A Simple Baseline for Bayesian Uncertainty in Deep Learning

Wesley J. Maddox[*1] **Timur Garipov**[*2] **Pavel Izmailov**[*1]
Dmitry Vetrov[2,3] **Andrew Gordon Wilson**[1]

[1] New York University
[2] Samsung AI Center Moscow
[3] Samsung-HSE Laboratory, National Research University Higher School of Economics

Abstract

We propose SWA-Gaussian (SWAG), a simple, scalable, and general purpose approach for uncertainty representation and calibration in deep learning. Stochastic Weight Averaging (SWA), which computes the first moment of stochastic gradient descent (SGD) iterates with a modified learning rate schedule, has recently been shown to improve generalization in deep learning. With SWAG, we fit a Gaussian using the SWA solution as the first moment and a low rank plus diagonal covariance also derived from the SGD iterates, forming an approximate posterior distribution over neural network weights; we then sample from this Gaussian distribution to perform Bayesian model averaging. We empirically find that SWAG approximates the shape of the true posterior, in accordance with results describing the stationary distribution of SGD iterates. Moreover, we demonstrate that SWAG performs well on a wide variety of tasks, including out of sample detection, calibration, and transfer learning, in comparison to many popular alternatives including MC dropout, KFAC Laplace, SGLD, and temperature scaling.

1 Introduction

Ultimately, machine learning models are used to make decisions. Representing uncertainty is crucial for decision making. For example, in medical diagnoses and autonomous vehicles we want to protect against rare but costly mistakes. Deep learning models typically lack a representation of uncertainty, and provide overconfident and miscalibrated predictions [e.g., 28, 19].

Bayesian methods provide a natural probabilistic representation of uncertainty in deep learning [e.g., 6, 31, 9], and previously had been a gold standard for inference with neural networks [49]. However, existing approaches are often highly sensitive to hyperparameter choices, and hard to scale to modern datasets and architectures, which limits their general applicability in modern deep learning.

In this paper we propose a different approach to Bayesian deep learning: we use the information contained in the SGD trajectory to efficiently approximate the posterior distribution over the weights of the neural network. We find that the Gaussian distribution fitted to the first two moments of SGD iterates, with a modified learning rate schedule, captures the local geometry of the posterior surprisingly well. Using this Gaussian distribution we are able to obtain convenient, efficient, accurate and well-calibrated predictions in a broad range of tasks in computer vision. In particular, our contributions are the following:

- In this work we propose SWAG (SWA-Gaussian), a scalable approximate Bayesian inference technique for deep learning. SWAG builds on Stochastic Weight Averaging [27], which

[*]Equal contribution. Correspondence to wjm363 AT nyu.edu

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

computes an average of SGD iterates with a high constant learning rate schedule, to provide improved generalization in deep learning and the interpretation of SGD as approximate Bayesian inference [43]. SWAG additionally computes a low-rank plus diagonal approximation to the covariance of the iterates, which is used together with the SWA mean, to define a Gaussian posterior approximation over neural network weights.

- SWAG is motivated by the theoretical analysis of the stationary distribution of SGD iterates [e.g., 43, 10], which suggests that the SGD trajectory contains useful information about the geometry of the posterior. In Appendix B we show that the assumptions of Mandt et al. [43] do not hold for deep neural networks, due to non-convexity and over-parameterization (with further analysis in the supplementary material). However, we find in Section 4 that in the low-dimensional subspace spanned by SGD iterates the shape of the posterior distribution is approximately Gaussian within a basin of attraction. Further, SWAG is able to capture the geometry of this posterior remarkably well.
- In an exhaustive empirical evaluation we show that SWAG can provide well-calibrated uncertainty estimates for neural networks across many settings in computer vision. In particular SWAG achieves higher test likelihood compared to many state-of-the-art approaches, including MC-Dropout [14], temperature scaling [19], SGLD [59], KFAC-Laplace [54] and SWA [27] on CIFAR-10, CIFAR-100 and ImageNet, on a range of architectures. We also demonstrate the effectiveness of SWAG for out-of-domain detection, and transfer learning. While we primarily focus on image classification, we show that SWAG can significantly improve test perplexities of LSTM networks on language modeling problems, and in Appendix G we also compare SWAG with Probabilistic Back-propagation (PBP) [23], Deterministic Variational Inference (DVI) [60], and Deep Gaussian Processes [7] on regression problems.
- We release PyTorch code at `https://github.com/wjmaddox/swa_gaussian`.

