# NeurIPS Reproducibility Challenge: "A Simple Baseline for Bayesian Uncertainty in Deep Learning"

**Victor Löfgren**
KTH Royal Institute of Technology
vlov@kth.se

**Josef Malmström**
KTH Royal Institute of Technology
josefmal@kth.se

## 1 Introduction

As deep learning methods are being applied in increasingly critical settings (e.g. self-driving cars and medical diagnosis) the importance of being able to accurately estimate the uncertainty of a model is undeniable. A variety of methods for uncertainty estimation in deep learning have been suggested, varying in complexity from simple post-processing of predictions, to more intricate methods that can provide fully Bayesian inference.

In this work we reproduce the method proposed in [1] (henceforth referred to as"the SWAG paper" or just "the paper") which is suggested as a simple baseline for Bayesian uncertainty in deep learning. The proposed method provides an intuitive way of getting uncertainty estimates, building on the work of [2], by estimating a distribution over model weights using gradient descent iterates from the training procedure.

Our report is structured as follows. In Section 2, we provide the necessary background and briefly summarize and analyze the method suggested in the paper. In Section 3, we describe our experiments in reproducing the method, and analyze its reproducibility. We describe any additional experiments we did to evaluate the method in Section 4. In Sections 5, 6, and 7 respectively, we present our results, analyze them, and provide our conclusions.

The code we used for our experiments is available at https://github.com/DD2412-Final-Projects/swag-reproduced.

An error in our implementation meant we were not initially able to fully follow the method presented by the authors. In this revised version of the report, we have fixed this error and are now able to use an identical configuration to that of the paper. As our previous experiments are still deemed valid however, we leave the core of the report untouched and present our new found results in Appendix A.

## 2 Background

### 2.1 Bayesian uncertainty

In typical Bayesian uncertainty in deep learning, a distribution over network weights is estimated. This in turn allows for estimation of uncertainties in the predictions. A variety of different methods for estimating a distribution over model parameters have been suggested in the field, including Markov chain Monte Carlo (MCMC), variational inference, Dropout variational inference, and Laplace approximations.

A related problem is the issue of miscalibration. If a model is able to provide accurate estimates of the confidence associated with its predictions, we say the network is well-calibrated. If the confidence output by the model does not accurately represent the ground truth correctness likelihood we say the model is miscalibrated. There exists a variety of methods for improving the calibration of a

model. These include temperature scaling, the addition of an adversarial loss term during training, or ensembling the predictions of multiple models. A commonly used metric for miscalibration is the expected calibration error (ECE), defined as

$$\mathop{\mathbb{E}}_{\hat{P}}\left[\left|\mathbb{P}(\hat{Y} = Y|\hat{P} = p) - p\right|\right]. \tag{1}$$

where $\hat{Y}$ is the set of predicted labels and $Y$ is the set of ground-truth labels. This quantity can be estimated by splitting the data samples into bins based on their confidence, and computing a discretized approximation of the expected value, as described in [3].

### 2.2 SWA and SWAG

Stochastic Weight Averaging (SWA) is an algorithm proposed in [2], that computes an average of network weights across training iterations. At prediction time, the model is then instantiated with the average weights. It is shown that SWA improves generalization of the model while adding very little additional computational overhead during training. As such, it is not a Bayesian uncertainty estimator like MCMC nor is it a post-processing calibrator like temperature scaling, but rather a regularizing training method that estimates an ensemble.

In the SWAG paper, the authors build upon SWA to propose a modified version of the algorithm which allows a distribution over model weights to be estimated. Thus SWAG (or SWA-Gaussian) is a fully Bayesian uncertainty estimator. The distribution over weights is assumed to be Gaussian, $\theta \sim N(\bar{\theta}, \Sigma)$, and therefore only the mean, $\bar{\theta}$, and the covariance, $\Sigma$, need to be estimated. In a simple version of SWAG, referred to as SWAG-Diag, the covariance is assumed to be diagonal, $\Sigma = \Sigma_{\text{diag}}$. In the full version of SWAG the covariance is assumed to have a low-rank plus diagonal structure, $\Sigma = \frac{1}{2}\Sigma_{\text{diag}} + \frac{1}{2(K-1)}\hat{D}\hat{D}^T$, where $K$ is the rank of matrix $\hat{D}$. The mean $\bar{\theta}$, the diagonal part of the covariance $\Sigma_{\text{diag}}$, and the low-rank part of the covariance $\hat{D}$, is computed in the SWAG algorithm, outlined in Algorithm 1.

