PITFALLS OF IN-DOMAIN UNCERTAINTY ESTIMATION AND ENSEMBLING IN DEEP LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Uncertainty estimation and ensembling methods go hand-in-hand. Uncertainty estimation is one of the main benchmarks for assessment of ensembling performance. At the same time, deep learning ensembles have provided state-of-the-art results in uncertainty estimation. In this work, we focus on in-domain uncertainty for image classification. We explore the standards for its quantification and point out pitfalls of existing metrics. Avoiding these pitfalls, we perform a broad study of different ensembling techniques. To provide more insight in the broad comparison, we introduce the *deep ensemble equivalent* (DEE) and show that many sophisticated ensembling techniques are equivalent to an ensemble of very few independently trained networks in terms of the test log-likelihood.

1 INTRODUCTION

Deep neural networks (DNNs) have become one of the most popular families of machine learning models. The predictive performance of DNNs for classification is often measured in terms of accuracy. However, DNNs have been shown to yield inaccurate and unreliable *probability* estimates, or predictive uncertainty (Guo et al., 2017). This has brought considerable attention to the problem of uncertainty estimation with deep neural networks.

There are many faces to uncertainty estimation. Different desirable uncertainty estimation properties of a model require different settings and metrics to capture them. *Out-of-domain* uncertainty of the model is measured on data that does not follow the same distribution as the training dataset (out-of-domain data). Out-of-domain data can include images corrupted with rotations or blurring, adversarial attacks (Szegedy et al., 2013) or data points from a completely different dataset. The model is expected to be resistant to data corruptions and to be more uncertain on out-of-domain data than on in-domain data. This setting was explored in a recent study by (Ovadia et al., 2019). On the contrary, *in-domain* uncertainty of the model is measured on data taken from the training data distribution, i.e. data from the same domain. In this case, a model is expected to provide correct probability estimates: it should not be overconfident in the wrong predictions, and should not be too uncertain about the correct predictions.

Ensembles of deep neural networks have become a de-facto standard for uncertainty estimation and improving the quality of deep learning models (Hansen & Salamon, 1990; Krizhevsky et al., 2009; Lakshminarayanan et al., 2017). There are two main directions in the field of training ensembles of DNNs: training stochastic computation graphs and obtaining separate snapshots of neural network weights. Methods based on the paradigm of stochastic computation graphs introduce noise over weights or activations of deep learning models. When the model is trained, each sample of the noise corresponds to a member of the ensemble. During test time, the predictions are averaged across the noise samples. These methods include (test-time) data augmentation, dropout (Srivastava et al., 2014; Gal & Ghahramani, 2016), variational inference (Blundell et al., 2015; Kingma et al., 2015; Louizos & Welling, 2017), batch normalization (Ioffe & Szegedy, 2015; Teye et al., 2018; Atanov et al., 2019), Laplace approximation (Ritter et al., 2018) and many more. Snapshot-based methods aim to obtain sets of weights for deep learning models and then to average the predictions across these weights. The weights can be trained independently (e.g., deep ensembles (Lakshminarayanan et al., 2017)), collected on different stages of a training trajectory (e.g., snapshot ensembles (Huang et al., 2017) and fast geometric ensembles (Garipov et al., 2018)), or obtained from a sampling process (e.g., MCMC-based methods (Welling & Teh, 2011; Zhang et al., 2019)). These two paradigms

can be combined. Some works suggest construction of ensembles of stochastic computation graphs (Tomczak et al., 2018), while others make use of the collected snapshots to construct a stochastic computation graph (Wang et al., 2018; Maddox et al., 2019).

In this paper, we focus on assessing the quality of in-domain uncertainty estimation. We show that many common metrics in the field are either not comparable across different models or fail to provide a reliable ranking, and then address some of stated pitfalls. Following that, we perform a broad evaluation of modern DNN ensembles on CIFAR-10/100 and ImageNet datasets. To aid interpretatability, we introduce the *deep ensemble equivalent* score that essentially measures the number of "independent" models in an ensemble of DNNs. We draw a set of conclusions with regard to ensembling performance and metric reliability to guide future research practices. For example, we find that methods specifically designed to traverse different "optima" of the loss function (snapshot ensembles and cyclical SGLD) come close to matching the performance of deep ensembles while methods that only explore the vicinity of a single "optimum" (Dropout, FGE, K-FAC Laplace and variational inference) fall far behind.

2 Scope of the paper and related work

We use standard benchmark problems of image classification as it is a common setting in papers on learning ensembles of neural networks. There are other practically relevant settings where the correctness of probabilistic estimates can be a priority. These settings include, but are not limited to, regression, image segmentation, language modelling (Gal, 2016), active learning (Settles, 2012) and reinforcement learning (Buckman et al., 2018; Chua et al., 2018).

We focus on in-domain uncertainty, as opposed to out-of-domain uncertainty. Out-of-domain uncertainty includes detection of inputs that come from a completely different domain or have been corrupted by noise or adversarial attacks. This setting has been thoroughly explored by (Ovadia et al., 2019).

We only consider methods that are trained on clean data with simple data augmentation. Some other methods use out-of-domain data (Malinin & Gales, 2018) or more elaborate data augmentation e.g., mixup (Zhang et al., 2017), adversarial training (Lakshminarayanan et al., 2017) to improve accuracy, robustness and uncertainty.

We use conventional training procedures. We use the stochastic gradient descent (SGD) and use batch normalization (Ioffe & Szegedy, 2015), both being the de-facto standards in modern deep learning. We refrain from using more elaborate optimization techniques including works on super-convergence (Smith & Topin, 2019) and stochastic weight averaging (Izmailov et al., 2018). These techniques can be used to drastically accelerate training and improve the predictive performance. Because of that, we do not not comment on the training time of different ensembling methods since the use of more efficient training techniques would render such a comparison obsolete.

A number of related works study ways of approximating and accelerating prediction in ensembles. The distillation mechanism allows to approximate the prediction of an ensemble by a single neural network (Hinton et al., 2015; Balan et al., 2015), whereas fast dropout (Wang & Manning, 2013) and deterministic variational inference (Wu et al., 2018) allow to approximate the predictive distribution of specific stochastic computation graphs. We measure the raw power of ensembling techniques without these approximations.

All of the aforementioned alternative settings are orthogonal to the scope of this paper and are promising points of interest for further research.

3 Pitfalls of in-domain uncertainty estimation

No single metric measures all desirable properties of uncertainty estimates obtained with a model. Because of this, the community has used different metrics that aim to measure the quality of uncertainty estimation, e.g. the Brier score (Brier, 1950), log-likelihood (Quinonero-Candela et al., 2005), different calibration metrics (Guo et al., 2017; Nixon et al., 2019), performance of misclassification detection (Malinin & Gales, 2018), and threshold–accuracy curves (Lakshminarayanan et al., 2017).

We consider a classification problem with a dataset that consists of N training and n testing pairs $(x_i, y_i^*) \sim p(x, y)$, where x_i is an object and $y_i^* \in \{1, \dots, C\}$ is a discrete class label. A probabilistic classifier maps an object x_i into a predictive distribution $\hat{p}(y \mid x_i)$. The predictive distribution $\hat{p}(y \mid x_i)$ of deep neural networks is usually defined as a softmax function $\hat{p}(y \mid x) = \mathrm{Softmax}(z(x)/T)$, where $z(x)$ is a vector of logits and T is a scalar parameter standing for the temperature of the predictive distribution. The maximum probability $\max_c \hat{p}(y = c \mid x_i)$ is called a confidence of a classifier $\hat{p}$ on object x_i. The indicator function is denoted by $\mathbb{I}[\cdot]$ throughout the text.

3.1 Log-likelihood and Brier score

The average test log-likelihood $LL = \frac{1}{n} \sum_{i=1}^{n} \log \hat{p}(y = y_i^* \mid x_i)$ is a popular metric for measuring the quality of in-domain uncertainty of deep learning models. It directly penalizes high probability scores assigned to incorrect labels and low probability scores assigned to the correct labels y_i^*.

LL is sensitive to the temperature T. The temperature that has been learned during training can be far from optimal for the test data. However, a nearly optimal temperature can be found post-hoc by maximizing the log-likelihood on validation data. This approach is called temperature scaling or calibration (Guo et al., 2017). Despite its simplicity, temperature scaling results in a marked improvement in the LL.

While ensembling techniques tend to have better temperature than single models, the default choice of $T = 1$ is still sub-optimal. Comparing the LL with sub-optimal temperatures—that is often the case—can produce an arbitrary ranking of different methods.

Comparison of the log-likelihood should only be performed at the optimal temperature.

Empirically, we demonstrate that the overall ordering of methods and also the best ensembling method according to the LL can vary depending on temperature T. While this applies to most ensembling techniques (see Appendix C), this effect is most noticeable on experiments with data augmentation on ImageNet (Figure 1). We will call the log-likelihood at the optimal temperature *the calibrated log-likelihood*. We show how to obtain an unbiased estimate of the calibrated log-likelihood without a held-out validation set in Section 3.5.

LL also demonstrates a high correlation with accuracy ($\rho > 0.86$), that in case of calibrated LL becomes even stronger ($\rho > 0.95$). That suggest that while (calibrated) LL measures the uncertainty of the model, it still significantly depends on the accuracy and vice versa. A model with higher accuracy would likely have a higher log-likelihood even if the quality of its uncertainty is lower in some respects. See Appendix C for more details.

Brier score Brier score $BS = \frac{1}{n} \frac{1}{C} \sum_{i=1}^{n} \sum_{c=1}^{C} (\mathbb{I}[y_i^* = c] - \hat{p}(y = c \mid x_i))^2$ has been known for a long time as a metric for verification of predicted probabilities (Brier, 1950). Similarly to the log-likelihood, the Brier score penalizes low probabilities assigned to correct predictions and high probabilities assigned to wrong ones. It is also sensitive to the temperature of the softmax distribution and behaves similarly to the log-likelihood. While these metrics are not strictly equivalent, they show a high empirical correlation for a wide range of models on CIFAR-10, CIFAR-100 and ImageNet datasets (see Appendix A).

3.2 Misclassification detection

Detection of wrong predictions of the model, or misclassifications, is a popular downstream problem aiding in assessing the quality of in-domain uncertainty. Since misclassification detection is essentially a binary classification problem, some papers measure its quality using conventional metrics for binary classification such as AUC-ROC and AUC-PR (Malinin & Gales, 2018; Cui et al., 2019; Możejko et al., 2018). These papers use an uncertainty criterion like confidence or predictive entropy $\mathcal{H}[\hat{p}(y \mid x_i)]$ as a prediction score.

While these metrics can be used to assess the misclassification detection performance of a single model, they cannot be used to directly compare misclassification performance across different models. Correct and incorrect predictions are specific for every model, therefore, every model induces

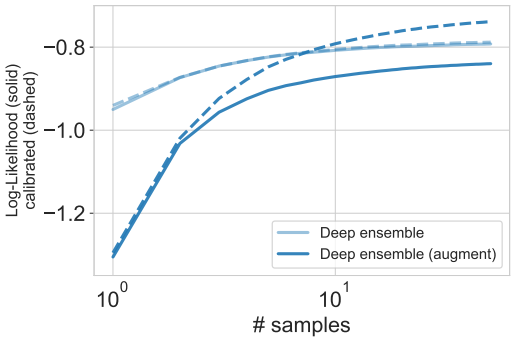


Figure 1: The average log-likelihood of different ensembles before (solid lines) and after (dashed lines) temperature scaling (TS) on ResNet50 on ImageNet dataset. Without TS test-time data augmentation causes worse log-likelihood of plain deep ensembles, while TS makes deep ensembles with test-time augmentation superior.