2 Related Work

2.1 Bayesian Methods

Bayesian approaches represent uncertainty by placing a distribution over model parameters, and then marginalizing these parameters to form a whole predictive distribution, in a procedure known as Bayesian model averaging. In the late 1990s, Bayesian methods were the state-of-the-art approach to learning with neural networks, through the seminal works of Neal [49] and MacKay [40]. However, modern neural networks often contain millions of parameters, the posterior over these parameters (and thus the loss surface) is highly non-convex, and mini-batch approaches are often needed to move to a space of good solutions [29]. For these reasons, Bayesian approaches have largely been intractable for modern neural networks. Here, we review several modern approaches to Bayesian deep learning.

Markov chain Monte Carlo (MCMC) was at one time a gold standard for inference with neural networks, through the Hamiltonian Monte Carlo (HMC) work of Neal [49]. However, HMC requires full gradients, which is computationally intractable for modern neural networks. To extend the HMC framework, stochastic gradient HMC (SGHMC) was introduced by Chen et al. [9] and allows for stochastic gradients to be used in Bayesian inference, crucial for both scalability and exploring a space of solutions that provide good generalization. Alternatively, stochastic gradient Langevin dynamics (SGLD) [59] uses first order Langevin dynamics in the stochastic gradient setting. Theoretically, both SGHMC and SGLD asymptotically sample from the posterior in the limit of infinitely small step sizes. In practice, using finite learning rates introduces approximation errors (see e.g. [43]), and tuning stochastic gradient MCMC methods can be quite difficult.

Variational Inference: Graves [18] suggested fitting a Gaussian variational posterior approximation over the weights of neural networks. This technique was generalized by Kingma and Welling [33] which proposed the *reparameterization trick* for training deep latent variable models; multiple variational inference methods based on the reparameterization trick were proposed for DNNs [e.g., 32, 6, 45, 39]. While variational methods achieve strong performance for moderately sized networks, they are empirically noted to be difficult to train on larger architectures such as deep residual networks [22]; Blier and Ollivier [5] argue that the difficulty of training is explained by variational methods

providing inusfficient data compression for DNNs despite being designed for data compression (minimum description length). Recent key advances [39, 60] in variational inference for deep learning typically focus on smaller-scale datasets and architectures. An alternative line of work re-interprets noisy versions of optimization algorithms: for example, noisy Adam [30] and noisy KFAC [64], as approximate variational inference.

Dropout Variational Inference: Gal and Ghahramani [14] used a spike and slab variational distribution to view dropout at test time as approximate variational Bayesian inference. Concrete dropout [15] extends this idea to optimize the dropout probabilities as well. From a practical perspective, these approaches are quite appealing as they only require ensembling dropout predictions at test time, and they were succesfully applied to several downstream tasks [28, 46].

Laplace Approximations assume a Gaussian posterior, $\mathcal{N}(\theta^*, \mathcal{I}(\theta^*)^{-1})$, where θ^* is a MAP estimate and $\mathcal{I}(\theta^*)^{-1}$ is the inverse of the Fisher information matrix (expected value of the Hessian evaluated at θ^*). It was notably used for Bayesian neural networks in MacKay [41], where a diagonal approximation to the inverse of the Hessian was utilized for computational reasons. More recently, Kirkpatrick et al. [34] proposed using diagonal Laplace approximations to overcome catastrophic forgetting in deep learning. Ritter et al. [54] proposed the use of either a diagonal or block Kronecker factored (KFAC) approximation to the Hessian matrix for Laplace approximations, and Ritter et al. [53] successfully applied the KFAC approach to online learning scenarios.

