Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning

Arsenii Ashukha[*1,3] **Alexander Lyzhov**[*1,2,4] **Dmitry Molchanov**[*1,3] **Dmitry Vetrov**[1,3]
[1]Samsung AI Center Moscow
[2]National Research University Higher School of Economics
[3]Samsung-HSE Laboratory, National Research University Higher School of Economics
[4]Skolkovo Institute of Science and Technology, [*] equal contribution

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 this study, 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

There are many faces to uncertainty estimation, and there are many settings and metrics to measure all desirable properties of a well-uncertain model. In the *out-of-domain setting* uncertainty of the model is measured on data that mismatches the training data distribution. The model is expected to be resistant to corruptions and to be more uncertain on out-of-domain data as compared to the in-domain data. This setting was explored in a recent study [21]. On the contrary, in the *in-domain setting*, uncertainty of the model is measured on data from the training data distribution. In this case, a model is expected to provide correct probability estimates that can be measured by metrics like log-likelihood, Brier score, calibration metrics, and performance of downstream problems, e.g., misclassification detection.

Ensembles of deep neural networks have become a de-facto standard for uncertainty estimation and improving the quality of deep learning models [14]. 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 stochastic computation graph paradigm introduce noise over the 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 [26, 5], variational inference [2, 12, 15], batch normalization [11, 1], Laplace approximation [24] and many more. Snapshot-based methods aim to obtain sets of weights for deep learning models and then average the predictions across these weights. The weights can be trained independently (e.g., deep ensembles [14]), collected on different stages of a training trajectory (e.g., snapshot ensembles [10] and fast geometric ensembles [6]), or obtained from a sampling process (e.g., MCMC-based methods [28, 30]).

4th workshop on Bayesian Deep Learning (NeurIPS 2019), Vancouver, Canada.
Correspondence to {ars.ashuha, alex.grig.lyzhov, dmolch111}@gmail.com

In this paper we focus on two problems in the field of in-domain uncertainty estimation. One is the multitude of different metrics and experimental setups, some of which we show to be flawed. The other one is a lack of consistent comparisons of different methods on the larger scale. We find that deep ensembles outperform all other ensembling techniques using much less computational resources at test-time. While methods that are specifically designed to traverse different "optima" of the loss function (e.g. snapshot ensembles and cyclical SGLD) can come close to matching the performance of deep ensembles, methods that are bound to a single "optima" are found to fall far behind. Finally, it turns out that test-time data augmentation is a surprisingly strong baseline that outperforms many conventional uncertainty estimation methods like variational inference, Laplace approximation and dropout.

2 Pitfalls of measuring the quality of in-domain uncertainty estimation

There are many definitions of uncertainty and there is no single metric that can measure all desirable properties of a well-uncertain model. To this end, the community has used different metrics that aim to measure the quality of uncertainty estimation e.g., Brier score [3], log-likelihood [23], different calibration metrics [7, 20], performance of misclassification detection [17], and threshold–accuracy curves [14].

We consider a classification problem with a dataset that consists of N train pairs and n test pairs $(x_i, y_i^*) \sim p(x, y)$ where $y_i^* \in \{1, \dots, C\}$. A probabilistic classifier is a function mapping each object x_i into a set of class probabilities $\hat{p}(y = c \,|\, x_i), c \in \{1, \dots, C\}$. $\max_c \hat{p}(y = c \,|\, x_i)$ is called classifier confidence on object x_i. $\mathbb{I}[\cdot]$ denotes the indicator function.

2.1 Log-likelihood

The average log-likelihood $LL = \frac{1}{n} \sum_{i=1}^{n} \log \hat{p}(y = y_i^* \,|\, x_i)$ is a popular metric for measuring the quality of in-domain uncertainty of deep learning models [14, 16]. It penalizes high probability scores assigned to incorrect labels and low probability scores assigned to the correct labels y_i^*.

The softmax function is used in various kinds of classification models. It transforms a vector of logits $z(x)$ into the predictive distribution $\hat{p}(y = c \,|\, x) = \frac{\exp(z_c(x)/T)}{\sum_j \exp(z_j(x)/T)}$. The temperature parameter T significantly affects the log-likelihood which can either improve or deteriorate with temperature change. The temperature T^* that maximizes the log-likelihood on validation data allows to significantly improve it on test data too [7]. For now it is not clear whether it is possible to find the optimal temperature without relying on hold-out data.