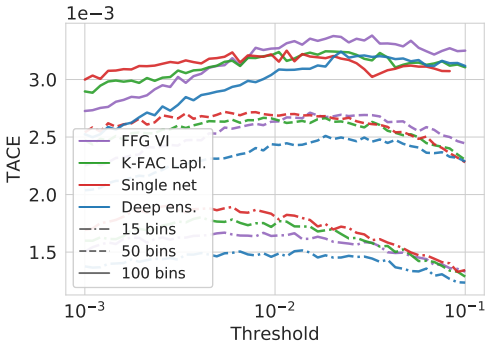


Figure 2: Thresholded adaptive calibration error (TACE) is highly sensitive to the threshold and the number of bins and does not provide a stable ranking for different methods. TACE is reported for VGG16BN model on CIFAR-100 dataset and is evaluated at the optimal temperature.

its own binary classification problem. The induced problems can differ significantly from each other since different models produce different confidences and misclassify different objects.

AUCs for misclassification detection can not be directly compared between different models.

While comparing AUCs is incorrect in this setting, it is correct to compare these metrics in many out-of-domain data detection problems. In that case, both objects and targets of induced binary classification problems remain fixed for all models. Note however that this condition still usually breaks down in the problem of detection of adversarial attacks since different models generally have different inputs after an adversarial attack.

3.3 Classification with rejection

Accuracy-confidence curves are another way to measure the performance of misclassification detection. These curves measure the accuracy on the set of objects with confidence $\max_c \hat{p}(y = c \,|\, x_i)$ above a certain threshold τ (Lakshminarayanan et al., 2017) and ignoring or *rejecting* the others.

The main problem with accuracy-confidence curves is that they rely too much on calibration and the actual values of confidence. Models with different temperatures have different numbers of objects at each confidence level which does not allow for a meaningful comparison. To overcome this problem, one can switch from thresholding by the confidence level to thresholding by the number of rejected objects. The corresponding curves are then less sensitive to temperature scaling and allow to compare the rejection ability in a more meaningful way. Such curves have been known as *accuracy-rejection curves* (Nadeem et al., 2009). In order to obtain a scalar metric for easy comparisons, one can compute the area under this curve, resulting in AU-ARC (Nadeem et al., 2009).

3.4 Calibration metrics

Informally speaking, a probabilistic classifier is calibrated if any predicted class probability is equal to the true class probability according to the underlying data distribution (see (Vaicenavicius et al., 2019) for formal definitions). Any deviation from perfect calibration is called miscalibration. For brevity, we will use $\hat{p}_{i,c}$ to denote $\hat{p}(y = c \,|\, x_i)$ in the current section.

Expected Calibration Error (ECE) (Naeini et al., 2015) is a metric that estimates model miscalibration by binning the assigned probability scores and comparing them to average accuracies inside these bins. Assuming B_m denotes the m-th bin and M is overall number of bins, the ECE is defined

as follows:

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\mathrm{acc}(B_m) - \mathrm{conf}(B_m)\right|, \tag{1}$$

where $\mathrm{acc}(B) = |B|^{-1} \sum_{i \in B} \mathbb{I}[\arg\max_c \hat{p}_{i,c} = y_i^*]$ and $\mathrm{conf}(B) = |B|^{-1} \sum_{i \in B} \hat{p}_{i,y_i^*}$.

A recent line of works on measuring calibration in deep learning (Nixon et al., 2019; Vaicenavicius et al., 2019) outline several problems of the ECE score. Firstly, ECE is a biased estimate of the true calibration. Secondly, ECE-like scores cannot be optimized directly since they are minimized by a model with constant uniform predictions, making the infinite temperature $T = +\infty$ its global optimum. Thirdly, ECE only estimates miscalibration in terms of the maximum assigned probability whereas practical applications may require the full predicted probability vector to be calibrated. Finally, biases of ECE on different models may not be equal, rendering the miscalibration estimates incompatible.

Thresholded Adaptive Calibration Error (TACE) was proposed as a step towards solving some of these problems (Nixon et al., 2019). TACE disregards all predicted probabilities that are less than a certain threshold (hence *thresholded*), chooses the bin locations adaptively so that each bin has the same number of objects (hence *adaptive*), and estimates miscalibration of probabilties across all classes in the prediction (not just the top-1 predicted class as in ECE). Assuming that B_m^{TA} denotes the m-th thresholded adaptive bin and M is the overall number of bins, TACE is defined as follows:

$$\mathrm{TACE} = \frac{1}{CM} \sum_{c=1}^{C} \sum_{m=1}^{M} \frac{|B_m^{\mathrm{TA}}|}{n} \left|\mathrm{objs}(B_m^{\mathrm{TA}}, c) - \mathrm{conf}(B_m^{\mathrm{TA}}, c)\right|, \tag{2}$$

where $\mathrm{objs}(B^{\mathrm{TA}}, c) = |B^{\mathrm{TA}}|^{-1} \sum_{i \in B^{\mathrm{TA}}} \mathbb{I}[y_i^* = c]$ and $\mathrm{conf}(B^{\mathrm{TA}}, c) = |B^{\mathrm{TA}}|^{-1} \sum_{i \in B^{\mathrm{TA}}} \hat{p}_{i,c}$.

Although TACE does solve several problems of ECE and is useful for measuring calibration of a specific model, it still cannot be used as a reliable criterion for comparing different models. Theory suggests that it is still a biased estimate of true calibration with different bias for each model. In practice, TACE is sensitive to its two parameters, the number of bins and the threshold, and does not provide a consistent ranking of different models which is shown in Figure 2.

3.5 Temperature scaling and test-time cross-validation

There are two common ways to perform temperature scaling using a validation set when training on datasets that only feature public training and test sets (e.g. CIFARs). The public training set might be divided into a smaller training set and validation set, or the public test set can be split into test and validation parts (Guo et al., 2017; Nixon et al., 2019). The problem with the first method is that the resulting models cannot be directly compared with all the other models that have been trained on the full training set. The second approach, however, provides an unbiased estimate of metrics such as log-likelihood and Brier score but introduces more variance.

In order to reduce the variance of the second approach, we perform a "test-time cross-validation". We randomly divide the test set into two equal parts, then compute metrics for each half of the test set using the temperature optimized on another half. We repeat this procedure five times and average the results across different random partitions to reduce the variance of the computed metrics.

4 A study of ensembling & Deep Ensemble Equivalent

In this paper we consider the following ensembling techniques: deep ensembles (Lakshminarayanan et al., 2017), snapshot ensembles (SSE by (Huang et al., 2017)), fast geometric ensembling (FGE by (Garipov et al., 2018)), SWA-Gaussian (SWAG by (Maddox et al., 2019)), cyclical SGLD (cSGLD by (Zhang et al., 2019)), variational inference (VI by (Blundell et al., 2015)), dropout (Srivastava et al., 2014) and test-time data augmentation (Krizhevsky et al., 2009). These techniques were chosen to cover a diverse set of approaches keeping their predictive performance in mind.

All these techniques can be summarized as distributions $q_m(\omega)$ over some parameters ω of computation graphs $z_\omega(x)$, where m stands for the technique. During testing, one can average the predictions

across parameters $\omega \sim q_m(\omega)$ to approximate the predictive distribution

$$\hat{p}(y_i \,|\, x_i) \approx \int p(y_i \,|\, x_i, \omega) q_m(\omega)\, d\omega \simeq \frac{1}{K} \sum_{k=1}^{K} p(y_i \,|\, x_i, \omega_k), \quad \omega_k \sim q_m(\omega) \tag{3}$$

For example, a deep ensemble of S networks can be represented in this form as a mixture of S Dirac's deltas $q_{\text{DE}}(\omega) = \frac{1}{S} \sum_{s=1}^{S} \delta(\omega - \omega_s)$, centered at independently trained snapshots ω_s. Similarly, a Bayesian neural network with a fully-factorized Gaussian approximate posterior distribution over the weight matrices and convolutional kernels ω is represented as $q_{\text{VI}}(\omega) = \mathcal{N}(\omega \,|\, \mu, \text{diag}(\sigma^2))$, μ and σ^2 being the optimal variational means and variances respectively.

If one considers data augmentation as a part of the computational graph, it can be parameterized by the coordinates of the random crop and the flag for whether to flip the image horizontally or not. Sampling from the corresponding $q_{\text{aug}}(\omega)$ would generate different ways to augment the data. However, as data augmentation is present by default during the training of all othe mentioned ensembling techniques, it is suitable to study it in combination with these methods and not as a separate ensembling technique. We perform such an evaluation in Section 4.3.

Typically, the approximation (equation 3) requires K independent forward passes through a neural network, making the test-time budget directly comparable across all methods.

4.1 Deep Ensemble Equivalent

Most ensembling techniques under consideration are either bounded to a single mode, or provide positively correlated samples. Deep ensembles, on the other hand, is a simple technique that provides independent samples from different modes of the loss landscape, which can intuitively result in a better ensemble. Therefore deep ensembles can be considered as a strong baseline for performance of other ensembling techniques given a fixed test-time budget. Instead of comparing the values of uncertainty estimation metrics directly, we ask the following question aiming to introduce perspective and interpretability in our comparison:

What number of independently trained networks combined yields the same performance as a particular ensembling method?

Following insights from the previous sections, we use the calibrated log-likelihood (CLL) as the main measure of uncertainty estimation performance of the ensemble. We define the Deep Ensemble Equivalent (DEE) for an ensembling method m and its upper and lower bounds as follows:

$$\text{DEE}_m(k) = \min \left\{ l \in \mathbb{R}, l \geq 1 \,\middle|\, \text{CLL}_{DE}^{\text{mean}}(l) \geq \text{CLL}_m^{\text{mean}}(k) \right\}, \tag{4}$$

$$\text{DEE}_m^{\substack{\text{upper} \\ \text{lower}}}(k) = \min \left\{ l \in \mathbb{R}, l \geq 1 \,\middle|\, \text{CLL}_{DE}^{\text{mean}}(l) \mp \text{CLL}_{DE}^{\text{std}}(l) \geq \text{CLL}_m^{\text{mean}}(k) \right\}, \tag{5}$$

where $\text{CLL}_m^{\text{mean}}(l)$ and $\text{CLL}_m^{\text{std}}(l)$ are the mean and the standard deviation of the calibrated log-likelihood achieved by an ensembling method m with l samples. We compute $\text{CLL}_{\text{DE}}^{\text{mean}}(l)$ and $\text{CLL}_{\text{DE}}^{\text{std}}(l)$ for natural numbers $l \in \mathbb{N}_{>0}$ and use linear interpolation to define them for real values $l \geq 1$. In the following plots we report $\text{DEE}_m(k)$ for different methods m with different numbers of samples k, and shade the area between the respective lower and upper bounds $\text{DEE}_m^{\text{lower}}(k)$ and $\text{DEE}_m^{\text{upper}}(k)$.

4.2 Experiments

We compute the deep ensemble equivalent (DEE) of various ensembling techniques for four popular deep architectures: VGG16 (Simonyan & Zisserman, 2014), PreResNet110/164 (He et al., 2016), and WideResNet28x10 (Zagoruyko & Komodakis, 2016) on CIFAR-10/100 datasets (Krizhevsky et al., 2009), and ResNet50 (He et al., 2016) on ImageNet dataset (Russakovsky et al., 2015). We use PyTorch (Paszke et al., 2017) for implementation of these models, building upon available public implementations. Our implementation closely matches the quality of methods that has been reported in original works. Technical details on training, hyperparameters and implementations can be found in Appendix D. We plan to make all computed metrics, source code and trained models publicly available.