2.2 SGD Based Approximations

Mandt et al. [43] proposed to use the iterates of averaged SGD as an MCMC sampler, after analyzing the dynamics of SGD using tools from stochastic calculus. From a frequentist perspective, Chen et al. [10] showed that under certain conditions a batch means estimator of the sample covariance matrix of the SGD iterates converges to $A = \mathcal{H}(\theta)^{-1}C(\theta)\mathcal{H}(\theta)^{-1}$, where $\mathcal{H}(\theta)^{-1}$ is the inverse of the Hessian of the log likelihood and $C(\theta) = \mathbb{E}(\nabla \log p(\theta) \nabla \log p(\theta)^T)$ is the covariance of the gradients of the log likelihood. Chen et al. [10] then show that using A and the sample average of the iterates for a Gaussian approximation produces well calibrated confidence intervals of the parameters and that the variance of these estimators achieves the Cramer Rao lower bound (the minimum possible variance). A description of the asymptotic covariance of the SGD iterates dates back to Ruppert [55] and Polyak and Juditsky [52], who show asymptotic convergence of Polyak-Ruppert averaging.

2.3 Methods for Calibration of DNNs

Lakshminarayanan et al. [36] proposed using ensembles of several networks for enhanced calibration, and incorporated an adversarial loss function to be used when possible as well. Outside of probabilistic neural networks, Guo et al. [19] proposed temperature scaling, a procedure which uses a validation set and a single hyperparameter to rescale the logits of DNN outputs for enhanced calibration. Kuleshov et al. [35] propose calibrated regression using a similar rescaling technique.

3 SWA-Gaussian for Bayesian Deep Learning

In this section we propose SWA-Gaussian (SWAG) for Bayesian model averaging and uncertainty estimation. In Section 3.2, we review stochastic weight averaging (SWA) [27], which we view as estimating the mean of the stationary distribution of SGD iterates. We then propose SWA-Gaussian in Sections 3.3 and 3.4 to estimate the covariance of the stationary distribution, forming a Gaussian approximation to the posterior over weight parameters. With SWAG, uncertainty in weight space is captured with minimal modifications to the SWA training procedure. We then present further theoretical and empirical analysis for SWAG in Section 4.

3.1 Stochastic Gradient Descent (SGD)

Standard training of deep neural networks (DNNs) proceeds by applying stochastic gradient descent on the model weights θ with the following update rule:

$$\Delta\theta_t = -\eta_t \left(\frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \log p(y_i | f_\theta(x_i)) - \frac{\nabla_\theta \log p(\theta)}{N} \right),$$

where the learning rate is η, the ith input (e.g. image) and label are $\{x_i, y_i\}$, the size of the whole training set is N, the size of the batch is B, and the DNN, f, has weight parameters θ.[2] The loss function is a negative log likelihood $-\sum_i \log p(y_i|f_\theta(x_i))$, combined with a regularizer $\log p(\theta)$. This type of maximum likelihood training does not represent uncertainty in the predictions or parameters θ.

3.2 Stochastic Weight Averaging (SWA)

The main idea of SWA [27] is to run SGD with a constant learning rate schedule starting from a pre-trained solution, and to average the weights of the models it traverses. Denoting the weights of the network obtained after epoch i of SWA training θ_i, the SWA solution after T epochs is given by $\theta_{\text{SWA}} = \frac{1}{T}\sum_{i=1}^{T}\theta_i$. A high constant learning rate schedule ensures that SGD explores the set of possible solutions instead of simply converging to a single point in the weight space. Izmailov et al. [27] argue that conventional SGD training converges to the boundary of the set of high-performing solutions; SWA on the other hand is able to find a more centered solution that is robust to the shift between train and test distributions, leading to improved generalization performance. SWA and related ideas have been successfully applied to a wide range of applications [see e.g. 2, 61, 62, 51]. A related but different procedure is Polyak-Ruppert averaging [52, 55] in stochastic convex optimization, which uses a learning rate decaying to zero. Mandt et al. [43] interpret Polyak-Ruppert averaging as a sampling procedure, with convergence occurring to the true posterior under certain strong conditions. Additionally, they explore the theoretical feasibility of SGD (and averaged SGD) as an approximate Bayesian inference scheme; we test their assumptions in Appendix A.