While ensembling techniques do seem to have better temperature than single models, the default choice is still suboptimal. Comparison of the average log-likelihood with suboptimal temperatures can significantly change the ranking of different methods.

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

While this applies to most ensembling techniques (see Appendix A), this effect is most noticeable on experiments with data augmentation on ImageNet (see Figure 1 and Table 1 for more details).

2.2 Brier score

Brier score $BS = \frac{1}{n} \frac{1}{C} \sum_{i=1}^{n} \sum_{c=1}^{C} (\mathbb{I}[y_i^* = c] - \hat{p}(y = c \,|\, x_i))^2$ [3] has been known for a long time as a metric for verification of predicted probabilities. 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.

2.3 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

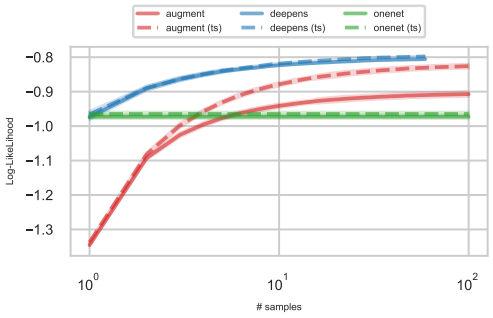


Figure 1: The average log-likelihood of different ensembles before and after temperature scaling (TS) on ResNet50 on ImageNet dataset. Before TS deep ensembles outperform data augmentation, while after TS the difference appears to be significantly lower.

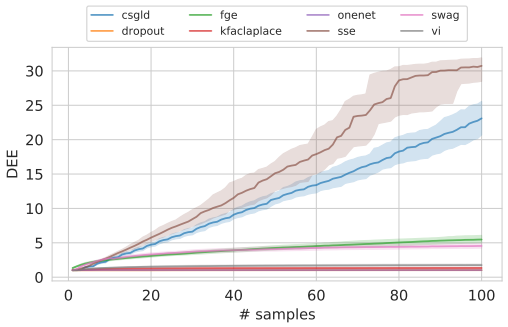


Figure 2: Solid lines: Deep Ensemble Equivalent (DEE, Section 3.1) for different models, averaged across different datasets and architectures. Area between average DEE_{lower} and DEE_{upper} is shaded.

binary classification such as AUC-ROC and AUC-PR [17, 4, 18]. These papers use an uncertainty criterion like confidence or predictive entropy $\mathcal{H}[\hat{p}(y \mid x_i)]$ as a prediction score.

These metrics, however, cannot be used to compare misclassification performance across different models. Correct and incorrect predictions are specific for every model, therefore, every model induces a different classification problem. The induced problems can differ from each other significantly since different models produce different confidences and misclassify different objects.

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

Note that while comparing AUCs is incorrect in this setting, it is correct to compare these metrics in many out-of-domain data detection problems. In this case, both the objects and the targets of the induced binary classification problems remain fixed for all models. However, this condition still breaks in the problem of detection of adversarial attacks.

2.4 Classification with rejection

Rejection curves are another way to measure the performance of misclassification detection. To construct the curves, the accuracy is measured only over objects that are assigned a confidence above a certain threshold [14].

The main problem with threshold-accuracy curves is that they rely too much on calibration and the actual values of confidence. Models with different temperatures would have different amounts 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 amount of rejected objects. The corresponding curves are then less sensitive to the temperature scaling and allow to compare the misclassification detection ability in a more meaningful way.

2.5 Calibration metrics

Informally speaking, probabilistic classifier is calibrated if any predicted class probability is equal to the true class probability according to the underlying data distribution (follow [27] for formal definitions).

Expected Calibration Error (ECE) [19] is a score that estimates model miscalibration. Typically, there is no access to expected accuracy of the classifier at the exact confidence level, so binning is needed to approximate expected accuracy with average accuracy over the bin. When computing ECE, predictions are binned uniformly according to their confidence level. Assuming B_m denotes the m-th bin, M is overall number of bins, accuracy and confidence are averaged over predictions of

each bin:

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

where, denoting $\hat{p}_{i,c} = \hat{p}(y = c \,|\, x_i)$ for brevity, $\mathrm{acc}(B) = |B|^{-1} \sum_{i \in B} \mathbb{I}[\arg\max_c \hat{p}_{i,c} = y_i^*]$ and $\mathrm{conf}(B) = |B|^{-1} \sum_{i \in B} \hat{p}_{i,y_i^*}$. A recent line of work on measuring calibration in deep learning [20, 27] outlines several problems with the ECE score. The bias-variance trade-off of estimation using ECE is very suboptimal. ECE-like scores cannot be optimized directly since they are minimized by a model with constant uniform predictions. ECE only estimates miscalibration in terms of confidence 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 miscalibration estimates incompatible.