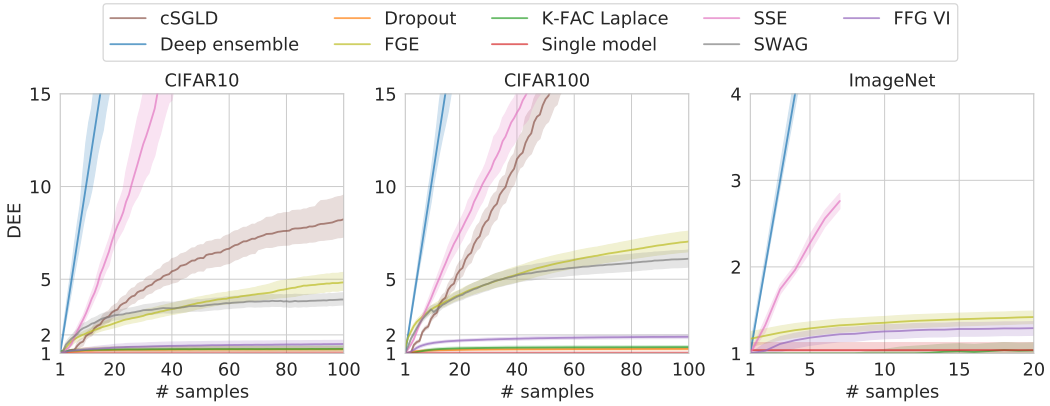


Figure 3: The deep ensemble equivalent for CIFAR10, CIFAR100 and ImageNet datasets averaged across different deep convolutional architectures. Area between the corresponding averaged lower and upper bounds of DEE is shaded. Note that the Deep Ensembles approximately correspond to the identity function, while other methods are divided into three groups, as discussed in Section 4.2.

As one can see on Figure 3, ensembling methods clearly fall into three categories. SSE and cSGLD outperform all other techniques except deep ensembles and enjoy a near-linear scaling of DEE with the number of samples. The investigation of weight-space trajectories of cSGLD and SSE (Huang et al., 2017; Zhang et al., 2019) suggests that these methods can efficiently explore different modes of the loss landscape. In terms of deep ensemble equivalent, these methods do not saturate unlike other methods that are bound to a single mode. More verbose results are presented in Appendix E.

In our experiments SSE typically outperforms cSGLD. This is mostly due to the fact that SSE has a much larger training budget. The cycle lengths and learning rates of SSE and cSGLD are comparable, however, SSE collects one snapshot per cycle while cSGLD collects three snapshots. This makes samples from SSE less correlated with each other while increasing the training budget. Both SSE and cSGLD can be adjusted to obtain a different trade-off between the training budget and the DEE-to-samples ratio. We reused the schedules provided in the original papers (Huang et al., 2017; Zhang et al., 2019).

Being more "local" methods, FGE and SWAG perform worse than SSE and cSGLD, but still significantly outperform "single-snapshot" methods like dropout, K-FAC Laplace approximation and variational inference. We hypothesize that by covering a single mode with a set of snapshots, FGE and SWAG provide a better fit for the local geometry than methods based on stochastic computation graphs. This implies that the performance of FGE and SWAG should be achievable by methods that approximate the geometry of a single mode. However, one might need more elaborate posterior approximations and better inference techniques in order to match the performance of FGE and SWAG by training a stochastic computation graph end-to-end (as opposed to SWAG that constructs a stochastic computation graph post-hoc).

4.3 Data augmentation improves Ensembles

Data augmentation is a time-honored technique that is widely used in deep learning, and is a crucial component for training modern DNNs. Test-time data augmentation have been used for a long time to improve the performance of convolutional networks. For example, multi-crop evaluation has long been a standard procedure for the ImageNet challenge (Simonyan & Zisserman, 2014; Szegedy et al., 2015; He et al., 2016). It, however, is not very popular in the literature on ensembling techniques in deep learning. In this section, we study the effect of test-time data augmentation on the aforementioned ensembling techniques.

We report the results on combination of ensembles and test-time data augmentation for CIFAR-10 in Table 1 (see Appendix G for results on CIFAR-100 and ImageNet). We sample one augmentation for each member of the ensemble to leave the test-time budget the same. Unsurprisingly, it consistently improves most ensembling methods. However, when combined with test-time data augmentation,

Model	cSGLD	Deep ensemble	FGE	K-FAC-L	Single model	SSE	SWAG	FFG VI	Dropout
ResNet110	0.115 vs 0.111↓	0.106 vs 0.105↓	0.121 vs 0.121≈	0.147 vs 0.130↓	0.150 vs 0.129↓	0.106 vs 0.106≈	0.126 vs 0.126↓	0.142 vs 0.130↓	
ResNet164	0.110 vs 0.108↓	0.100 vs 0.100≈	0.115 vs 0.115≈	0.142 vs 0.127↓	0.144 vs 0.124↓	0.104 vs 0.104≈	0.116 vs 0.116≈	0.141 vs 0.128↓	
VGG16	0.147 vs 0.146↓	0.138 vs 0.139↑	0.150 vs 0.150≈	0.200 vs 0.164↓	0.229 vs 0.170↓	0.137 vs 0.138↑	0.152 vs 0.152≈	0.184 vs 0.160↓	0.226 vs 0.171↓
WideResNet	0.099 vs 0.100↑	0.090 vs 0.094↑	0.102 vs 0.102≈	0.120 vs 0.111↓	0.124 vs 0.113↓		0.101 vs 0.101≈	0.117 vs 0.117≈	0.118 vs 0.111↓

Table 1: Negative calibrated log-likelihood for CIFAR10, w/o vs with test-time data augmentation.

"single-snapshot" methods like variational-inference, K-FAC Laplace and dropout start to perform very similarly to an augmented version of a single model. On CIFAR-10/100 the performance of powerful ensembles (deep ensembles, SSE, cSGLD, FGE and SWAG) is not affected by test-time augmentation, whereas on ImageNet we see a clear improvement across all methods. This is likely due to the fact that CIFAR images are small thus making data augmentation quite limited, whereas images from ImageNet allow for a large amount of diverse crops.

Interestingly, test-time data augmentation on ImageNet improves accuracy but decreases the (uncalibrated) log-likelihood of the deep ensembles (Figure 1, Table REF). It breaks the nearly optimal temperature of deep ensembles and requires temperature scaling to show the actual performance of the method, as discussed in Section 3.1. We show that test-time data augmentation with temperature scaling significantly improves predictive uncertainty of ensembling methods and should be considered as a baseline for them. It is a striking example that highlights the importance of temperature scaling. Our experiments demonstrate that ensembles may be severely miscalibrated by default while still providing superior predictive performance after calibration.

5 Discussion

We have explored the field of in-domain uncertainty estimation and performed an extensive evaluation of modern ensembling techniques. Our main findings can be summarized as follows:

- Temperature scaling is a must even for ensembles. While ensembles generally have better calibration out-of-the-box, they are not calibrated perfectly and can benefit from the procedure. Comparison of log-likelihoods of different ensembling methods without temperature scaling might not provide a fair ranking especially if some models happen to be miscalibrated.
- Many common metrics for measuring in-domain uncertainty are either unreliable (ECE and analogues) or cannot be used to compare different methods (AUC-ROC, AUC-PR for misclassification detection; accuracy-confidence curves). In order to perform a fair comparison of different methods, one needs to be cautious of these pitfalls.
- Many popular ensembling techniques require dozens of samples for test-time averaging, yet are essentially equivalent to a handful of independently trained models. Deep ensembles dominate other methods given a fixed test-time budget. The results indicate in particular that exploration of different modes in the loss landscape is crucial for good predictive performance.
- Methods that are stuck in a single mode are unable to compete with methods that are designed to explore different modes of the loss landscape. Would more elaborate posterior approximations and better inference techniques shorten this gap?
- Test-time data augmentation is a surprisingly strong baseline for in-domain uncertainty estimation and can significantly improve other methods without increasing training time or model size since data augmentation is usually already present during training.

Our takeaways are aligned with the take-home messages of (Ovadia et al., 2019) that relate to in-domain uncertainty estimation. We also observe a stable ordering of different methods in our experiments, and observe that deep ensembles with few members outperform methods based on stochastic computation graphs.

A large number of unreliable metrics inhibits a fair comparison of different methods. Because of this, we urge the community to aim for more reliable benchmarks in the numerous setups of uncertainty estimation.

REFERENCES

Andrei Atanov, Arsenii Ashukha, Dmitry Molchanov, Kirill Neklyudov, and Dmitry Vetrov. Uncertainty estimation via stochastic batch normalization. In *International Symposium on Neural Networks*, pp. 261–269. Springer, 2019.

Anoop Korattikara Balan, Vivek Rathod, Kevin P Murphy, and Max Welling. Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pp. 3438–3446, 2015.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, pp. 8224–8234, 2018.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pp. 4754–4765, 2018.

Yufei Cui, Wuguannan Yao, Qiao Li, Antoni B Chan, and Chun Jason Xue. Accelerating monte carlo bayesian inference via approximating predictive uncertainty over simplex. *arXiv preprint arXiv:1905.12194*, 2019.

Yarin Gal. *Uncertainty in deep learning*. PhD thesis, PhD thesis, University of Cambridge, 2016.

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

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, pp. 8789–8798, 2018.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1321–1330. JMLR. org, 2017.

Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):993–1001, 1990.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pp. 2575–2583, 2015.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017.

Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2218–2227. JMLR. org, 2017.

Wesley Maddox, Timur Garipov, Pavel Izmailov, Dmitry Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. *arXiv preprint arXiv:1902.02476*, 2019.

Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, pp. 7047–7058, 2018.

Marcin Możejko, Mateusz Susik, and Rafał Karczewski. Inhibited softmax for uncertainty estimation in neural networks. *arXiv preprint arXiv:1810.01861*, 2018.

Malik Sajjad Ahmed Nadeem, Jean-Daniel Zucker, and Blaise Hanczar. Accuracy-rejection curves (arcs) for comparing classification methods with a reject option. In *Machine Learning in Systems Biology*, pp. 65–81, 2009.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Jeremy Nixon, Mike Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. 2019.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

Joaquin Quinonero-Candela, Carl Edward Rasmussen, Fabian Sinz, Olivier Bousquet, and Bernhard Schölkopf. Evaluating predictive uncertainty challenge. In *Machine Learning Challenges Workshop*, pp. 1–27. Springer, 2005.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6 (1):1–114, 2012.

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

Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pp. 1100612. International Society for Optics and Photonics, 2019.

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.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. In *International Conference on Machine Learning (ICML)*, 2018.

Marcin B Tomczak, Siddharth Swaroop, and Richard E Turner. Neural network ensembles and variational inference revisited. In *1st Symposium on Advances in Approximate Bayesian Inference*, pp. 1–11, 2018.

Juozas Vaicenavicius, David Widmann, Carl Andersson, Fredrik Lindsten, Jacob Roll, and Thomas B Schön. Evaluating model calibration in classification. *arXiv preprint arXiv:1902.06977*, 2019.

Kuan-Chieh Wang, Paul Vicol, James Lucas, Li Gu, Roger Grosse, and Richard Zemel. Adversarial distillation of bayesian neural network posteriors. In *International Conference on Machine Learning*, pp. 5177–5186, 2018.

Sida Wang and Christopher Manning. Fast dropout training. In *international conference on machine learning*, pp. 118–126, 2013.

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

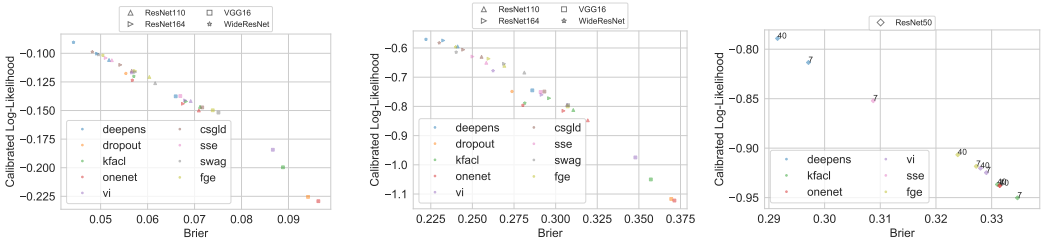
Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E Turner, José Miguel Hernández-Lobato, and Alexander L Gaunt. Deterministic variational inference for robust bayesian neural networks. 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for bayesian deep learning. *arXiv preprint arXiv:1902.03932*, 2019.

