844
		VWCI	**48.03**	**0.053**	**0.142**	**2.373**	**0.659**
		VWCI*	46.98	0.056	0.162	2.446	0.683

Effect of sample size for stochastic inferences Figure 3 illustrates ECE of the models trained with our VWCI loss by varying the number of stochastic inferences (T) during training. The increase of T is helpful to improve accuracy and calibration quality at the beginning but its benefit is saturated when T is between 5 and 10 in general. Such tendency is consistent in all the tested architectures, datasets, and evaluation metrics including ECE. We set T to 5 in all the experiments.

Training cost Although our approach allows single-shot confidence calibration at test-time, it increases time complexity for training due to multiple stochastic inferences. Fortunately, the calibrated models can be trained more efficiently without stochastic inferences in the majority ($\geq 80\%$) of iterations by initializing the networks with the pretrained baseline models. Table 3 confirms that the performance of our models trained from the uncalibrated pretrained models is as competitive as (or often even better than) the ones trained from scratch with the VWCI losses.

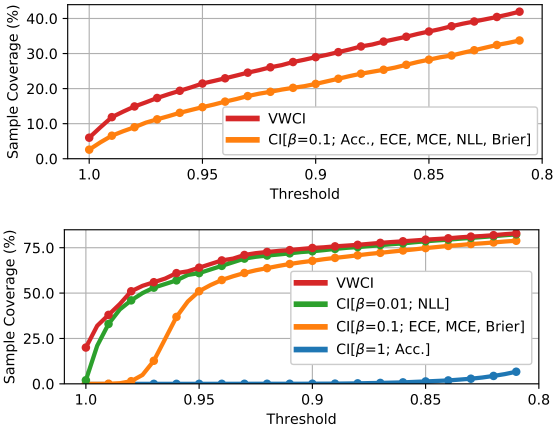


Figure 4. Coverage of ResNet-34 models with respect to confidence interval on Tiny ImageNet (top) and CIFAR-100 (bottom). The coverage is computed by the portion of examples with higher accuracy and confidence than the thresholds shown in x-axis. We present results from multiple CI models with best performances with respect to individual metrics, which are shown in the legends.

Reliability Our approach effectively maintains examples with high accuracy and confidence, which is a desirable property for building reliable real-world systems. Figure 4 illustrates portion of test examples with higher accuracy and confidence than the various thresholds in ResNet-34, where VWCI presents better coverage of the examples than CI[Oracle]. Note that coverage of CI often depends on the choice of β significantly as demonstrated in Figure 4 (right) while VWCI maintains higher coverage than CI using accurately calibrated prediction scores. These results imply that using the predictive uncertainty for balancing the loss terms is preferable to setting with a constant coefficient.

6. Conclusion

We presented a generic framework for uncertainty estimation of a prediction in deep neural networks by calibrating accuracy and score based on stochastic inferences. Based on Bayesian interpretation of stochastic regularization and our empirical observation results, we claim that variation of multiple stochastic inferences for a single example is a crucial factor to estimate uncertainty of the average prediction. Inspired by this fact, we design the variance-weighted confidence-integrated loss to learn confidence-calibrated networks and enable uncertainty to be estimated by a single prediction. The proposed algorithm is also useful to understand existing confidence calibration methods in a unified way, and we compared our algorithm with other variations within our framework to analyze their properties.

Acknowledgments This work was partly supported by Samsung Advanced Institute of Technology and Korean ICT R&D program of the MSIP/IITP grant [2014-0-00059, 2017-0-01778].

References

[1] David Barber and Christopher M Bishop. Ensemble Learning for Multi-layer Networks. In *NIPS*, 1998. 2

[2] Glenn W Brier and Roger A Allen. Verification of weather forecasts. In *Compendium of meteorology*, pages 841–848. Springer, 1951. 7

[3] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *ICML*, 2016. 2, 3

[4] Alex Graves. Practical variational inference for neural networks. In *NIPS*, 2011. 2

[5] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. In *ICML*, 2017. 2, 5, 7

[6] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York Inc., 2001. 7

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 5

[8] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic Variational Inference. *J. Mach. Learn. Res.*, 14(1):1303–1347, May 2013. 2

[9] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. In *CVPR*, 2017. 5

[10] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. In *ECCV*. 1, 2, 3

[11] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 2015. 1

[12] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *NIPS*, 2017. 2

[13] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples. In *ICLR*, 2018. 2, 5

[14] David J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Comput.*, 4(3):448–472, May 1992. 2

[15] Patrick McClure and Nikolaus Kriegeskorte. Representation of uncertainty in deep neural networks through sampling. *CoRR*, abs/1611.01639, 2016. 2

[16] Mahdi Pakdaman Naeini, Gregory F Cooper, and Milos Hauskrecht. Obtaining Well Calibrated Probabilities Using Bayesian Binning. In *AAAI*, 2015. 5

[17] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, 1996. 2

[18] Alexandru Niculescu-Mizil and Rich Caruana. Predicting Good Probabilities with Supervised Learning. In *ICML*, 2005. 2, 5

[19] Hyeonwoo Noh, Tackgeun You, Jonghwan Mun, and Bohyung Han. Regularizing Deep Neural Networks by Noise: Its Interpretation and Optimization. In *NIPS*, 2017. 7

[20] Nick Pawlowski, Andrew Brock, Matthew CH Lee, Martin Rajchl, and Ben Glocker. Implicit Weight Uncertainty in Neural Networks. *arXiv preprint arXiv:1711.01297*, 2017. 2

[21] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing Neural Networks by Penalizing Confident Output Distributions. *arXiv preprint arXiv:1701.06548*, 2017. 2, 5

[22] John Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Advanced in Large Margin Classifiers*, 10, 06 2000. 2

[23] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. In *ICLR*, 2015. 5

[24] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 1, 2, 5

[25] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, 2016. 2

[26] Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian Uncertainty Estimation for Batch Normalized Deep Networks. *arXiv preprint arXiv:1802.06455*, 2018. 3

[27] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of Neural Networks using Dropconnect. In *ICML*, 2013. 2

[28] Bianca Zadrozny and Charles Elkan. Obtaining Calibrated Probability Estimates from Decision Trees and Naive Bayesian Classifiers. In *ICML*, 2001. 2

[29] Bianca Zadrozny and Charles Elkan. Transforming Classifier Scores into Accurate Multiclass Probability Estimates. In *KDD*, 2002. 5

[30] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *BMVC*, 2016. 5