A modification of ECE called Thresholded Adaptive Calibration Error (TACE) was proposed as a step towards solving some of these problems [20]. As opposed to ECE, TACE estimates miscalibration of probabilties across all classes in a prediction. When computing TACE, binning is carried out over predicted probabilities which are larger than a predefined threshold, separately for each class (not just the top-1 predicted class as in ECE). Bin widths are varied so that each bin has the same number of objects. Assuming that $B_m^{adaptive}$ denotes the m-th bin and M is the overall number of bins:

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

where $\mathrm{objs}(B, c) = |B|^{-1} \sum_{i \in B} \mathbb{I}[y_i^* = c]$ and $\mathrm{conf}(B, c) = |B|^{-1} \sum_{i \in B} \hat{p}_{i,c}$. 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 criterion for comparing different models. This is because TACE is sensitive to its two parameters, the number of bins and the threshold, and does not provide a consistent ranking of different models (see Figure 7 in Appendix C for details). Biases of TACE on different models may not be equal for different models just like in the case of ECE. Moreover, they are highly dependent on the number of objects, further complicating its usage as a criterion for comparison of different models or ensembling techniques.

2.6 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). Some authors divide the public training set into a smaller training set and validation set [7, 16]. Others divide the public test set in half, use one half for validation, and report test metrics for the other half [7, 20]. 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 provides an unbiased estimate of metrics such as log-likelihood and Brier score but introduces more variance.

In order to reduce the variance, we perform "test-time cross-validation": we compute metrics on each half of the test set using temperature optimized on another half of the test set. We repeat this procedure five times and average the resulting metrics across different splits to reduce the variance.

3 A broad comparison of ensembling techniques

In this paper, we consider the following ensembling techniques: deep ensembles [14], snapshot ensembles (SSE [10]), fast geometric ensembling (FGE [6]), SWA-Gaussian (SWAG [16]), cyclical SGLD (cSGLD [30]), variational inference [2], dropout [26] and test-time data augmentation. These techniques were chosen judging on predictive performance and diversity of approaches.

All these techniques can be summarized as distributions $q_m(w)$ over computation graphs, parameterized by weights w, where m stands for the name of the technique. During testing, one can average the predictions of different models $w_k \sim q_m(w)$ to approximate the predictive distribution $\hat{p}(y_i \,|\, x_i)$:

$$\hat{p}(y_i \,|\, x_i) \approx \int p(y_i \,|\, x_i, w) q_m(w) dw \simeq \frac{1}{C} \sum_{c=1}^{C} p(y_i \,|\, x_i, w_c) \tag{3}$$

Method	Error (top-1)	NLL	NLL (ts)	Brier	Brier (ts)
One network	24.52 ± 0.15	0.973 ± 0.006	0.965 ± 0.006	0.341 ± 0.002	0.340 ± 0.002
Augmentation (Aug.)	21.72 ± 0.10	0.911 ± 0.007	0.832 ± 0.006	0.329 ± 0.002	0.307 ± 0.001
Deep ensembles (DE)	20.82 ± 0.02	$\mathbf{0.807} \pm 0.000$	0.799 ± 0.000	$\mathbf{0.298} \pm 0.000$	0.295 ± 0.000
DE + Aug	$\mathbf{19.61} \pm 0.03$	0.863 ± 0.000	$\mathbf{0.749} \pm 0.001$	0.315 ± 0.000	$\mathbf{0.281} \pm 0.000$

Table 1: Results on combination of deep ensembles and augmentation (50 samples each) on ResNet50 and ImageNet dataset. Test-time data augmentation improves both accuracy and negative log-likelihood, however it breaks a nearly perfect temperature naturally provided by deep ensembles. This effect is shown by significant improvement of negative log-likelihood after temperature scaling. Comparison of log-likelihoods without scaling the softmax temperature first may result in incorrect ranking of ensembling methods.