**Require:** $\theta_0$: initial weights; $\eta$: learning rate; $T$: number of epochs; $c$: moment update frequency; $K$: maximum number of columns in deviation matrix; $S$: number of samples in Bayesian averaging at test time.

**Train SWAG:**
$\bar{\theta} \leftarrow \theta_0$, $\bar{\theta^2} \leftarrow \theta_0^2$
**for** $i = 1, 2, ..., T$ **do**
  $\theta_i \leftarrow \theta_i - \eta\nabla_{\theta_{i-1}}\mathcal{L}_\theta(\theta_{i-1})$
  **if** *MOD*$(i, c) == 0$ **then**
    $n \leftarrow i/c$
    $\bar{\theta} = \frac{n\bar{\theta}+\theta_i}{n+1}$, $\bar{\theta^2} = \frac{n\bar{\theta^2}+\theta_i^2}{n+1}$
    **if** *NUM_COLS*$(\hat{D}) = K$ **then**
      REMOVE_COL$(\hat{D}[:, 1])$
    **end**
    APPEND_COL$(\hat{D}, \theta_i - \bar{\theta})$
  **end**
**end**
**return** $\bar{\theta}$, $\Sigma_{\text{diag}} = \text{diag}(\bar{\theta^2} - \bar{\theta}^2)$, $\hat{D}$

**Test time** - Do Bayesian model averaging
**for** $i \leftarrow 1, ..., S$ **do**
  Draw $\theta_i \sim N(\bar{\theta}, \Sigma)$,
  where $\Sigma = \Sigma_{\text{diag}}$ for SWAG-Diag, and $\Sigma = \frac{1}{2}\Sigma_{\text{diag}} + \frac{1}{2(K-1)}\hat{D}\hat{D}^T$ for SWAG.
  $p(y^*|\text{Data}) = p(y^*|\text{Data}) + \frac{1}{S}p(y^*; \theta_i)$
**end**
**return** $p(y^*|\text{Data})$

**Algorithm 1:** SWAG algorithm

At test time, network weights are iteratively sampled from the learned weight distribution $S$ times, and the set of predicted probability distributions are simply averaged to produce the calibrated predictions. This procedure is also outlined in Algorithm 1. In the case of SWAG-Diag, the samples from $N(\bar{\theta}, \Sigma)$ are obtained through the identity

$$\theta = \bar{\theta} + \Sigma_{\text{diag}}^{\frac{1}{2}} z_1, \quad z_1 \sim N(0, I_d). \tag{2}$$

For SWAG the samples are obtained through the identity

$$\theta = \bar{\theta} + \frac{1}{\sqrt{2}} \Sigma_{\text{diag}}^{\frac{1}{2}} z_1 + \frac{1}{\sqrt{2(K-1)}} \hat{D} z_2, \quad z_1 \sim N(0, I_d), z_2 \sim N(0, I_K). \tag{3}$$

## 3 Reproducibility

In this section we describe our experiments in attempting to reproduce the methods in the paper. We start by describing, on a high level, our implementation as compared to the original from the paper. We then cover in detail the scope of our conducted experiments, and any details surrounding their execution.

In addition to simply reproducing the results and conclusions of the paper by copying the baselines, architectures, methods and evaluation metrics, we have implemented them in a different framework. The authors conducted their experiments in PyTorch whereas we do ours in TensorFlow 1.14. For this report we have limited our focus to the results produced with the VGG-16 [4] architecture. We base our implementation on the implementation available at https://www.cs.toronto.edu/ frossard/vgg16/vgg16.py. Since this implementation is not related to the works of the paper, a few changes were necessary to match the PyTorch implementation provided by the authors (e.g. the size of the fully connected layers and weight initialization). Although the PyTorch implementation includes an option for batch normalization, there are no mentions of this in the paper and therefore we exclude it from our experiments.

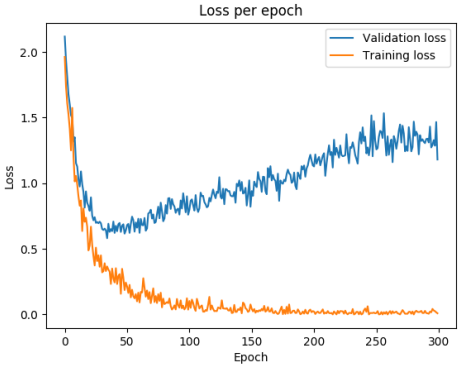
Further we decided to only use the CIFAR-10, CIFAR-100 and STL-10 datasets because our resources were too limited in both time and computational power to handle ImgageNet. With the architecture in place, we implemented the methods SWA, SWAG-Diag and SWAG from scratch with only the paper to guide us. We also implemented regular SGD as a baseline.