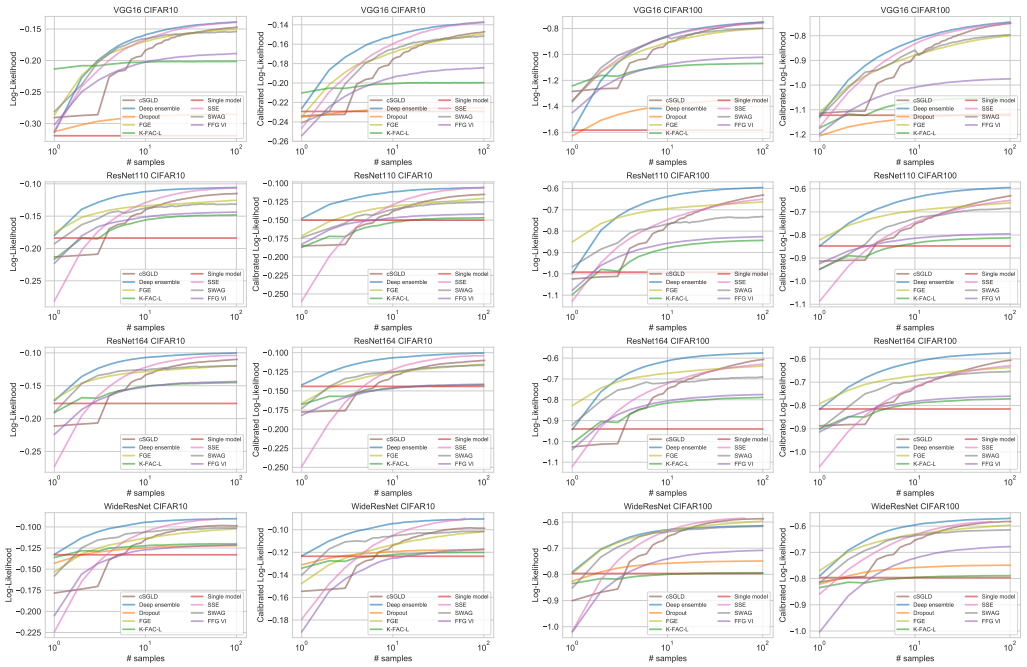
A CORRELATION OF LOG-LIKELIHOOD AND BRIER SCORES



(a) CIFAR-100 dataset $|\rho| = 0.986$ (b) CIFAR-100 dataset $|\rho| = 0.974$ (c) ImageNet dataset $|\rho| = 0.999$

Figure 4: The average log-likelihood vs the Brier score on a test dataset for different ensemble methods on CIFAR-10 (a) and CIFAR-10 (b) and ImageNet (c) datasets. While not being equivalent, these metrics demonstrate a strong linear correlation. The correlation coefficient is denoted as ρ.

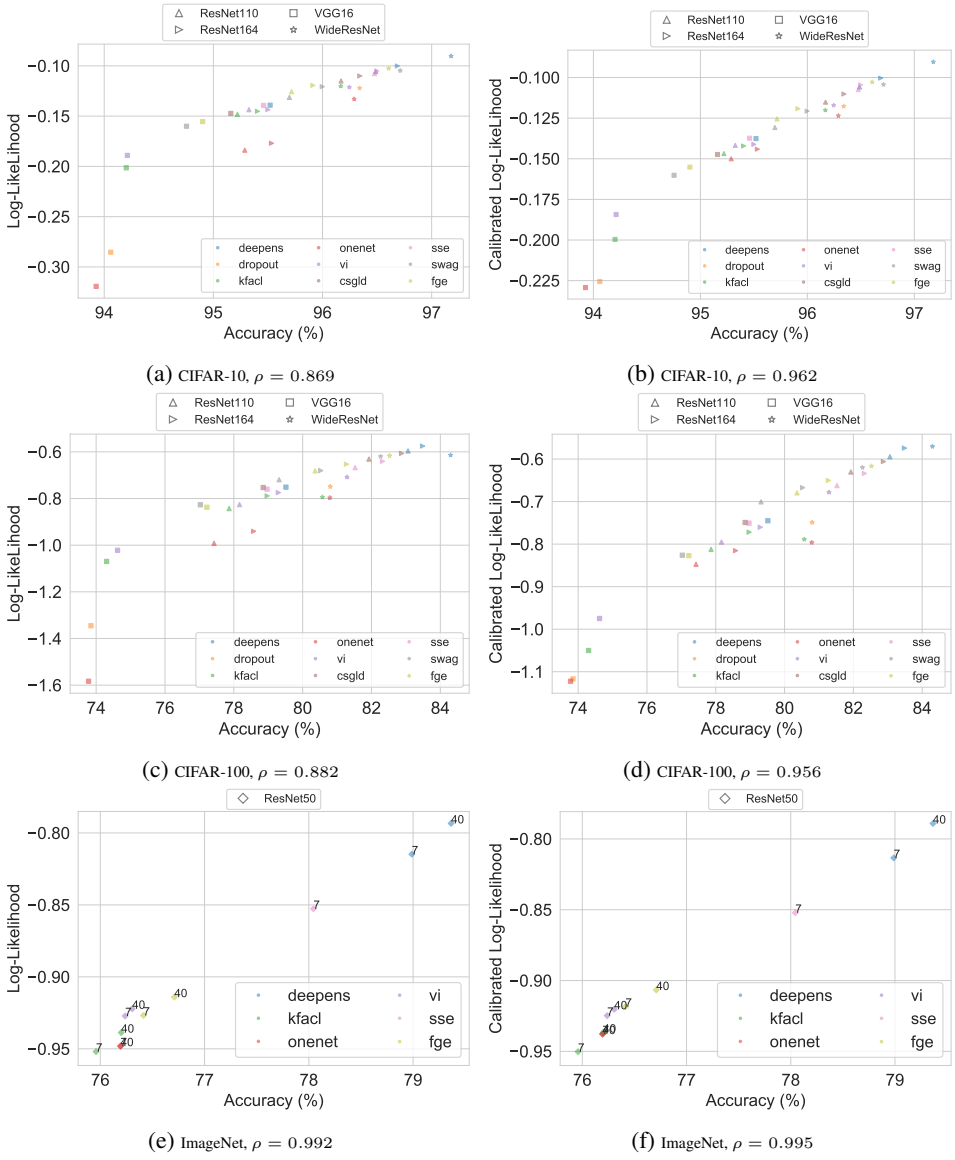
B LOG-LIKELIHOOD VS. CALIBRATED LOG-LIKELIHOOD



(a) CIFAR-10 (b) CIFAR-100

Figure 5: A side-by-side comparison of log-likelihood and calibrated log-likelihood on CIFAR-10 (a) and CIFAR-100 (b) datasets. On CIFAR-10, the performance of one network becomes close to dropout, variational inference (vi), and K-FAC Laplace approximation (kfacl) on all models except VGG. On CIFAR-100 on WideResNet and VGG deep ensembles move to the first position in the ranking after calibration. See Section 3.1 for details on the calibrated log-likelihood.

C CORRELATION OF ACCURACY AND LOG-LIKELIHOOD



(a) CIFAR-10, $\rho = 0.869$

(b) CIFAR-10, $\rho = 0.962$

(c) CIFAR-100, $\rho = 0.882$

(d) CIFAR-100, $\rho = 0.956$

(e) ImageNet, $\rho = 0.992$

(f) ImageNet, $\rho = 0.995$

Figure 6: Log-likelihood vs accuracy for different ensembles before (a, c, e) and after (b, d, f) calibration. Both plain log-likelihood and especially calibrated log-likelihood are highly correlated with accuracy.

D EXPERIMENTAL DETAILS

Implementations of deep ensembles, SWA, SWAG, FGE and K-FAC Laplace are heavily based on the original PyTorch implementations of SWA[1] and SWAG [2]. Implementations of cyclical MCMC and Snapshot Ensembles are based on the original implementation of cyclical MCMC [3]. Implementations of all architectures are based on torchvision.models[4].

[1] github.com/timgaripov/swa

[2] github.com/wjmaddox/swa_gaussian

[3] github.com/ruqizhang/csgmcmc/tree/master/experiments

[4] https://pytorch.org/docs/stable/torchvision/models.html

Model	lr_{init}	epochs	wd
VGG	0.05	400	5e-4
PreResNet110	0.1	300	3e-4
PreResNet164	0.1	300	3e-4
WideResNet28x10	0.1	300	5e-4

Table 2: Hyperparameters of various models trained on CIFARs

Implied probabilistic model Conventional neural networks for classification are usually trained using the average cross-entropy loss function with weight decay regularization hidden inside an optimizer in a deep learning framework like PyTorch. The actual underlying optimization problem can be written as follows:

$$L(w) = -\frac{1}{N}\sum_{i=1}^{N}\log\hat{p}(y_i^* \,|\, x_i, w) + \frac{\lambda}{2}\|w\|^2 \to \min_w, \tag{6}$$

where $\{(x_i, y_i^*)\}_{i=1}^N$ is the training dataset of N objects x_i with corresponding labels y_i^*, λ is the weight decay scale and $\hat{p}(y_i^* = j \,|\, x_i, w)$ denotes the probability that a neural network with parameters w assigns to class j when evaluated on object x_i.

The cross-entropy loss defines a likelihood function $p(y^* \,|\, x, w)$ and weight decay regularization, or L_2 regularization, corresponds to a certain Gaussian prior distribution $p(w)$. The whole optimization objective then corresponds to *maximum a posteriori* inference in the following probabilistic model:

$$p(y^*, w \,|\, x) = p(y^* \,|\, x, w)p(w), \tag{7}$$

$$\log p(y^* \,|\, x, w) = \log\prod_{i=1}^{N} p(y_i^* \,|\, x_i, w) = \sum_{i=1}^{N}\log\hat{p}(y_i^* \,|\, x_i, w), \tag{8}$$

$$\log p(w) = \frac{-N\lambda}{2}\|w\|^2 + \text{const} \iff p(w) = \mathcal{N}\left(w \,\middle|\, 0, (N\lambda)^{-1}I\right) \tag{9}$$

As many of the considered methods are probabilistic in nature, we use the same probabilistic model for all of them. We use the SoftMax-based likelihood for all models, and use the fully-factorized zero-mean Gaussian prior distribution with variances $\sigma^2 = (N\lambda)^{-1}$, where the number of objects N and the weight decay scale λ are dictated by the particular datasets and neural architectures, as defined in the following paragraph. In order to make the result comparable across all ensembling techniques, we use the same prabapilistic model for all methods, choosing fixed weight decay parameters for each architecture.