3.3 SWAG-Diagonal

We first consider a simple diagonal format for the covariance matrix. In order to fit a diagonal covariance approximation, we maintain a running average of the second uncentered moment for each weight, and then compute the covariance using the following standard identity at the end of training: $\overline{\theta^2} = \frac{1}{T}\sum_{i=1}^{T}\theta_i^2$, $\Sigma_{\text{diag}} = \text{diag}(\overline{\theta^2} - \theta_{\text{SWA}}^2)$; here the squares in θ_{SWA}^2 and θ_i^2 are applied elementwise. The resulting approximate posterior distribution is then $\mathcal{N}(\theta_{\text{SWA}}, \Sigma_{\text{Diag}})$. In our experiments, we term this method SWAG-Diagonal.

Constructing the SWAG-Diagonal posterior approximation requires storing two additional copies of DNN weights: θ_{SWA} and $\overline{\theta^2}$. Note that these models do not have to be stored on the GPU. The additional computational complexity of constructing SWAG-Diagonal compared to standard training is negligible, as it only requires updating the running averages of weights once per epoch.

3.4 SWAG: Low Rank plus Diagonal Covariance Structure

We now describe the full SWAG algorithm. While the diagonal covariance approximation is standard in Bayesian deep learning [6, 34], it can be too restrictive. We extend the idea of diagonal covariance approximations to utilize a more flexible low-rank plus diagonal posterior approximation. SWAG approximates the sample covariance Σ of the SGD iterates along with the mean θ_{SWA}.[3]

Note that the sample covariance matrix of the SGD iterates can be written as the sum of outer products, $\Sigma = \frac{1}{T-1}\sum_{i=1}^{T}(\theta_i - \theta_{\text{SWA}})(\theta_i - \theta_{\text{SWA}})^\top$, and is of rank T. As we do not have access to the value of θ_{SWA} during training, we approximate the sample covariance with $\Sigma \approx \frac{1}{T-1}\sum_{i=1}^{T}(\theta_i - \bar{\theta}_i)(\theta_i - \bar{\theta}_i)^\top = \frac{1}{T-1}DD^\top$, where D is the deviation matrix comprised of columns $D_i = (\theta_i - \bar{\theta}_i)$, and $\bar{\theta}_i$ is the running estimate of the parameters' mean obtained from the first i samples. To limit the rank of the estimated covariance matrix we only use the last K of D_i vectors corresponding to the last K

[2] We ignore momentum for simplicity in this update; however we utilized momentum in the resulting experiments and it is covered theoretically [43].

[3] We note that stochastic gradient Monte Carlo methods [9, 59] also use the SGD trajectory to construct samples from the approximate posterior. However, these methods are principally different from SWAG in that they (1) require adding Gaussian noise to the gradients, (2) decay learning rate to zero and (3) do not construct a closed-form approximation to the posterior distribution, which for instance enables SWAG to draw new samples with minimal overhead. We include comparisons to SGLD [59] in the Appendix.

epochs of training. Here K is the rank of the resulting approximation and is a hyperparameter of the method. We define $\widehat{D}$ to be the matrix with columns equal to D_i for $i = T-K+1, \ldots, T$.

We then combine the resulting low-rank approximation $\Sigma_{\text{low-rank}} = \frac{1}{K-1} \cdot \widehat{D}\widehat{D}^\top$ with the diagonal approximation Σ_{diag} of Section 3.3. The resulting approximate posterior distribution is a Gaussian with the SWA mean θ_{SWA} and summed covariance: $\mathcal{N}(\theta_{\text{SWA}}, \frac{1}{2} \cdot (\Sigma_{\text{diag}} + \Sigma_{\text{low-rank}}))$.[4] In our experiments, we term this method SWAG. Computing this approximate posterior distribution requires storing K vectors D_i of the same size as the model as well as the vectors θ_{SWA} and $\overline{\theta^2}$. These models do not have to be stored on a GPU.