Since it is of importance that improvements seen by the implemented methods is not due to simply improving SGD, it is critical that SGD has reached convergence on its own. We were unable to achieve the level of performance of the SGD with the provided hyper-parameters, usage of momentum and weight decay. In our attempts we faced strange behaviors during training which we assume are due to implementation errors or poorly translated hyper-parameters and settings between programming frameworks. At some point both loss and accuracy changed trends, either drastically (as in Figure 2) or slowly. Either way, further training was unable to recover or change the trend likely due to the now very small weights [1]. We therefore decided to conduct our experiments with a vanilla SGD and make use only of the hyper-parameters and learning rate schedule provided by the authors. With convergence of the vanilla SGD we consider comparisons to remain fair to the proposed methods and perhaps even add to their credibility.

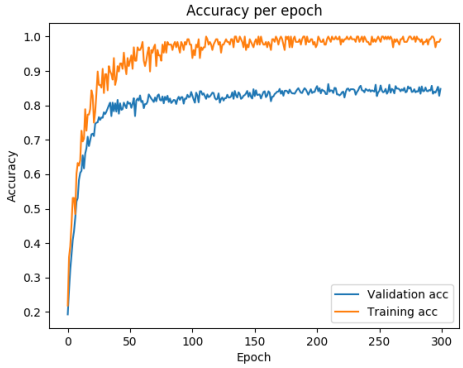
### 3.1 The experiments

For the two main studied datasets, CIFAR-10 and CIFAR-100, we conduct experiments as close to the ones presented in the paper as possible. As previously described, we find that, despite mimicking the choice of hyper-parameters from the paper to the best of our ability, we do not achieve convergence of the same quality with the authors' configuration. We instead opt for a plain SGD setup without momentum, and with 5 % Dropout as the only regularization. For the learning rate we adopt the same decaying schedule as in the paper: a constant learning rate of $5 \times 10^{-2}$ up to 50 % of the total number of epochs, and then a linear decay down to a learning rate of $5 \times 10^{-4}$ until 90 % of the total number of epochs, where once again the learning rate is kept constant until the end of training. We use a

[1] We have now identified the cause of this behavior, and are able to train the models with the authors' configuration. We present updated results using their setup in Appendix A. Note however, that our previous experiments and results are still deemed valid.

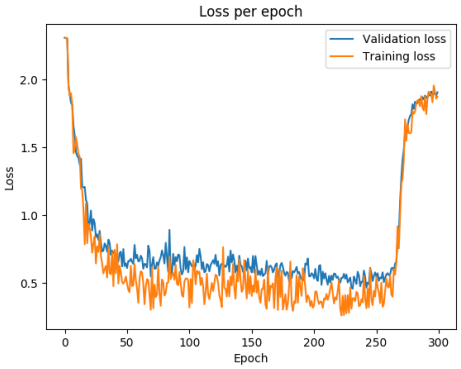



(a) Loss during training.

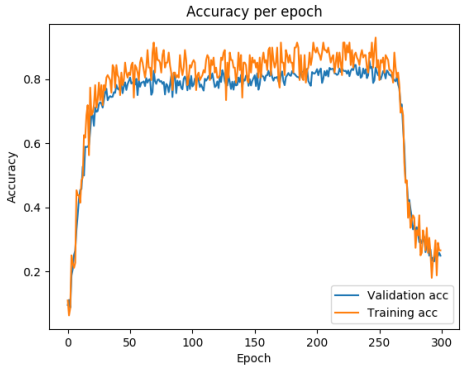



(b) Accuracy during training.

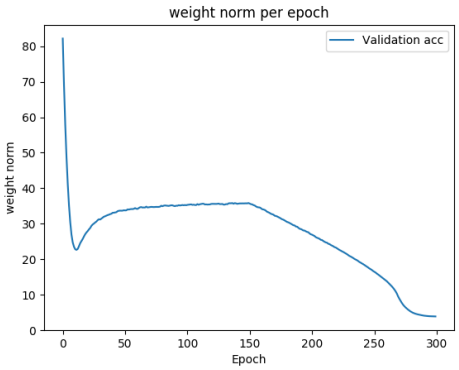
Figure 1: Loss and accuracy on the training and validation set respectively, during one of the SGD training runs.




(a) Loss evolution during training.