3.1 Deep Ensemble Equivalent

Instead of comparing the values of different uncertainty estimation metrics directly, we ask the following question aiming to introduce perspective and interpretability in our method comparison: what number of independently trained networks combined yields the same performance as an application of a particular ensembling method?

Following insights from the previous sections, we use log-likelihood at the optimal temperature, or the calibrated log-likelihood (CLL) as the main measure of performance of the ensemble. We define the Deep Ensemble Equivalent (DEE) and its upper and lower bounds as follows:

$$\mathrm{DEE}(l) = \min k \;\text{ s.t. }\; \mathrm{CLL}^{\mathrm{DE}}_{\mathrm{mean}}(k) \geq l, \tag{4}$$

$$\mathrm{DEE}_{upper/lower}(l) = \min k \;\text{ s.t. }\; \mathrm{CLL}^{\mathrm{DE}}_{\mathrm{mean}}(k) \mp \mathrm{CLL}^{\mathrm{DE}}_{\mathrm{std}}(k) \geq l, \tag{5}$$

where $\mathrm{CLL}^{\mathrm{DE}}_{\mathrm{mean}}(k)$ and $\mathrm{CLL}^{\mathrm{DE}}_{\mathrm{std}}(k)$ are the mean and the standard deviation of the calibrated log-likelihood of a deep ensemble of k networks. We compute $\mathrm{CLL}^{\mathrm{DE}}_{\mathrm{mean}}(k)$ and $\mathrm{CLL}^{\mathrm{DE}}_{\mathrm{std}}(k)$ for natural numbers $k = 1, \ldots, 100$ and use linear interpolation to define them for real-valued k. In the following plots we report DEE for different number of samples from different methods, and shade the area between the respective lower and upper bounds DEE_{lower} and DEE_{upper}.

3.2 Experiments

We considered three popular deep architectures (VGG16 [25, 29], PreResNet110 and PreResNet164 [8]) on CIFAR-10/100 datasets [13]. We used public PyTorch [22] implementations of ensembling techniques that were available, and reimplemented the rest of the techniques. All the reported results have been reproduced by us; we do not report the results published in the previous papers. Technical details on training, hyperparameters and implementation can be found in Appendix B. Source code for all mentioned methods as well as scripts for training and ensembling will be available at `https://github.com/bayesgroup/pytorch-ensembles`.

As one can see on Figure 2, the 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. It means that these methods can efficiently explore different modes of the loss landscape and do not saturate unlike other methods that are bound to a single mode. More verbose results are presented in Figure 5 in Appendix B.

Other more "local" methods like FGE and SWAG perform worse than SSE and cSGLD, but are still able to obtain more diversity from the local snapshots than "single-snapshot" models like dropout, Laplace approximation and variational inference.

3.3 Data augmentation

Interestingly, simple test-time data augmentation performs surprisingly well and outperforms dropout, Laplace approximation and variational inference (see preliminary results in Appendix A and Figure 6). It is, however, orthogonal to these techniques, and can be used to improve them. In Table 1 we show results of combination of deep ensembles and augmentation on ImageNet dataset.

Augmentation improves accuracy of deep ensembles by a large margin, however, it breaks the nearly optimal temperature of deep ensembles and requires temperature scaling (see Section 2.1). Other ensemble methods from the list above are coming soon.

The performance is better than a state-of-the-art Bayesian ResNet50 on ImageNet that achieved 22.5 top1-error and 0.883 negative log-likelihood [9]. These numbers, however, are not directly comparable due to slight variations in architectures and the absence of temperature scaling.

4 Discussion

We have explored the field of in-domain uncertainty estimation and performed a large-scale evaluation of modern ensembling techniques. Our main conclusions 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 can focus on calibration and might not provide a fair ranking.
- Many common metrics for measuring in-domain uncertainty are either unreliable (ECE and analogues) or cannot be used to compare different methods (AUROC, AUPR 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 or hundreds of samples for test-time averaging, yet are essentially equivalent to a handful of independently trained models. 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 explore the loss landscape. Would more elaborate posterior approximations shorten this gap or is there not enough diversity in a vicinity of a single mode?
- 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.

A large number of unreliable metrics inhibits fair comparison of different methods. We urge the community to take time and aim for more reliable benchmarks in the numerous setups of uncertainty estimation.