To sample from SWAG we use the following identity

$$\widetilde{\theta} = \theta_{\text{SWA}} + \frac{1}{\sqrt{2}} \cdot \Sigma_{\text{diag}}^{\frac{1}{2}} z_1 + \frac{1}{\sqrt{2(K-1)}} \widehat{D} z_2, \quad \text{where } z_1 \sim \mathcal{N}(0, I_d),\ z_2 \sim \mathcal{N}(0, I_K). \tag{1}$$

Here d is the number of parameters in the network. Note that Σ_{diag} is diagonal, and the product $\Sigma_{\text{diag}}^{\frac{1}{2}} z_1$ can be computed in $\mathcal{O}(d)$ time. The product $\widehat{D} z_2$ can be computed in $\mathcal{O}(Kd)$ time.

Related methods for estimating the covariance of SGD iterates were considered in Mandt et al. [43] and Chen et al. [10], but store full-rank covariance Σ and thus scale quadratically in the number of parameters, which is prohibitively expensive for deep learning applications. We additionally note that using the deviation matrix for online covariance matrix estimation comes from viewing the online updates used in Dasgupta and Hsu [12] in matrix fashion.

The full Bayesian model averaging procedure is given in Algorithm 1. As in Izmailov et al. [27] (SWA) we update the batch normalization statistics after sampling weights for models that use batch normalization [25]; we investigate the necessity of this update in Appendix D.4.

Algorithm 1 Bayesian Model Averaging with SWAG

θ_0: pretrained weights; η: learning rate; T: number of steps; c: moment update frequency; K: maximum number of columns in deviation matrix; S: number of samples in Bayesian model averaging

Train SWAG
- $\overline{\theta} \leftarrow \theta_0,\ \overline{\theta^2} \leftarrow \theta_0^2$ {Initialize moments}
- **for** $i \leftarrow 1, 2, ..., T$ **do**
  - $\theta_i \leftarrow \theta_{i-1} - \eta \nabla_\theta \mathcal{L}(\theta_{i-1})$ {Perform SGD update}
  - **if** $\texttt{MOD}(i, c) = 0$ **then**
    - $n \leftarrow i/c$ {Number of models}
    - $\overline{\theta} \leftarrow \frac{n\overline{\theta} + \theta_i}{n+1},\ \overline{\theta^2} \leftarrow \frac{n\overline{\theta^2} + \theta_i^2}{n+1}$ {Moments}
    - **if** $\texttt{NUM\_COLS}(\widehat{D}) = K$ **then**
      - $\texttt{REMOVE\_COL}(\widehat{D}[:, 1])$
    - $\texttt{APPEND\_COL}(\widehat{D}, \theta_i - \overline{\theta})$ {Store deviation}
- **return** $\theta_{\text{SWA}} = \overline{\theta},\ \Sigma_{\text{diag}} = \overline{\theta^2} - \overline{\theta}^2,\ \widehat{D}$

Test Bayesian Model Averaging
- **for** $i \leftarrow 1, 2, ..., S$ **do**
  - Draw $\widetilde{\theta}_i \sim \mathcal{N}\left(\theta_{\text{SWA}}, \frac{1}{2}\Sigma_{\text{diag}} + \frac{\widehat{D}\widehat{D}^\top}{2(K-1)}\right)$ (1)
  - Update batch norm statistics with new sample.
  - $p(y^*|\text{Data}) \mathrel{+}= \frac{1}{S} p(y^*|\widetilde{\theta}_i)$
- **return** $p(y^*|\text{Data})$

3.5 Bayesian Model Averaging with SWAG

Maximum a-posteriori (MAP) optimization is a procedure whereby one maximizes the (log) posterior with respect to parameters θ: $\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}|\theta) + \log p(\theta)$. Here, the prior $p(\theta)$ is viewed as a regularizer in optimization. However, MAP is *not* Bayesian inference, since one only considers a single setting of the parameters $\hat{\theta}_{\text{MAP}} = \text{argmax}_\theta p(\theta|\mathcal{D})$ in making predictions, forming $p(y_*|\hat{\theta}_{\text{MAP}}, x_*)$, where x_* and y_* are test inputs and outputs.