(b) Accuracy evolution during training.




(c) The weight norm evolution during training.

Figure 2: **Top:** Loss and accuracy on the training and validation set respectively, during SGD training with the full configuration described in the paper. **Bottom:** The evolution of the weight norm during the same training.

batch size of 128. Images input to the network during training are normalized so that pixel values are in the interval $[0, 1]$. We also use the same data augmentation discovered in the authors' code, but not described in the paper: images are zero-padded and cropped randomly, as well as flipped horizontally with 50 % probability. Review of the authors' code also revealed other implementational details that were not described in the paper. This includes hyperparameters connected to Dropout and weight decay, as well as additional input normalization done in VGG-16.

For each of the methods (i.e. SGD, SWA, SWAG-Diag and SWAG) we train our models for 300 epochs, and start collecting samples for SWA/SWAG-Diag/SWAG after epoch 160. We use the above learning rate schedule in the case of SGD, and adopt a constant learning rate of 0.01 for SWA, SWAG-Diag, and SWAG, as prescribed in the paper. The training configuration is otherwise identical for all methods. Figure 1 shows loss and accuracy on the training and validation sets, during one of the SGD training runs. The equivalent plot for other training runs (including SWA/SWAG-Diag/SWAG runs) looked essentially identical. We note that despite our lack of regularization, apart from a 5 % Dropout rate, the model does not appear to be overfitting since the validation accuracy is still increasing (albeit slightly) toward the end of training. We do not expect our models to achieve the same performance as in the paper, given our simplified training configuration. However, we argue that since the training appears to have converged to a stable minimum without overfitting, comparing the relative performance between the different methods should still be a sound approach for evaluating the validity of the results in the paper.

It is not clear from the paper whether SWA, SWAG-Diag, and SWAG are each trained in separate runs, or all derived from the same training run. Since SWAG is just an extension of SWAG-Diag, which in turn is just an extension of SWA, one could simply run the full SWAG algorithm once and obtain the other algorithms as well by making modifications only at test time. Since we see no real disadvantage of doing so, we choose to derive all methods from the same training run (i.e. run the full SWAG algorithm, and then also run SWA, SWAG-Diag, and SWA at test time). At test time we use the reported "sufficiently good" $S$ value of 30 samples for SWAG-Diag and SWAG.

In order to evaluate the difference between framework implementations, avoid overfitting and conduct additional experiments with varying hyper-parameters we split our data into 90% and 10% for train and validation respectively.

For each of the methods we do 3 training runs, with different random initialization. At test time, we estimate the expected calibration error (ECE), compute the accuracy and negative log-likelihood (NLL), and generate the modified version of a reliability diagram with the method described in the SWAG paper. For the ECE estimation we use 20 bins, just as when producing the reliability diagram. For the ECE, accuracy, and NLL we report the mean and the standard deviation of the 3 runs for each method, just as in the paper.

During training we found that the diagonal covariance matrix $\Sigma_{diag} = diag(\overline{\theta^2} - \overline{\theta}^2)$ to produce negative values which is inconsistent with variance being non-negative. This occurrence was not mentioned in the paper so its significance is unknown to us. However, upon review of the authors' code we found a solution where all values are clipped at $10^{-30}$ which we adopted.

We also conducted experiments with transfer learning where the models ware all trained on CIFAR-10 and then evaluated on STL-10. The two datasets have similar classes but the image distribution is different. When considering that the distribution is different from what was trained on, it is expected for the model to exhibit a larger uncertainty no matter the accuracy. How the ECE and reliability change due to this is interesting since most real world applications are susceptible to out of distribution samples.

## 4 Additional experiments

In an attempt to validate the necessity of the full SWAG method for sampling, we propose a naive approach to creating an ensemble. We do this by simply drawing samples from a compeletely arbitrarily devised Gaussian distribution with the SGD model parameters as mean and a factor $\alpha$ of their values as standard deviation, such that

$$\theta \sim N(\theta_{SGD}, \alpha \cdot \text{diag}(\theta_{SGD})). \tag{4}$$

This naive method of sampling makes no attempt to estimate the actual weight distribution, and should therefore not yield better results than SWAG-Diag or SWAG, proving the importance of a