Conventional networks On CIFAR-10/100 datasets all networks were trained by SGD optimizer with batch size of 128, momentum 0.9 and model-specific parameters i.e., initial learning rate (lr_{init}), weight decay (wd), and number of optimization epoch (epoch). The specific hyperparameters are shown in Table 2. The models used a unified learning rate scheduler that is shown in equation 10. All models have been trained using data augmentation that consists of horizontal flips, random crop of size 32 with padding 4. The standard data normalization has also been applied. Weight decays, initial learning rates, and the learning rate scheduler were taken from (Garipov et al., 2018) paper. Compared with hyperparameters of (Garipov et al., 2018), the number of optimization epochs has been increased since we found that all models were underfitted. While original WideResNet28x10 includes number of dropout layers with $p = 0.3$ and 200 training epoch, in this setting we find that WideResNet28x10 underfits, and requires a longer training. Thus, we used $p = 0$, effectively it does not affect the final performance of the model in our experiments, but reduces training time.

$$\texttt{lr}(i) = \begin{cases} lr_{init}, & i \in [0, 0.5 \cdot \text{epochs}] \\ lr_{init} \cdot (1.0 - 0.99 * (i/\text{epochs} - 0.5)/0.4), & i \in [0.5 \cdot \text{epochs}, 0.9 \cdot \text{epochs}] \\ lr_{init} \cdot 0.01, & \text{otherwise} \end{cases} \tag{10}$$

On ImageNet dataset we used ResNet50 examples with a default hyperparameters from PyTorch examples [5]. Specifically SGD optimizer with momentum 0.9, batch size of 256, initial learning

[5] github.com/pytorch/examples/tree/ee964a2/imagenet

rate 0.1, and with decay 1e-4. The training also includes data augmentation random crop of size 224×224, horizontal flips, and normalization, and learning rate scheduler $lr = lr_{init} \cdot 0.1^{\text{epoch}//30}$, where $//$ denotes integer division. We only deviated from standard parameters by increasing the number of training epochs from 90 to 130. Or models achived top-1 error of 23.81 ± 0.15 that closely matches accuracy of the ResNet50 probided by PyTorch which is 23.85 [6]. Training of one model on a single NVIDIA Tesla V100 GPU takes approximately 5.5 days.

Deep Ensembles Deep ensembles (Lakshminarayanan et al., 2017) average the predictions across networks trained independently starting from different initializations. To obtain Deep Ensemble we repeat the procedure of training standard networks 128 times for all architectures on CIFAR-10 and CIFAR-100 datasets (1024 networks over all) and 50 times for ImageNet dataset. Every single member of Deep Ensembles were actually trained with exactly the same hyperparameters as conventional models of the same arhitecture.

Dropout The binary dropout (or MC dropout) (Srivastava et al., 2014; Gal & Ghahramani, 2016) is one of the most known ensembling techniques. It puts a multiplicative Bernoulli noise with parameter p over activations of ether fully-connected or convolutional layer, averaging predictions of the network w.r.t. the noise during test. The dropout layers have been applied to VGG, and WideResNet networks on CIFAR-10 and CIFAR-100 datasets. For VGG the dropout has been applied to fully-connected (fc) layers with $p = 0.5$, overall two dropout layers, one before the first fc-layer and one before the second one. While original version of VGG for CIFARs (Zagoruyko, 2015) exploits more dropout layers, we observed that any additional dropout layer deteriorates the performance on the model in ether deterministic or stochastic mode. For WideResNet network we applied dropout consistently with the original paper (Zagoruyko & Komodakis, 2016) with $p = 0.3$. The dropout usually increases the time to convergence, thus, VGG and WideResNet networks with dropout was trained for 400 epoch instead of 300 epoch for deterministic case. The all other hyperparameters was the same as in case of conventional models.

Variational Inference The VI approximates a true posterior distribution $p(w \,|\, Data)$ with a tractable variational approximation $q_\theta(w)$, by maximizing so-called variational lower bound $\mathcal{L}$ (eq. 11) w.r.t. parameters of variational approximation θ. We used fully-factorized Gaussian approximation $q(w)$, and Gaussian prior distribution $p(w)$.

$$\mathcal{L}(\theta) = \mathbb{E}_q \log p(y^* \,|\, x, w) - KL(q_\theta(w) \,||\, p(w)) \to \max_\theta \tag{11}$$

$$q(w) = \mathcal{N}(w \,|\, \mu, \text{diag}(\sigma^2)) \quad p(w) = N(w \,|\, 0, \text{diag}(\sigma_p^2)), \quad \text{where } \sigma_p^2 = (N \cdot \text{wd})^{-1} \tag{12}$$

In the case of such a prior $p(w)$ the probabilistic model remains consistent with conventional training which corresponds to MAP inference in the same probabilistic model. We used variational inference for both convolutional and fully-connected layers, where variances of the weights was parameterized by $\log \sigma$. For fully-connected layers we applied the LRT (Kingma et al., 2015).

While variational inference provide a theoretical grounded way to approximate a true posterior, on practice, it tends to underfit deep learning models (Kingma et al., 2015). The following tricks are applied to deal with it: pre-training (Molchanov et al., 2017) or equivalently annealing of β (Sønderby et al., 2016), scaling down β (Kingma et al., 2015; Ullrich et al., 2017).

Consistently with the practical tricks we use a pre-training, specifically, we initialize μ with a snapshot of the weights of pretrained conventional model, and initialize $\log \sigma$ with model-specific constant $\log \sigma_{init}$. The KL-divergence – except the term that corresponds to a weight decay – was scaled on model specific parameter β. The weigh decay term was implemented as a part of the optimizer. We used a fact that KL-divergence between two Gaussian distributions can be rewritten as two terms one of which is equal to wd regularization.

On CIFAR-10 and CIFAR-100 we used β 1e-4 for VGG, ResNet100 and ResNet164 networks, and β 1e-5 for WideResNet. The initialization of log-variance $\log \sigma_{init}$ was set to -5 for all models. Parameters μ were optimized with conventional SGD (with the same parameters as conventional networks, except initial learning rate lr_{init} that was set to 1e-3). We used a separate Adam optimizer with constant learning rate 1e-3 to optimize log-variances of the weights $\log \sigma$. The training was held

[6] pytorch.org/docs/stable/torchvision/models.html

for 100 epochs, that corresponds to 400 epochs of training (including pre-training). On ImageNet we used $\beta =$ 1e-3, $lr_{init} = 0.01$, $\log \sigma_{init} = -6$, and held training for a 45 epoch form a per-trained model.

K-FAC Laplace The Laplace approximation uses the curvature information of the appropriately scaled loss function to construct a Gaussian approximation to the posterior distribution. Ideally, one would use the Hessian of the loss function as the covariance matrix and use the maximum a posteriori estimate w^{MAP} as the mean of the Gaussian approximation:

$$\log p(w \,|\, x, y^*) = \log p(y^* \,|\, x, w) + \log p(w) + \text{const} \tag{13}$$

$$w^{MAP} = \arg\max_w \log p(w \,|\, x, y^*); \quad \Sigma = -\nabla\nabla \log p(w \,|\, x, y^*) \tag{14}$$

$$p(w \,|\, x, y^*) \approx \mathcal{N}(w \,|\, w^{MAP}, \Sigma) \tag{15}$$

In order to keep the method scalable, we use the Fisher Information Matrix as an approximation to the true Hessian (Martens & Grosse, 2015). For K-FAC Laplace, we use the whole dataset to construct an approximation to the empirical Fisher Information Matrix, and use the π correction to reduce the bias (Ritter et al., 2018; Martens & Grosse, 2015). Following (Ritter et al., 2018), we find the optimal noise scale for K-FAC Laplace on a held-out validation set by averaging across five random initializations. We then reuse this scale for networks trained without a hold-out validation set. We report the optimal values of scales in Table 3. Note that the optimal scale is different depending on whether we use test-time data augmentation or not. Since the data augmentation also introduces some amount of additional noise, the optimal noise scale for K-FAC Laplace with data augmentation is lower.

Snapshot Ensembles Snapshot Ensembles (SSE) (Huang et al., 2017) is a simple example of an array of methods which collect samples from a training trajectory of a network in weight space to construct an ensemble. Samples are collected in a cyclical manner: each cycle learning rate goes from a large value to near-zero and weights snapshot is taken at the end of the cycle. SSE uses SGD with a cosine learning schedule defined as follows:

$$\alpha(t) = \frac{\alpha_0}{2}\left(\cos\left(\frac{\pi \bmod (t-1, \lceil T/M \rceil)}{\lceil T/M \rceil}\right) + 1\right), \tag{16}$$

where α_0 is the initial learning rate, T is the total number of training iterations and M is the number of cycles.

On CIFAR-10/100 parameters from the original paper are reused, length of cycle is 40 epochs, maximum learning rate is 0.2, batch size is 64. On ResNet50 on ImageNet we used hyperparameters from the original paper which are 45 epoch per cycle, maximum learning rate 0.1, and cosine scheduler of learning rate (eq. 16). All other parameters are equal to the ones as were used conventional networks.

Cyclical SGLD Cyclical Stochastic Gradient Langevin Dynamics (cSGLD) (Zhang et al., 2019) is a state-of-the-art ensembling method for deep neural networks pertaining to stochastic Markov Chain Monte Carlo family of methods. It bears similarity to SSE, e.g. it employs SGD with a learning rate schedule described with the equation 16 and training is cyclic in the same manner. Its main differences from SSE are introducing gradient noise and capturing several snapshots per cycle, both of which aid in sampling from posterior distribution over neural network weights efficiently.

Some parameters from the original paper are reused: length of cycle is 50 epochs, maximum learning rate is 0.5, batch size is 64. Number of epochs with gradient noise per cycle is 3 epochs. This was found to yield much higher predictive performance and better uncertainty estimation compared to the original paper choice of 10 epochs for CIFAR-10 and 3 epochs for CIFAR-100.

Finally, cyclical Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) which reportedly has marginally better performance compared with cyclical SGLD (Zhang et al., 2019) could not be reproduced with a wide range of values of SGD momentum term. Because of this, we only include cyclical SGLD in our benchmark.

FGE Fast Geometric Ensembling (FGE) is an ensembling method that is similar to SSE in that it collects samples from a training trajectory of a network in weight space to construct an ensemble.

Architecture	Optimal noise scale			
	CIFAR10	CIFAR10-aug	CIFAR100	CIFAR100-aug
VGG16BN	0.042	0.042	0.100	0.100
PreResNet110	0.213	0.141	0.478	0.401
PreResNet164	0.120	0.105	0.285	0.225
WideResNet28x10	0.022	0.018	0.022	0.004

Table 3: Optimal noise scale for K-FAC Laplace for different datasets and architectures. For ResNet50 on ImageNet, the optimal scale found was 2.0 with test-time augmentation and 6.8 without test-time augmentation.

Its main differences from SSE are pretraining, a short cycle length and a piecewise-linear learning rate schedule

$$\alpha(i) = \begin{cases} (1 - 2t(i))\alpha_1 + 2t(i)\alpha_2 & 0 < t(i) \leq \frac{1}{2} \\ (2 - 2t(i))\alpha_2 + (2t(i) - 1)\alpha_1 & \frac{1}{2} < t(i) \leq 1 \end{cases} . \tag{17}$$

Original hyperparameters are reused. Model pretraining is done with SGD for 160 epochs according to the standard learning rate schedule described in equation 10 with maximum learning rates from Table 2. After that, a desired number of FGE cycles is done with one snapshot per cycle collected. Learning rate in a cycle is changed with parameters $\alpha_1 = 1e-2, \alpha_2 = 5e-4$, cycle length of 2 epochs for VGG and $\alpha_1 = 5e-2, \alpha_2 = 5e-4$, cycle length of 4 epochs for other networks. Batch size is 128.

SWAG SWA-Gaussian (SWAG) (Maddox et al., 2019) is an ensembling method based on fitting a Gaussian distribution to model weights on the SGD training trajectory and sampling from this distribution to construct an ensemble.

Like FGE, SWAG has a pretraining stage which is done according to the standard learning rate schedule described in equation 10 with maximum learning rates from Table 2. After that, training continues with a constant learning rate of 1e-2 for all models except for PreResNet110 and PreResNet164 on CIFAR-100 where it continues with a constant learning rate of 5e-2 in accordance with the original paper. Rank of the empirical covariance matrix which is used for estimation of Gaussian distribution parameters is set to be 20.