Acknowledgments

Dmitry Vetrov was supported by the Russian Science Foundation grant no. 19-71-30020. The authors thank NRU HSE for providing computational resources.

References

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

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

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

[4] 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.

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

[6] 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*, pages 8789–8798, 2018.

[7] 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*, pages 1321–1330. JMLR. org, 2017.

[8] 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*, pages 770–778, 2016.

[9] Jonathan Heek and Nal Kalchbrenner. Bayesian inference for large scale image classification. *arXiv preprint arXiv:1908.03491*, 2019.

[10] 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.

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

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

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

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

[15] 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*, pages 2218–2227. JMLR. org, 2017.

[16] 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.

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

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

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

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

[21] 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.

[22] 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.

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

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

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

[26] 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.

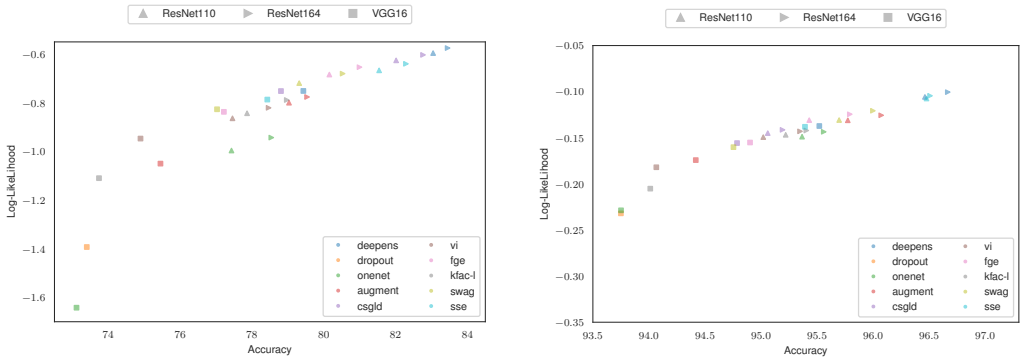
[27] Juozas Vaicenavicius, David Widmann, Carl Andersson, Fredrik Lindsten, Jacob Roll, and Thomas B Schön. Evaluating model calibration in classification. *arXiv preprint arXiv:1902.06977*, 2019.

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

[29] Sergey Zagoruyko. 92.45 on cifar-10 in torch, 2015.

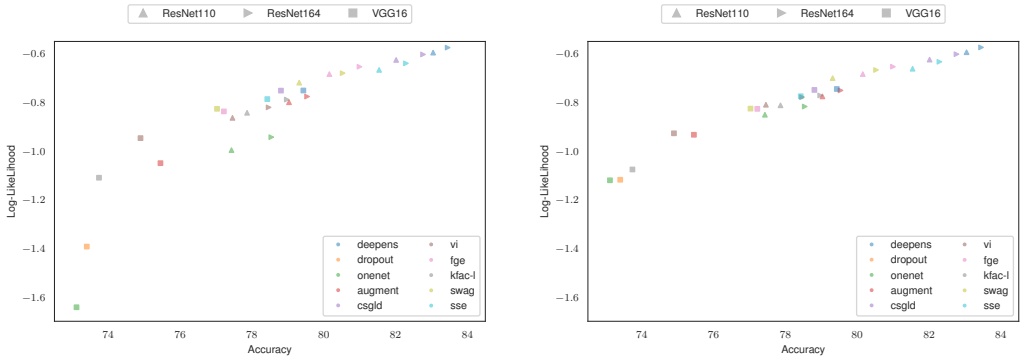
[30] Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for bayesian deep learning. *arXiv preprint arXiv:1902.03932*, 2019.

A Accuracy/LL plots with and without temperature scaling



(a) Before temperature scaling, $\rho = 0.856$ (b) After temperature scaling, $\rho = 0.960$

Figure 3: The average log-likelihood and accuracy on a test dataset for different trained networks of CIFAR-10 dataset [13] before (a) and after (b) temperature scaling. ρ denotes the correlation between the accuracy and log-likelihood.



(a) Before temperature scaling, $\rho = 0.891$ (b) After temperature scaling, $\rho = 0.977$

Figure 4: The average log-likelihood and accuracy on a test dataset for different trained networks of CIFAR-100 dataset [13] before (a) and after (b) temperature scaling. ρ denotes the correlation between the accuracy and log-likelihood.