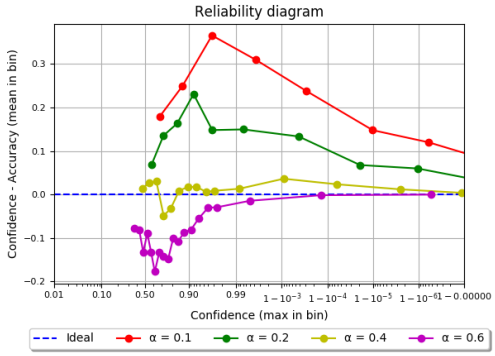



Figure 3: Reliability diagram when testing on CIFAR-10 for SGD-Noise with different values of the parameter $\alpha$.

good estimator. To keep the comparisons fair we sample $S = 30$, as with SWAG-Diag and SWAG. To select an appropriate value for $\alpha$, we test this method on the validation set of CIFAR-10 for a range of different $\alpha$-values and select a value that achieves respectable performance in terms of calibration, negative log-likelihood and accuracy.

## 5 Results

In this section we present the results from our reproductions of the methods as well as from our additional experiments.

In Table 1 we report mean and standard deviation of negative log-likelihood (NLL), expected calibration error (ECE), and accuracy, when testing our SGD-Noise method on the CIFAR-10 validation set, with a variety of values of the parameter $\alpha$. A corresponding reliability diagram, for each value of $\alpha$ can be found in Figure 3. We observe that $\alpha = 0.4$ appears to yield decent performance in terms of calibration and accuracy, so we opt to use this value for the rest of our comparisons to the methods from the paper.

| Dataset | 0.1 | 0.2 | 0.4 | 0.6 |
|---|---|---|---|---|
| NLL | $0.9006 \pm 0.0239$ | $0.6405 \pm 0.0183$ | $\mathbf{0.4747 \pm 0.0032}$ | $0.5666 \pm 0.0279$ |
| ECE | $0.0919 \pm 0.0003$ | $0.0630 \pm 0.0007$ | $\mathbf{0.0130 \pm 0.0032}$ | $0.0784 \pm 0.0035$ |
| Accuracy | $\mathbf{0.8701 \pm 0.0007}$ | $0.8725 \pm 0.0013$ | $0.8611 \pm 0.0044$ | $0.8251 \pm 0.0148$ |

Table 1: NLL, ECE and Accuracy evaluated for SGD-Noise with varying $\alpha$ trained and evaluated on CIFAR-10.

In Table 2 we report the mean and standard deviation of NLL for all considered models, on all considered datasets, for the methods from the paper as well as for our SGD-Noise with $\alpha = 0.4$. Similarly, the ECE is presented in Table 3 and accuracy in Table 4. As a visual complement, the corresponding reliability plots are presented in Figure 4 and Figure 5, providing further insight into how reliability varies in different confidence regions.

## 6 Discussion

Firstly, we need to keep in mind the possibility that, due to our baseline implementation not being identical to that of the paper, the results might be inherently flawed. However, the authors made no claim that their method was only applicable to certain training schemes or only high preforming models. With that in mind we continue to analyze our findings as relevant and valid.

In accordance with the SWAG paper we find that all proposed models improve on NLL and ECE compared to standard SGD. For NLL we see the same relative improvement of SWAG/SGD as they do, while ECE improves to a lesser extent in our case. Interestingly, the SGD-Noise model improves the most over all the datasets. This is contradicting to both our initial expectation and the

| Dataset | SGD | SWA | SWAG-Diag | SWAG | SGD-Noise($\alpha = 0.4$) |
|---|---|---|---|---|---|
| CIFAR-10 | $1.2179 \pm 0.0177$ | $0.8646 \pm 0.0163$ | $0.8233 \pm 0.0430$ | $0.8298 \pm 0.0164$ | $\mathbf{0.4721 \pm 0.0111}$ |
| CIFAR-100 | $5.8375 \pm 0.4832$ | $4.5064 \pm 0.4206$ | $3.5330 \pm 0.9499$ | $3.4998 \pm 0.9175$ | $\mathbf{2.3309 \pm 0.1216}$ |
| CIFAR-10 $\rightarrow$ STL-10 | $13.7784 \pm 0.2199$ | $7.9330 \pm 0.0694$ | $7.6975 \pm 0.0748$ | $7.7937 \pm 0.0642$ | $\mathbf{5.6247 \pm 0.2322}$ |

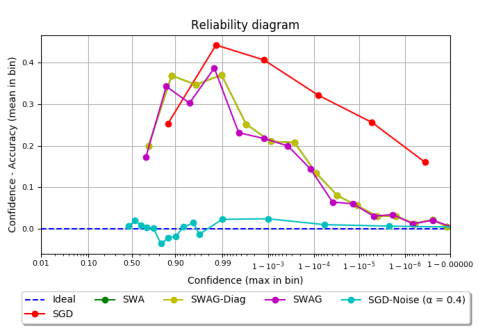
Table 2: Negative log-loss of evaluated models on different datasets.