E DEEP ENSEMBLE EQUIVALENT

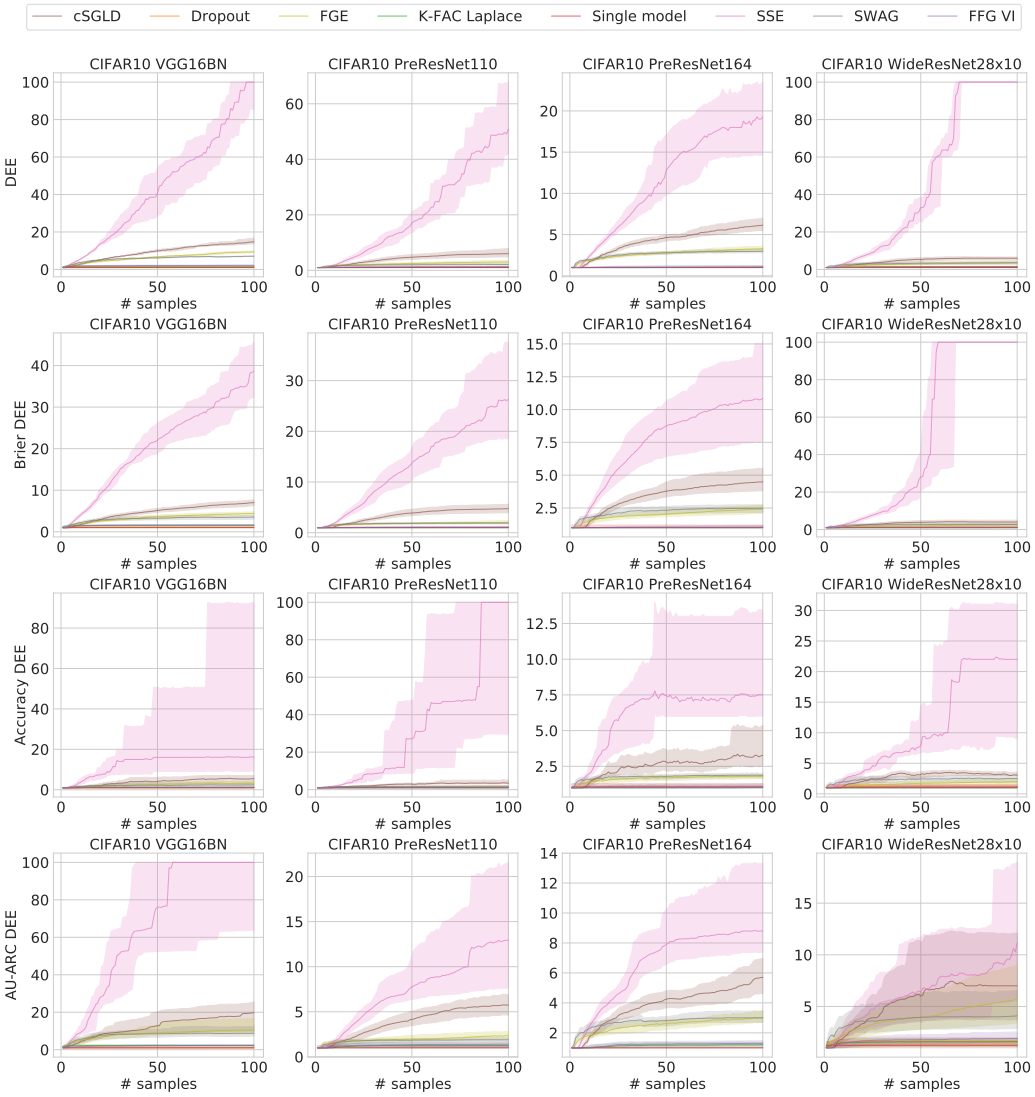


Figure 7: The deep ensemble equivalent of various ensembling techniques on CIFAR-10. Solid lines: mean DEE for different methods and architectures. Area between $\mathrm{DEE}^{\mathrm{lower}}$ and $\mathrm{DEE}^{\mathrm{upper}}$ is shaded. Lines 2–4 correspond to DEE based on other metrics, defined similarly to the log-likelihood-based DEE. Note that while the actual scale of DEE varies from metric to metric, the ordering of different methods and the overall behaviour of the lines remain the same.
SSE outperforms deep ensembles on CIFAR-10 on the WideResNet architecture. It possibly indicates that the cosine learning rate schedule of SSE is more suitable for this architecture than the piecewise-linear learning rate schedule used in deep ensembles. We will change the learning rate schedule on WideResNets to a more suitable option in further revisions of the paper.

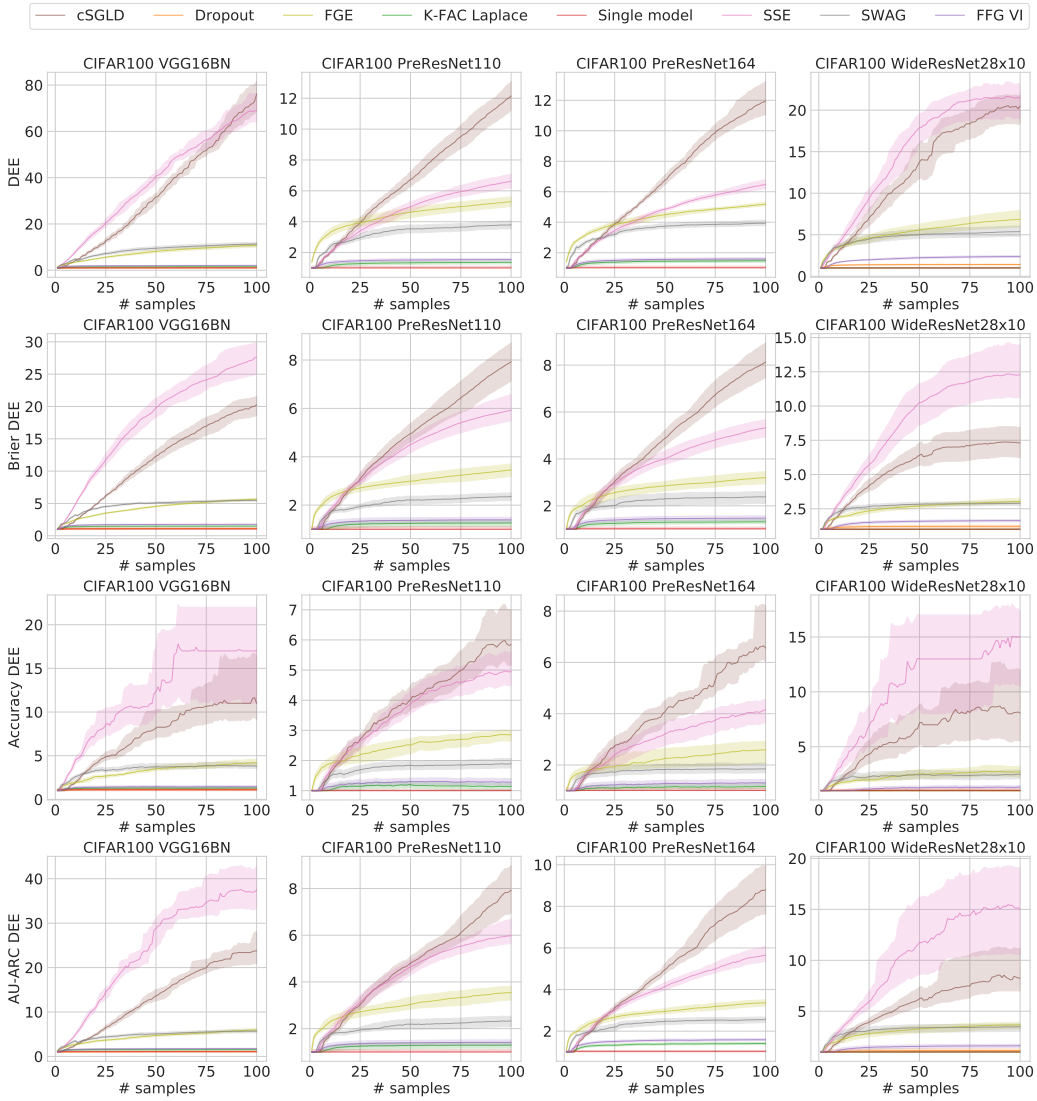


Figure 8: The deep ensemble equivalent of various ensembling techniques on CIFAR-100. Solid lines: mean DEE for different methods and architectures. Area between $\mathrm{DEE}^{\mathrm{lower}}$ and $\mathrm{DEE}^{\mathrm{upper}}$ is shaded. Lines 2–4 correspond to DEE based on other metrics, defined similarly to the log-likelihood-based DEE. Note that while the actual scale of DEE varies from metric to metric, the ordering of different methods and the overall behaviour of the lines remain the same.

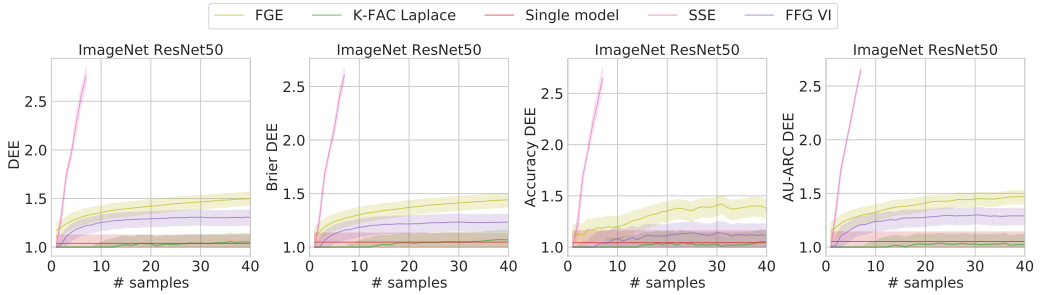


Figure 9: The deep ensemble equivalent of various ensembling techniques on ImageNet. Solid lines: mean DEE for different methods and architectures. Area between $\text{DEE}^{\text{lower}}$ and $\text{DEE}^{\text{upper}}$ is shaded. Columns 2–4 correspond to DEE based on other metrics, defined similarly to the log-likelihood-based DEE. Note that while the actual scale of DEE varies from metric to metric, the ordering of different methods and the overall behaviour of the lines remain the same.