A Bayesian procedure instead *marginalizes* the posterior distribution over θ, in a Bayesian model average, for the unconditional predictive distribution: $p(y_*|\mathcal{D}, x_*) = \int p(y_*|\theta, x_*) p(\theta|\mathcal{D}) d\theta$. In practice, this integral is computed through a Monte Carlo sampling procedure:
$p(y_*|\mathcal{D}, x_*) \approx \frac{1}{T} \sum_{t=1}^{T} p(y_*|\theta_t, x_*), \quad \theta_t \sim p(\theta|\mathcal{D})$.

We emphasize that in this paper we are approximating *fully Bayesian inference*, rather than MAP optimization. We develop a Gaussian approximation to the posterior from SGD iterates, $p(\theta|\mathcal{D}) \approx$

[4] We use one half as the scale here because both the diagonal and low rank terms include the variance of the weights. We tested several other scales in Appendix D.

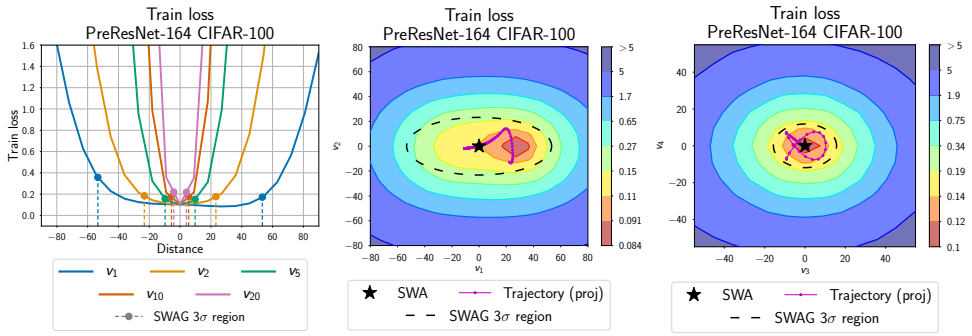


Figure 1: **Left:** Posterior joint density cross-sections along the rays corresponding to different eigenvectors of SWAG covariance matrix. **Middle:** Posterior joint density surface in the plane spanned by eigenvectors of SWAG covariance matrix corresponding to the first and second largest eigenvalues and (**Right:**) the third and fourth largest eigenvalues. All plots are produced using PreResNet-164 on CIFAR-100. The SWAG distribution projected onto these directions fits the geometry of the posterior density remarkably well.

$\mathcal{N}(\theta; \mu, \Sigma)$, and then sample from this posterior distribution to perform a Bayesian model average. In our procedure, optimization with different regularizers, to characterize the Gaussian posterior approximation, corresponds to approximate Bayesian inference with different priors $p(\theta)$.

Prior Choice Typically, weight decay is used to regularize DNNs, corresponding to explicit L2 regularization when SGD without momentum is used to train the model. When SGD is used *with* momentum, as is typically the case, implicit regularization still occurs, producing a vague prior on the weights of the DNN in our procedure. This regularizer can be given an explicit Gaussian-like form (see Proposition 3 of Loshchilov and Hutter [38]), corresponding to a prior distribution on the weights.

Thus, SWAG is an approximate Bayesian inference algorithm in our experiments (see Section 5) and can be applied to most DNNs without any modifications of the training procedure (as long as SGD is used with weight decay or explicit L2 regularization). Alternative regularization techniques could also be used, producing different priors on the weights. It may also be possible to similarly utilize Adam and other stochastic first-order methods, which view as a promising direction for future work.