| Dataset | SGD | SWA | SWAG-Diag | SWAG | SGD-Noise($\alpha = 0.4$) |
|---|---|---|---|---|---|
| CIFAR-10 | $0.1119 \pm 0.0002$ | $0.1159 \pm 0.0014$ | $0.1100 \pm 0.0059$ | $0.1111 \pm 0.0015$ | $\mathbf{0.0114 \pm 0.0037}$ |
| CIFAR-100 | $0.3652 \pm 0.0205$ | $0.3517 \pm 0.0312$ | $0.2660 \pm 0.1444$ | $0.2615 \pm 0.1450$ | $\mathbf{0.0571 \pm 0.0129}$ |
| CIFAR-10 $\rightarrow$ STL-10 | $0.5597 \pm 0.0016$ | $0.5447 \pm 0.0020$ | $0.5275 \pm 0.0016$ | $0.5339 \pm 0.0017$ | $\mathbf{0.3621 \pm 0.0107}$ |

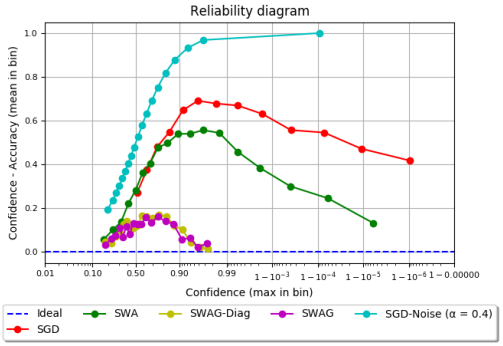
Table 3: Expected calibration error of evaluated models on different datasets.

| Dataset | SGD | SWA | SWAG-Diag | SWAG | SGD-Noise($\alpha = 0.4$) |
|---|---|---|---|---|---|
| CIFAR-10 | $\mathbf{0.8690 \pm 0.0008}$ | $0.8466 \pm 0.0014$ | $0.8471 \pm 0.0012$ | $0.8469 \pm 0.0017$ | $0.8592 \pm 0.0033$ |
| CIFAR-100 | $0.4941 \pm 0.0274$ | $0.4565 \pm 0.0274$ | $0.4797 \pm 0.0120$ | $0.4807 \pm 0.0118$ | $\mathbf{0.5141 \pm 0.0116}$ |
| CIFAR-10 $\rightarrow$ STL-10 | $\mathbf{0.3989 \pm 0.0034}$ | $0.3867 \pm 0.0031$ | $0.3865 \pm 0.0031$ | $0.3878 \pm 0.0028$ | $0.3840 \pm 0.0080$ |

Table 4: Accuracy of evaluated models on different datasets.




(a) Reliability diagram when testing on CIFAR-10.




(b) Reliability diagram when testing on CIFAR-100.

Figure 4: Reliability diagrams when testing on CIFAR-10 and CIFAR-100

reported reasoning for SWAG: that a good estimation of the prior from which to draw samples is essential. Our findings suggest that it is simply the act of sampling around a good set of parameters that constitutes the improvements. SWAG-Diag and most of our SGD-Noise models achieve lower NLL and ECE than SWAG, indicating at the very least that the prior estimation is not the main contributor to improvement.

The accuracy is slightly lower than that for SGD for all models on all datasets with the exception of SGD-Noise on CIFAR-100. This again causes concern to the validity of our results as it also contradicts the findings for SWA in [2]. On the other hand, in the both the SWA and SWAG papers,

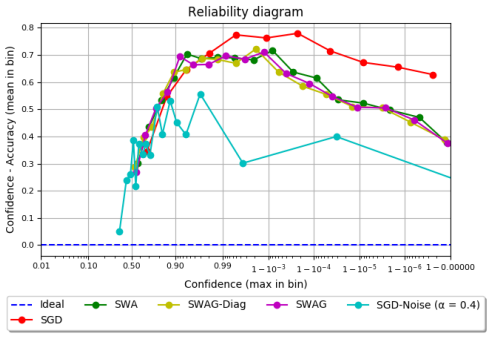



Figure 5: Reliability diagram when training on CIFAR-10, and testing on STL-10.

the observed improvements in accuracy are minor (on the scale of 1 %), so this may not be sufficient grounds to say our results differ significantly.