B Experimental details

Our implementations of deep ensembles, SWAG, FGE and KFAC-Laplace is heavily based on the original SWA / SWAG repositories `https://github.com/timgaripov/swa` and `https://github.com/wjmaddox/swa_gaussian`. We reuse original hyperparameters for CIFAR-10 and CIFAR-100. We trained members of deep ensembles and other methods using the hyperparameters listed in Table 2 on CIFAR datasest. We trained ResNet50 on ImageNet for 130 epochs with standard hyperparameters (`https://github.com/pytorch/examples/tree/master/imagenet`).

Model	Data	optimizer	init_lr	total_epochs	wd	Pretraining
VGG	CIFAR-10/100	SGD	0.05	400	5e-4	no
PreResNet110	CIFAR-10/100	SGD	0.1	300	3e-4	no
PreResNet164	CIFAR-10/100	SGD	0.1	300	3e-4	no
VI-*	CIFAR-10/100	weights: SGD (lr=1e-4), log_vars: Adam (lr = 0.1)	-	100	*	yes

Table 2: Hyperparameters of various models trained on CIFARs

For KFAC-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 [24]. Following the original paper [24], we find the optimal noise scale for KFAC-Laplace on a hold-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 resulting optimum is different 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 KFAC-Laplace with data augmentation is lower.

Learning rate for base models is scheduled as follows:

$$\texttt{lr}(\text{epoch; init\_lr, total\_epochs}) = \begin{cases} \text{init\_lr}, & \text{epoch} \leq 0.5 \cdot \text{total\_epochs} \\ 1.0 - 0.99 * (\frac{\text{epoch}}{\text{total\_epochs}} - 0.5)/0.4, & 0.5 \cdot \text{total\_epochs} \leq \text{epoch} < 0.9 \cdot \text{total\_epochs} \\ 0.01, & \text{otherwise} \end{cases} \tag{6}$$

All hyperparameters from original papers on SSE and cSGLD are reused except for the number of noise epochs per cycle of learning rate for cyclical SGLD. It was set to the empirically optimal choice of 3 noise epochs per cycle consistently across all datasets. Cyclical SGHMC which reportedly has marginally better performance compared with cyclical SGLD could not be reproduced using any value of momentum term. Because of this, we only include cyclical SGLD in our work.