4 Does the SGD Trajectory Capture Loss Geometry?

To analyze the quality of the SWAG approximation, we study the posterior density along the directions corresponding to the eigenvectors of the SWAG covariance matrix for PreResNet-164 on CIFAR-100. In order to find these eigenvectors we use randomized SVD [21].[5] In the left panel of Figure 1 we visualize the ℓ_2-regularized cross-entropy loss $L(\cdot)$ (equivalent to the joint density of the weights and the loss with a Gaussian prior) as a function of distance t from the SWA solution θ_{SWA} along the i-th eigenvector v_i of the SWAG covariance: $\phi(t) = L(\theta_{\text{SWA}} + t \cdot \frac{v_i}{\|v_i\|})$. Figure 1 (left) shows a clear correlation between the variance of the SWAG approximation and the width of the posterior along the directions v_i. The SGD iterates indeed contain useful information about the shape of the posterior distribution, and SWAG is able to capture this information. We repeated the same experiment for SWAG-Diagonal, finding that there was almost no variance in these eigen-directions. Next, in Figure 1 (middle) we plot the posterior density surface in the 2-dimensional plane in the weight space spanning the two top eigenvectors v_1 and v_2 of the SWAG covariance: $\psi(t_1, t_2) = L(\theta_{\text{SWA}} + t_1 \cdot \frac{v_1}{\|v_1\|} + t_2 \cdot \frac{v_2}{\|v_2\|})$. Again, SWAG is able to capture the geometry of the posterior. The contours of constant posterior density appear remarkably well aligned with the eigenvalues of the SWAG covariance. We also present the analogous plot for the third and fourth top eigenvectors in Figure 1 (right). In Appendix C, we additionally present similar results for PreResNet-164 on CIFAR-10 and VGG-16 on CIFAR-100.

As we can see, SWAG is able to capture the geometry of the posterior in the subspace spanned by SGD iterates. However, the dimensionality of this subspace is very low compared to the dimensionality of

[5] From `sklearn.decomposition.TruncatedSVD`.

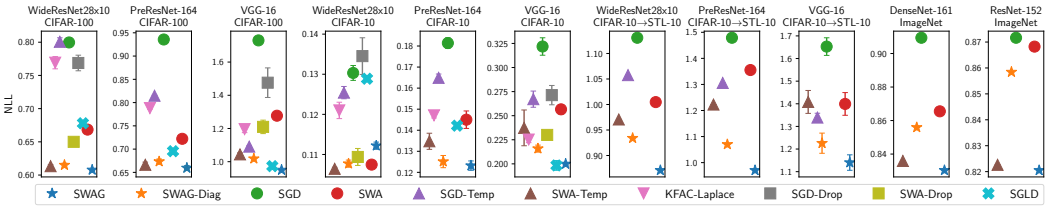


Figure 2: Negative log likelihoods for SWAG and baselines. Mean and standard deviation (shown with error-bars) over 3 runs are reported for each experiment on CIFAR datasets. SWAG (blue star) consistently outperforms alternatives, with lower negative log likelihood, with the largest improvements on transfer learning. Temperature scaling applied on top of SWA (SWA-Temp) often performs close to as well on the non-transfer learning tasks, but requires a validation set.

the weight space, and we can not guarantee that SWAG variance estimates are adequate along all directions in weight space. In particular, we would expect SWAG to under-estimate the variances along random directions, as the SGD trajectory is in a low-dimensional subspace of the weight space, and a random vector has a close-to-zero projection on this subspace with high probability. In Appendix A we visualize the trajectory of SGD applied to a quadratic function, and further discuss the relation between the geometry of objective and SGD trajectory. In Appendices A and B, we also empirically test the assumptions behind theory relating the SGD stationary distribution to the true posterior for neural networks.

5 Experiments

We conduct a thorough empirical evaluation of SWAG, comparing to a range of high performing baselines, including MC dropout [14], temperature scaling [19], SGLD [59], Laplace approximations [54], deep ensembles [36], and ensembles of SGD iterates that were used to construct the SWAG approximation. In Section 5.1 we evaluate SWAG predictions and uncertainty estimates on image classification tasks. We also evaluate SWAG for transfer learning and out-of-domain data detection. We investigate the effect of hyperparameter choices and practical limitations in SWAG, such as the effect of learning rate on the scale of uncertainty, in Appendix D.