F METRIC VALUES

	Error (%) on CIFAR10 dataset (100 samples)								
Model	cSGLD*	Deep ensemble*	FGE*	K-FAC Laplace*	Single model*	SSE*	SWAG*	FFG VI*	Dropout*
ResNet110	3.83±0.02	3.51±0.01	4.11±0.08	4.78±0.08	4.71±0.16	3.47±0.10	4.09±0.08	4.67±0.12	
ResNet164	3.66±0.06	3.31±0.02	3.87±0.14	4.60±0.09	4.47±0.12	3.37±0.05	3.86±0.07	4.50±0.05	
VGG16	4.84±0.04	4.48±0.05	4.99±0.11	5.80±0.19	6.07±0.20	4.54±0.05	5.04±0.06	5.79±0.13	5.94±0.11
WideResNet	3.19±0.04	2.82±0.02	3.34±0.06	3.83±0.16	3.71±0.10		3.27±0.10	3.75±0.06	3.66±0.16
	cLog-LikeLihood on CIFAR10 dataset (100 samples)								
Model	cSGLD*	Deep ensemble*	FGE*	K-FAC Laplace*	Single model*	SSE*	SWAG*	FFG VI*	Dropout*
ResNet110	-0.115±0.001	-0.106±0.000	-0.121±0.001	-0.147±0.003	-0.150±0.003	-0.106±0.001	-0.126±0.005	-0.142±0.002	
ResNet164	-0.110±0.002	-0.100±0.000	-0.115±0.001	-0.142±0.004	-0.144±0.003	-0.104±0.001	-0.116±0.002	-0.141±0.002	
VGG16	-0.147±0.003	-0.138±0.000	-0.150±0.002	-0.200±0.008	-0.229±0.005	-0.137±0.001	-0.152±0.001	-0.184±0.001	-0.226±0.003
WideResNet	-0.099±0.002	-0.090±0.000	-0.102±0.001	-0.120±0.003	-0.124±0.003		-0.101±0.002	-0.117±0.002	-0.118±0.002
	cBrier on CIFAR10 dataset (100 samples)								
Model	cSGLD*	Deep ensemble*	FGE*	K-FAC Laplace*	Single model*	SSE*	SWAG*	FFG VI*	Dropout*
ResNet110	0.057±0.001	0.052±0.000	0.060±0.001	0.071±0.001	0.071±0.002	0.052±0.001	0.062±0.002	0.069±0.001	
ResNet164	0.054±0.001	0.049±0.000	0.057±0.001	0.068±0.001	0.068±0.001	0.051±0.000	0.057±0.001	0.068±0.001	
VGG16	0.072±0.002	0.066±0.000	0.074±0.001	0.089±0.002	0.096±0.002	0.067±0.000	0.075±0.001	0.087±0.001	0.094±0.001
WideResNet	0.048±0.001	0.044±0.000	0.050±0.000	0.057±0.002	0.057±0.001		0.050±0.001	0.057±0.001	0.055±0.001
	cAURC on CIFAR10 dataset (100 samples)								
Model	cSGLD*	Deep ensemble*	FGE*	K-FAC Laplace*	Single model*	SSE*	SWAG*	FFG VI*	Dropout*
ResNet110	0.0037±0.00	0.0032±0.00	0.0041±0.00	0.0051±0.00	0.0054±0.00	0.0035±0.00	0.0043±0.00	0.0049±0.00	
ResNet164	0.0035±0.00	0.0031±0.00	0.0039±0.00	0.0049±0.00	0.0053±0.00	0.0034±0.00	0.0038±0.00	0.0049±0.00	
VGG16	0.0051±0.00	0.0046±0.00	0.0053±0.00	0.0076±0.00	0.0109±0.00	0.0045±0.00	0.0054±0.00	0.0076±0.00	0.0116±0.00
WideResNet	0.0031±0.00	0.0029±0.00	0.0031±0.00	0.0040±0.00	0.0043±0.00		0.0031±0.00	0.0037±0.00	0.0039±0.00

Table 4: The joint table of results for CIFAR10. Note that std of deep ensembles is underestimated since we are choosing 100 random samples from a subset of approx. 130 independently trained networks.

	Error (%) on CIFAR100 dataset (100 samples)								
Model	cSGLD*	Deep ensemble*	FGE*	K-FAC Laplace*	Single model*	SSE*	SWAG*	FFG VI*	Dropout*
ResNet110	18.07±0.16	16.94±0.04	19.19±0.21	22.14±0.29	22.57±0.21	18.24±0.27	20.28±0.21	21.83±0.13	
ResNet164	17.13±0.18	16.52±0.06	18.36±0.13	21.03±0.38	21.42±0.32	17.63±0.17	19.33±0.24	20.71±0.09	
VGG16	21.15±0.11	20.49±0.08	22.16±0.23	25.70±0.38	26.22±0.32	21.03±0.10	22.38±0.25	25.38±0.24	26.15±0.16
WideResNet	16.29±0.10	15.70±0.05	17.12±0.16	19.43±0.21	19.21±0.26		17.10±0.22	18.71±0.31	19.20±0.28
	cLog-LikeLihood on CIFAR100 dataset (100 samples)								
Model	cSGLD*	Deep ensemble*	FGE*	K-FAC Laplace*	Single model*	SSE*	SWAG*	FFG VI*	Dropout*
ResNet110	-0.630±0.002	-0.595±0.001	-0.661±0.004	-0.812±0.010	-0.848±0.009	-0.651±0.002	-0.684±0.008	-0.795±0.002	
ResNet164	-0.606±0.005	-0.574±0.001	-0.637±0.004	-0.772±0.007	-0.815±0.011	-0.629±0.005	-0.654±0.004	-0.760±0.001	
VGG16	-0.749±0.004	-0.745±0.001	-0.800±0.002	-1.050±0.008	-1.122±0.015	-0.751±0.003	-0.796±0.003	-0.975±0.004	-1.117±0.004
WideResNet	-0.583±0.004	-0.570±0.001	-0.596±0.003	-0.789±0.006	-0.796±0.012		-0.614±0.005	-0.678±0.011	-0.749±0.006
	cBrier on CIFAR100 dataset (100 samples)								
Model	cSGLD*	Deep ensemble*	FGE*	K-FAC Laplace*	Single model*	SSE*	SWAG*	FFG VI*	Dropout*
ResNet110	0.256±0.002	0.241±0.000	0.269±0.002	0.311±0.004	0.319±0.003	0.258±0.001	0.281±0.003	0.307±0.000	
ResNet164	0.244±0.002	0.232±0.000	0.260±0.001	0.296±0.003	0.305±0.004	0.250±0.002	0.269±0.001	0.292±0.000	
VGG16	0.293±0.001	0.286±0.000	0.307±0.001	0.357±0.003	0.371±0.004	0.291±0.001	0.307±0.002	0.348±0.001	0.369±0.002
WideResNet	0.230±0.001	0.222±0.000	0.240±0.001	0.281±0.003	0.280±0.003		0.240±0.002	0.262±0.003	0.274±0.003
	cAURC on CIFAR100 dataset (100 samples)								
Model	cSGLD*	Deep ensemble*	FGE*	K-FAC Laplace*	Single model*	SSE*	SWAG*	FFG VI*	Dropout*
ResNet110	0.0421±0.00	0.0382±0.00	0.0456±0.00	0.0607±0.00	0.0648±0.00	0.0426±0.00	0.0496±0.00	0.0593±0.00	
ResNet164	0.0388±0.00	0.0359±0.00	0.0429±0.00	0.0556±0.00	0.0600±0.00	0.0402±0.00	0.0455±0.00	0.0534±0.00	
VGG16	0.0528±0.00	0.0508±0.00	0.0570±0.00	0.0779±0.00	0.0855±0.00	0.0518±0.00	0.0572±0.00	0.0741±0.00	0.0857±0.00
WideResNet	0.0343±0.00	0.0324±0.00	0.0364±0.00	0.0498±0.00	0.0499±0.00		0.0366±0.00	0.0439±0.00	0.0476±0.00

Table 5: The joint table of results for CIFAR100. Note that std of deep ensembles is underestimated since we are choosing 100 random samples from a subset of approx. 130 independently trained networks.

	Error (%) on ImageNet dataset for different number of samples					
Model (# samples)	Deep ensemble	FGE	K-FAC-L	Single model	SSE	FFG VI
ResNet50 (7 samples)	21.01±0.0485	23.59	24.04±0.1230	23.81±0.1526	21.96	23.76±0.0660
ResNet50 (40 samples)	20.64±0.0290	23.29	23.80±0.0113	23.81±0.1526		23.69±0.0198
ResNet50 (50 samples)	20.63±0.0000		23.82±0.0269	23.81±0.1526		23.69±0.0297
	cLog-Likelihood on ImageNet dataset for diferent number of samples					
Model (# samples)	Deep ensemble*	FGE	K-FAC-L	Single model	SSE	FFG VI
ResNet50 (7 samples)	-0.813±0.0008	-0.918	-0.950±0.0130	-0.938±0.0046	-0.852	-0.925±0.0004
ResNet50 (40 samples)	-0.789±0.0005	-0.907	-0.937±0.0043	-0.938±0.0046		-0.920±0.0000
ResNet50 (50 samples)	-0.788±0.0000		-0.936±0.0015	-0.938±0.0046		-0.920±0.0002
	cBrier on ImageNet dataset for diferent number of samples					
Model (# samples)	Deep ensemble	FGE	K-FAC-L	Single model*	SSE	FFG VI
ResNet50 (7 samples)	0.297±0.0002	0.327	0.335±0.0028	0.331±0.0015	0.309	0.329±0.0002
ResNet50 (40 samples)	0.292±0.0002	0.324	0.331±0.0007	0.331±0.0015		0.328±0.0000
ResNet50 (50 samples)	0.291±0.0000		0.331±0.0002	0.331±0.0015		0.328±0.0000
	cAURC on ImageNet dataset for diferent number of samples					
Model (# samples)	Deep ensemble	FGE	K-FAC-L	Single model	SSE	FFG VI
ResNet50 (7 samples)	0.0582±0.00	0.0684±nan	0.0714±0.00	0.0701±0.00	0.0619±nan	0.0688±0.00
ResNet50 (40 samples)	0.0565±0.00	0.0673±nan	0.0703±0.00	0.0701±0.00		0.0685±0.00
ResNet50 (50 samples)	0.0564±0.00		0.0702±0.00	0.0701±0.00		0.0684±0.00

Table 6: The joint table of results for ImageNet. Note that std of deep ensembles is underestimated since we are choosing 50 random samples from a subset of approx. 60 independently trained networks.

G METRIC VALUES WITH TEST-TIME AUGMENTATION

	Error (%) on CIFAR10 dataset before and after data augmentation (100 samples)								
Model	cSGLD	Deep ensemble	FGE	K-FAC-L	Single model	SSE	SWAG	FFG VI	Dropout
ResNet110	3.83 vs 3.64↓	3.51 vs 3.44↓	4.11 vs 4.11≈	4.78 vs 4.17↓	4.71 vs 4.13↓	3.47 vs 3.47≈	4.09 vs 4.09≈	4.67 vs 4.25↓	
ResNet164	3.66 vs 3.43↓	3.31 vs 3.29↓	3.87 vs 3.87≈	4.60 vs 4.08↓	4.47 vs 3.92↓	3.37 vs 3.37≈	3.86 vs 3.86≈	4.50 vs 4.05↓	
VGG16	4.84 vs 4.88↑	4.48 vs 4.62↑	4.99 vs 4.99≈	5.80 vs 5.17↓	6.07 vs 5.27↓	4.54 vs 4.68↑	5.04 vs 5.04≈	5.79 vs 5.26↓	5.94 vs 5.35↓
WideResNet	3.19 vs 3.22↑	2.82 vs 2.92↑	3.34 vs 3.34≈	3.83 vs 3.50↓	3.71 vs 3.51↓		3.27 vs 3.27≈	3.75 vs 3.69↓	3.66 vs 3.47↓
	Negative cLog-Likelihood on CIFAR10 dataset before and after data augmentation (100 samples)								
Model	cSGLD	Deep ensemble	FGE	K-FAC-L	Single model	SSE	SWAG	FFG VI	Dropout
ResNet110	0.115 vs 0.111↓	0.106 vs 0.105↓	0.121 vs 0.121≈	0.147 vs 0.130↓	0.150 vs 0.129↓	0.106 vs 0.106≈	0.126 vs 0.126↓	0.142 vs 0.130↓	
ResNet164	0.110 vs 0.108↓	0.100 vs 0.100≈	0.115 vs 0.115≈	0.142 vs 0.127↓	0.144 vs 0.124↓	0.104 vs 0.104≈	0.116 vs 0.116≈	0.141 vs 0.128↓	
VGG16	0.147 vs 0.146↓	0.138 vs 0.139↑	0.150 vs 0.150≈	0.200 vs 0.164↓	0.229 vs 0.170↓	0.137 vs 0.138↑	0.152 vs 0.152≈	0.184 vs 0.160↓	0.226 vs 0.171↓
WideResNet	0.099 vs 0.100↑	0.090 vs 0.094↑	0.102 vs 0.102≈	0.120 vs 0.111↓	0.124 vs 0.113↓		0.101 vs 0.101≈	0.117 vs 0.117↓	0.118 vs 0.111↓
	cBrier on CIFAR10 dataset before and after data augmentation (100 samples)								
Model	cSGLD	Deep ensemble	FGE	K-FAC-L	Single model	SSE	SWAG	FFG VI	Dropout
ResNet110	0.057 vs 0.055↓	0.052 vs 0.051↓	0.060 vs 0.060≈	0.071 vs 0.063↓	0.071 vs 0.062↓	0.052 vs 0.052≈	0.062 vs 0.062≈	0.069 vs 0.064↓	
ResNet164	0.054 vs 0.053↓	0.049 vs 0.049≈	0.057 vs 0.057≈	0.068 vs 0.062↓	0.068 vs 0.059↓	0.051 vs 0.051≈	0.057 vs 0.057≈	0.068 vs 0.062↓	
VGG16	0.072 vs 0.072≈	0.066 vs 0.068↑	0.074 vs 0.074≈	0.089 vs 0.077↓	0.096 vs 0.080↓	0.067 vs 0.068↑	0.075 vs 0.075≈	0.087 vs 0.078↓	0.094 vs 0.080↓
WideResNet	0.048 vs 0.049↑	0.044 vs 0.046↑	0.050 vs 0.050≈	0.057 vs 0.053↓	0.057 vs 0.053↓		0.050 vs 0.050≈	0.057 vs 0.056≈	0.055 vs 0.053↓
	cAURC on CIFAR10 dataset before and after data augmentation (100 samples)								
Model	cSGLD	Deep ensemble	FGE	K-FAC-L	Single model	SSE	SWAG	FFG VI	Dropout
ResNet110	0.0037 vs 0.0036↓	0.0032 vs 0.0033↑	0.0041 vs 0.0041↑	0.0051 vs 0.0044↓	0.0054 vs 0.0045↓	0.0035 vs 0.0035≈	0.0043 vs 0.0043≈	0.0049 vs 0.0044↓	
ResNet164	0.0035 vs 0.0035≈	0.0031 vs 0.0032↑	0.0039 vs 0.0039≈	0.0049 vs 0.0043↓	0.0053 vs 0.0042↓	0.0034 vs 0.0034≈	0.0038 vs 0.0038≈	0.0049 vs 0.0043↓	
VGG16	0.0051 vs 0.0052↑	0.0046 vs 0.0046≈	0.0053 vs 0.0053≈	0.0076 vs 0.0059↓	0.0109 vs 0.0069↓	0.0045 vs 0.0046↑	0.0054 vs 0.0054≈	0.0076 vs 0.0059↓	0.0116 vs 0.0068↓
WideResNet	0.0031 vs 0.0031↑	0.0029 vs 0.0030↑	0.0031 vs 0.0031≈	0.0040 vs 0.0036↓	0.0043 vs 0.0038↓		0.0031 vs 0.0031≈	0.0037 vs 0.0038↑	0.0039 vs 0.0036↓