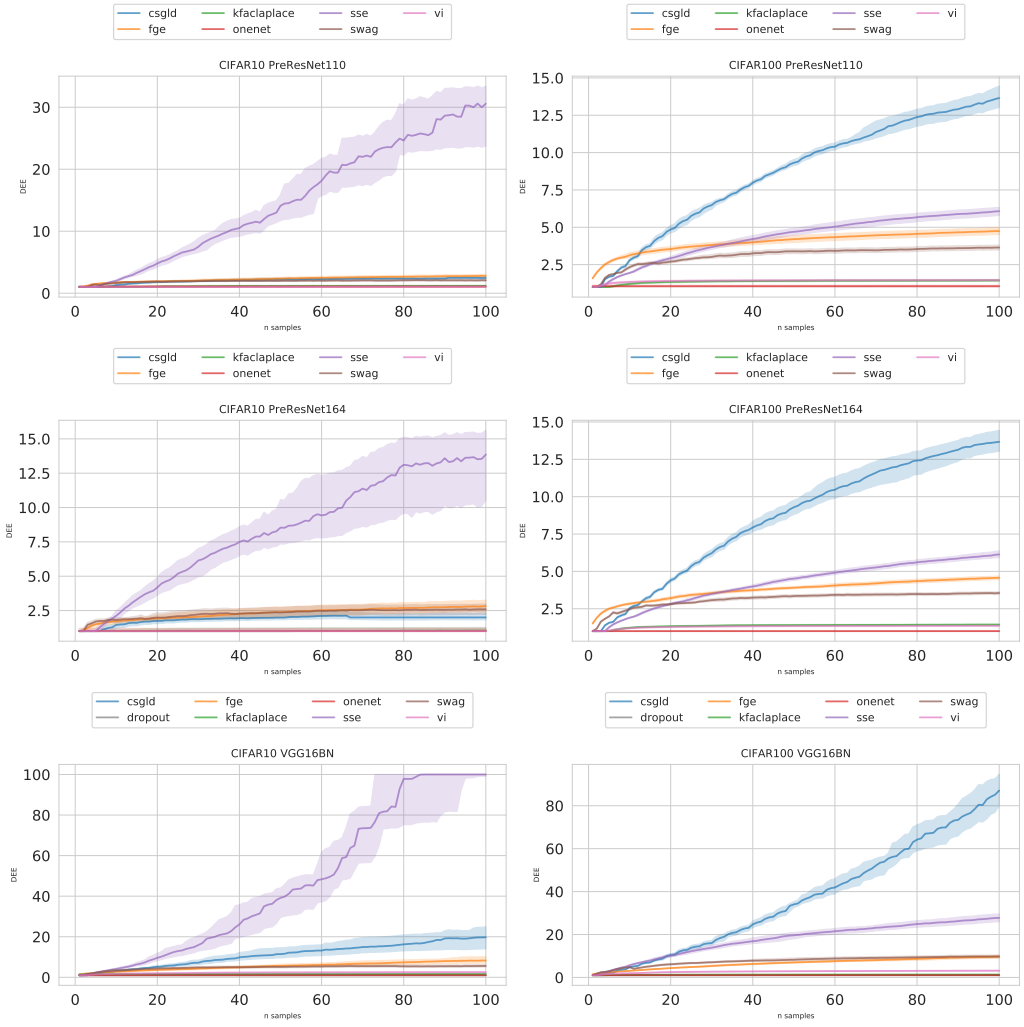


Figure 5: Solid lines: mean DEE for different methods, architectures and datasets. Area between DEE_{lower} and DEE_{upper} is shaded.

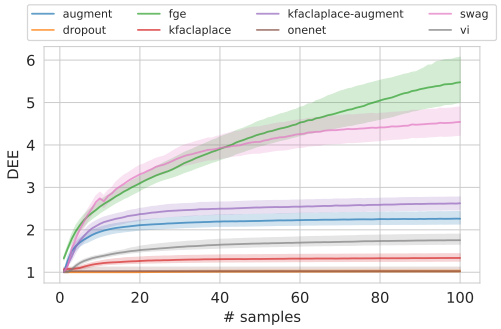


Figure 6: Solid lines: DEE for different models, averaged across different datasets and architectures. Area between average DEE_{lower} and DEE_{upper} is shaded. Test-time data augmentation alone outperforms methods line KFAC-Laplace and variational inference, and improves the performance of KFAC-Laplace.

	Optimal noise scale			
Architecture	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

Table 3: Optimal noise scale for KFAC-Laplace for different datasets and architectures.

C Calibration error

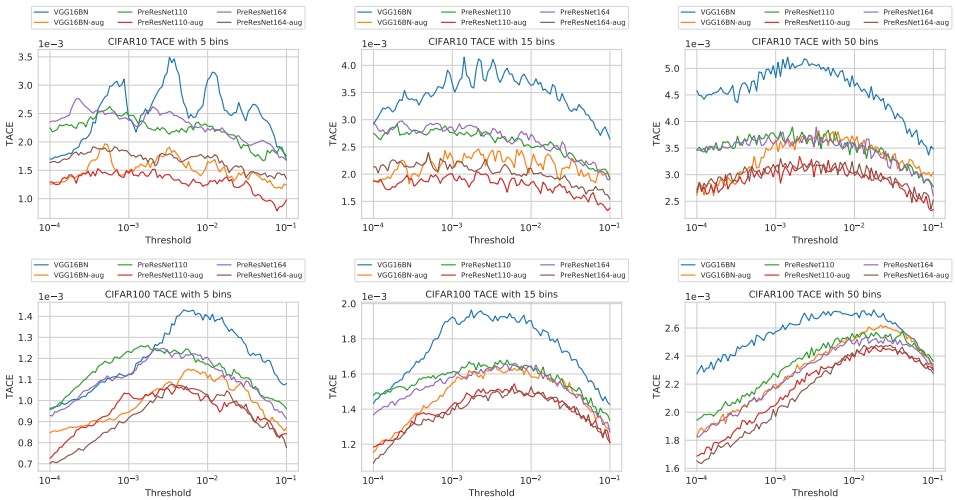


Figure 7: Top: KFAC-Laplace models on CIFAR10. Bottom: KFAC-Laplace models on CIFAR100. By choosing different number of bins and different thresholds, one can obtain substantially different rankings across the models.