Transfer learning in our experiments suffers greatly, which is expected considering the very limited regularization applied. However, the loss of accuracy is roughly the same for the different models allowing relevant analysis of NLL and ECE. As expected, both NLL and ECE are reduced for for all the proposed models but surprisingly SGD-Noise was again the best.

## 7 Conclusions

In terms of ease of reproducibility, we found some difficulty to reproduce certain details of the paper but could to an extent find answers hidden within referenced code. As mentioned, translating the baseline implementation from PyTorch to TensorFlow proved difficult, or rather the provided hyper-parameters did not translate well. Ideally one would redo experiments after achieving an identical baseline in order to verify the validity of the results presented in this report.

However, assuming that our results are valid, we find they show similar improvements to calibration when applying SWAG as was presented in the paper. We do not however, observe the same improvements in accuracy as in the paper. Worthy of note is also that our simple method, SGD-Noise, achieves similar, or better performance than the methods suggested in the paper, which may suggest that accurately estimating the prior weight distribution is not of as vital importance as perceived by the authors.

### Acknowledgments

We would like to thank Erik Englesson for providing some useful guidance for our project, and Hossein Azizpour for helping identify a critical error we had originally made in our implementation.

## References

[1] W. Maddox, T. Garipov, P. Izmailov, D. Vetrov, and A. G. Wilson, "A Simple Baseline for Bayesian Uncertainty in Deep Learning," 2 2019. [Online]. Available: http://arxiv.org/abs/1902.02476

[2] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging Weights Leads to Wider Optima and Better Generalization," 3 2018. [Online]. Available: http://arxiv.org/abs/1803.05407

[3] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," 6 2017. [Online]. Available: http://arxiv.org/abs/1706.04599

[4] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 9 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

## A Updated results with authors' optimizer configuration

In this appendix, we present updated results after fixing a previously unknown mistake in our recreation of the authors optimization configuration. The cause of the behavior shown in Figure 2 was that the weight decay in our optimizer was decoupled from the loss, in such a way that the optimizer first takes a gradient step, and then decays the network weights separately. As the learning rate is annealed, the weight decay remains constant and eventually fully dominates the cross-entropy loss, causing the training to explode. After fixing this error, and thereby enabling a setup identical to that of the paper, we find much better convergence where the loss monotonically decreases. In the following subsections, we present our results with this configuration and briefly append to the previous discussion and conclusion.

### A.1 Results

In Tables 5, 6, and 7 we present mean and standard deviation of NLL, ECE, and accuracy for all methods on all datasets.

| Dataset | SGD | SWA | SWAG-Diag | SWAG | SGD-Noise($\alpha = 0.4$) |
|---|---|---|---|---|---|
| CIFAR-10 | $0.4323 \pm 0.0066$ | $0.3663 \pm 0.0083$ | $0.3079 \pm 0.0047$ | $0.3262 \pm 0.0060$ | $\mathbf{0.2795 \pm 0.0027}$ |
| CIFAR-100 | $2.3205 \pm 0.0646$ | $1.9320 \pm 0.0868$ | $1.5081 \pm 0.0757$ | $1.6199 \pm 0.0882$ | $\mathbf{1.4852 \pm 0.0259}$ |
| CIFAR-10 $\rightarrow$ STL-10 | $6.1335 \pm 0.0707$ | $5.7642 \pm 0.0794$ | $5.0635 \pm 0.0627$ | $5.2649 \pm 0.0965$ | $\mathbf{4.4450 \pm 0.0103}$ |

Table 5: Negative log-loss of evaluated models on different datasets (authors' training configuration).