Table 7: Results before and after data augmentation on CIFAR10.

	Error (%) on CIFAR100 dataset (100 samples)								
Model	cSGLD	Deep ensemble	FGE	K-FAC-L	Single model	SSE	SWAG	FFG VI	Dropout
ResNet110	18.07 vs 17.66↓	16.94 vs 16.82↓	19.19 vs 19.19≈	22.14 vs 20.99↓	22.57 vs 21.05↓	18.24 vs 18.24≈	20.28 vs 20.28≈	21.83 vs 20.84↓	
ResNet164	17.13 vs 16.94↓	16.52 vs 16.26↓	18.36 vs 18.36≈	21.03 vs 20.18↓	21.42 vs 20.00↓	17.63 vs 17.63≈	19.33 vs 19.33≈	20.71 vs 20.02↓	
VGG16	21.15 vs 21.03↓	20.49 vs 20.67↑	22.16 vs 22.16≈	25.70 vs 23.57↓	26.22 vs 24.10↓	21.03 vs 21.03≈	22.38 vs 22.38≈	25.38 vs 23.71↓	26.15 vs 23.98↓
WideResNet	16.29 vs 16.25↓	15.70 vs 15.75↑	17.12 vs 17.12≈	19.43 vs 18.93↓	19.21 vs 18.62↓		17.10 vs 17.10≈	18.71 vs 18.48↓	19.20 vs 18.86↓
	Negative cLog-Likelihood on CIFAR100 dataset before and after data augmentation (100 samples)								
Model	cSGLD	Deep ensemble	FGE	K-FAC-L	Single model	SSE	SWAG	FFG VI	Dropout
ResNet110	0.630 vs 0.616↓	0.595 vs 0.591↓	0.661 vs 0.661≈	0.812 vs 0.766↓	0.848 vs 0.768↓	0.651 vs 0.651≈	0.684 vs 0.684≈	0.795 vs 0.749↓	
ResNet164	0.606 vs 0.595↓	0.574 vs 0.574≈	0.637 vs 0.637≈	0.772 vs 0.740↓	0.815 vs 0.743↓	0.629 vs 0.629≈	0.654 vs 0.654≈	0.760 vs 0.729↓	
VGG16	0.749 vs 0.741↓	0.745 vs 0.753↑	0.800 vs 0.800≈	1.050 vs 0.909↓	1.122 vs 0.936↓	0.751 vs 0.750↓	0.796 vs 0.796≈	0.975 vs 0.892↓	1.117 vs 0.942↓
WideResNet	0.583 vs 0.581↓	0.570 vs 0.574↑	0.596 vs 0.596≈	0.789 vs 0.747↓	0.796 vs 0.736↓		0.614 vs 0.614≈	0.678 vs 0.672↓	0.749 vs 0.721↓
	cBrier on CIFAR100 dataset before and after data augmentation (100 samples)								
Model	cSGLD	Deep ensemble	FGE	K-FAC-L	Single model	SSE	SWAG	FFG VI	Dropout
ResNet110	0.256 vs 0.251↓	0.241 vs 0.240↓	0.269 vs 0.269≈	0.311 vs 0.296↓	0.319 vs 0.296↓	0.258 vs 0.258≈	0.281 vs 0.281≈	0.307 vs 0.294↓	
ResNet164	0.244 vs 0.241↓	0.232 vs 0.233≈	0.260 vs 0.260≈	0.296 vs 0.286↓	0.305 vs 0.284↓	0.250 vs 0.250≈	0.269 vs 0.269≈	0.292 vs 0.282↓	
VGG16	0.293 vs 0.293↓	0.286 vs 0.290↑	0.307 vs 0.307≈	0.357 vs 0.327↓	0.371 vs 0.336↓	0.291 vs 0.291≈	0.307 vs 0.308≈	0.348 vs 0.328↓	0.369 vs 0.334↓
WideResNet	0.230 vs 0.230≈	0.222 vs 0.224↑	0.240 vs 0.240≈	0.281 vs 0.271↓	0.280 vs 0.267↓		0.240 vs 0.240≈	0.262 vs 0.260↓	0.274 vs 0.268↓
	cAURC on CIFAR100 dataset before and after data augmentation (100 samples)								
Model	cSGLD	Deep ensemble	FGE	K-FAC-L	Single model	SSE	SWAG	FFG VI	Dropout
ResNet110	0.0421 vs 0.0406↓	0.0382 vs 0.0378↓	0.0456 vs 0.0456≈	0.0607 vs 0.0557↓	0.0648 vs 0.0559↓	0.0426 vs 0.0426≈	0.0496 vs 0.0496≈	0.0593 vs 0.0550↓	
ResNet164	0.0388 vs 0.0379↓	0.0359 vs 0.0358↓	0.0429 vs 0.0429≈	0.0556 vs 0.0522↓	0.0600 vs 0.0523↓	0.0402 vs 0.0402↓	0.0455 vs 0.0456≈	0.0534 vs 0.0507↓	
VGG16	0.0528 vs 0.0524↓	0.0508 vs 0.0514↑	0.0570 vs 0.0570↑	0.0779 vs 0.0646↓	0.0855 vs 0.0682↓	0.0518 vs 0.0516↓	0.0572 vs 0.0572↑	0.0741 vs 0.0654↓	0.0857 vs 0.0676↓
WideResNet	0.0343 vs 0.0344↑	0.0324 vs 0.0330↑	0.0364 vs 0.0364≈	0.0498 vs 0.0471↓	0.0499 vs 0.0454↓		0.0366 vs 0.0366≈	0.0439 vs 0.0433↓	0.0476 vs 0.0456↓

Table 8: Results before and after data augmentation on CIFAR100.

Model (# samples)	Deep ensemble	FGE	K-FAC-L	Single model	SSE	FFG VI
ResNet50 (7 samples)	21.01 vs 20.66↓	23.59 vs 21.61↓	24.04 vs 21.97↓	23.81 vs 21.97↓	21.96 vs 21.29↓	23.76 vs 22.07↓
ResNet50 (40 samples)	20.64 vs 19.40↓	23.29 vs 20.75↓	23.80 vs 21.14↓	23.81 vs 21.08↓		23.69 vs 21.20↓
ResNet50 (50 samples)	20.63 vs 19.36↓		23.82 vs 21.04↓	23.81 vs 21.06↓		23.69 vs 21.11↓
Model (# samples)	Deep ensemble	FGE	K-FAC-L	Single model	SSE	FFG VI
ResNet50 (7 samples)	0.813 vs 0.817↑	0.918 vs 0.863↓	0.950 vs 0.877↓	0.938 vs 0.872↓	0.852 vs 0.843↓	0.925 vs 0.877↓
ResNet50 (40 samples)	0.789 vs 0.742↓	0.907 vs 0.798↓	0.937 vs 0.812↓	0.938 vs 0.808↓		0.920 vs 0.807↓
ResNet50 (50 samples)	0.788 vs 0.739↓		0.936 vs 0.808↓	0.938 vs 0.805↓		0.920 vs 0.804↓
Model (# samples)	Deep ensemble	FGE	K-FAC-L	Single model	SSE	FFG VI
ResNet50 (7 samples)	0.297 vs 0.296↓	0.327 vs 0.309↓	0.335 vs 0.313↓	0.331 vs 0.312↓	0.309 vs 0.304↓	0.329 vs 0.314↓
ResNet50 (40 samples)	0.292 vs 0.278↓	0.324 vs 0.294↓	0.331 vs 0.299↓	0.331 vs 0.298↓		0.328 vs 0.299↓
ResNet50 (50 samples)	0.291 vs 0.278↓		0.331 vs 0.299↓	0.331 vs 0.298↓		0.328 vs 0.298↓
Model (# samples)	Deep ensemble	FGE	K-FAC-L	Single model	SSE	FFG VI
ResNet50 (7 samples)	0.0582 vs 0.0603↑	0.0684 vs 0.0653↓	0.0714 vs 0.0666↓	0.0701 vs 0.0665↓	0.0619 vs 0.0630↑	0.0688 vs 0.0669↓
ResNet50 (40 samples)	0.0565 vs 0.0546↓	0.0673 vs 0.0600↓	0.0703 vs 0.0616↓	0.0701 vs 0.0613↓		0.0685 vs 0.0614↓
ResNet50 (50 samples)	0.0564 vs 0.0544↓		0.0702 vs 0.0614↓	0.0701 vs 0.0611↓		0.0684 vs 0.0612↓

Table 9: Results before and after data augmentation on ImageNet.

REFERENCES

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

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, pp. 8789–8798, 2018.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pp. 2575–2583, 2015.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017.

Wesley Maddox, Timur Garipov, Pavel Izmailov, Dmitry Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. *arXiv preprint arXiv:1902.02476*, 2019.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417, 2015.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2498–2507. JMLR. org, 2017.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. 2018.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. How to train deep variational autoencoders and probabilistic ladder networks. In *33rd International Conference on Machine Learning (ICML 2016)*, 2016.

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.

Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*, 2017.

Sergey Zagoruyko. 92.45 on cifar-10 in torch, 2015. URL `http://torch.ch/blog/2015/07/30/cifar.html`.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for bayesian deep learning. *arXiv preprint arXiv:1902.03932*, 2019.