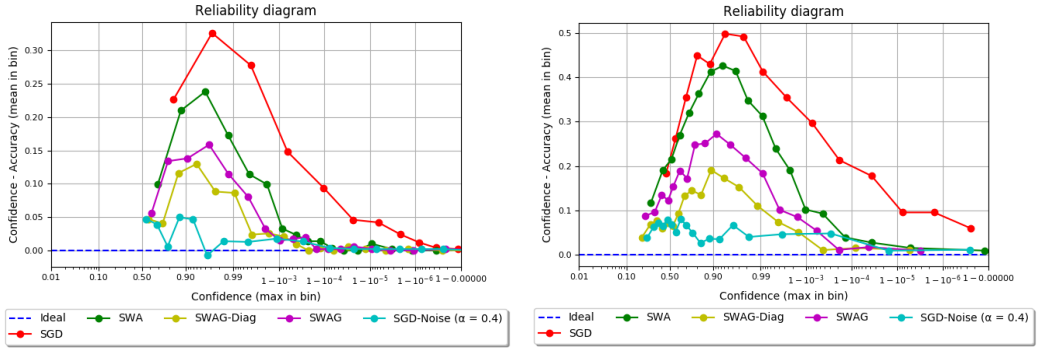
| Dataset | SGD | SWA | SWAG-Diag | SWAG | SGD-Noise($\alpha = 0.4$) |
|---|---|---|---|---|---|
| CIFAR-10 | $0.0595 \pm 0.0017$ | $0.0541 \pm 0.0020$ | $0.0315 \pm 0.0013$ | $0.0407 \pm 0.0014$ | $\mathbf{0.0124 \pm 0.0014}$ |
| CIFAR-100 | $0.2292 \pm 0.0072$ | $0.2016 \pm 0.0102$ | $0.0824 \pm 0.0116$ | $0.1254 \pm 0.0145$ | $\mathbf{0.0476 \pm 0.0056}$ |
| CIFAR-10 $\rightarrow$ STL-10 | $0.5061 \pm 0.0051$ | $0.4783 \pm 0.0025$ | $0.4272 \pm 0.0045$ | $0.4480 \pm 0.0022$ | $\mathbf{0.3983 \pm 0.0017}$ |

Table 6: Expected calibration error of evaluated models on different datasets (authors' training configuration).

| Dataset | SGD | SWA | SWAG-Diag | SWAG | SGD-Noise($\alpha = 0.4$) |
|---|---|---|---|---|---|
| CIFAR-10 | $\mathbf{0.9188 \pm 0.0018}$ | $0.9070 \pm 0.0025$ | $0.9067 \pm 0.0021$ | $0.9074 \pm 0.0017$ | $0.9119 \pm 0.0014$ |
| CIFAR-100 | $\mathbf{0.6684 \pm 0.0062}$ | $0.6343 \pm 0.0136$ | $0.6401 \pm 0.0137$ | $0.6387 \pm 0.0139$ | $0.6536 \pm 0.0042$ |
| CIFAR-10 $\rightarrow$ STL-10 | $\mathbf{0.4292 \pm 0.0035}$ | $0.4258 \pm 0.0008$ | $0.4263 \pm 0.0016$ | $0.4263 \pm 0.0010$ | $0.4288 \pm 0.0023$ |

Table 7: Accuracy of evaluated models on different datasets (authors' training configuration).

The resulting reliability diagrams when testing on CIFAR-10 and CIFAR-100 are shown in Figure 6. The resulting reliability diagram when training on CIFAR-10, and testing on STL-10 is shown in Figure 7.




(a) Reliability diagram when testing on CIFAR-10. (b) Reliability diagram when testing on CIFAR-100.

Figure 6: Reliability diagrams when testing on CIFAR-10 and CIFAR-100 (authors' training configuration).

## A.2 Discussion

In relative comparison between the different methods, we see similar results to our previous findings. However, compared to the original paper, we do not see the same improvement in ECE, NLL, nor accuracy when using any of the SWA/SWAG-Diag/SWAG methods over regular SGD. We note also, that for regular SGD we do not reach the level of performance achieved by the authors in any of the metrics (NLL, ECE, and accuracy), which would indicate that our implementation does not mimic theirs exactly. Since this difference in baseline performance is still rather small, we assume it stems from differences in the frameworks, or minor differences in the implementations, rather than additional implementational errors.

Looking at the reliability diagrams also further confirms the results from our previous findings. SGD-Noise outperforms the other methods in terms of calibration, closely followed by SWAG-Diag and SWAG.

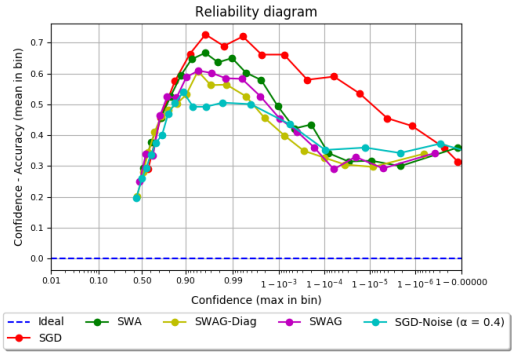



Figure 7: Reliability diagram when training on CIFAR-10, and testing on STL-10 (authors' training configuration).

### A.3 Conclusion

With our mistake corrected, we find very similar trends to our original findings. Considering that our mistake constitutes a completely separate case, we consider our analysis to have increased credibility, that sampling is the main source of improvements rather than the quality of the estimated prior weight distribution.