5.1 Calibration and Uncertainty Estimation on Image Classification Tasks

In this section we evaluate the quality of uncertainty estimates as well as predictive accuracy for SWAG and SWAG-Diagonal on CIFAR-10, CIFAR-100 and ImageNet ILSVRC-2012 [56].

For all methods we analyze test negative log-likelihood, which reflects both the accuracy and the quality of predictive uncertainty. Following Guo et al. [19] we also consider a variant of *reliability diagrams* to evaluate the calibration of uncertainty estimates (see Figure 3) and to show the difference between a method's confidence in its predictions and its accuracy. To produce this plot for a given method we split the test data into 20 bins uniformly based on the confidence of a method (maximum predicted probability). We then evaluate the accuracy and mean confidence of the method on the images from each bin, and plot the difference between confidence and accuracy. For a well-calibrated model, this difference should be close to zero for each bin. We found that this procedure gives a more effective visualization of the actual confidence distribution of DNN predictions than the standard reliability diagrams used in Guo et al. [19] and Niculescu-Mizil and Caruana [50].

We provide tables containing the test accuracy, negative log likelihood and expected calibration error for all methods and datasets in Appendix E.3.

CIFAR datasets On CIFAR datasets we run experiments with VGG-16, PreResNet-164 and WideResNet-28x10 networks. In order to compare SWAG with existing alternatives we report the results for standard SGD and SWA [27] solutions (single models), MC-Dropout [14], temperature scaling [19] applied to SWA and SGD solutions, SGLD [59], and K-FAC Laplace [54] methods. For all the methods we use our implementations in PyTorch (see Appendix H). We train all networks for 300 epochs, starting to collect models for SWA and SWAG approximations once per epoch after epoch 160. For SWAG, K-FAC Laplace, and Dropout we use 30 samples at test time.

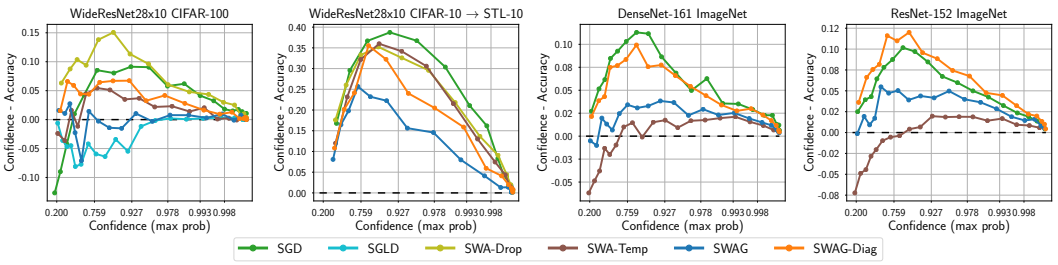


Figure 3: Reliability diagrams for WideResNet28x10 on CIFAR-100 and transfer task; ResNet-152 and DenseNet-161 on ImageNet. Confidence is the value of the max softmax output. A perfectly calibrated network has no difference between confidence and accuracy, represented by a dashed black line. Points below this line correspond to under-confident predictions, whereas points above the line are overconfident predictions. SWAG is able to substantially improve calibration over standard training (SGD), as well as SWA. Additionally, SWAG significantly outperforms temperature scaling for transfer learning (CIFAR-10 to STL), where the target data are not from the same distribution as the training data.

ImageNet On ImageNet we report our results for SWAG, SWAG-Diagonal, SWA and SGD. We run experiments with DenseNet-161 [24] and Resnet-152 [22]. For each model we start from a pre-trained model available in the `torchvision` package, and run SGD with a constant learning rate for 10 epochs. We collect models for the SWAG versions and SWA 4 times per epoch. For SWAG we use 30 samples from the posterior over network weights at test-time, and use randomly sampled 10% of the training data to update batch-normalization statistics for each of the samples. For SGD with temperature scaling, we use the results reported in Guo et al. [19].

Transfer from CIFAR-10 to STL-10 We use the models trained on CIFAR-10 and evaluate them on STL-10 [11]. STL-10 has a similar set of classes as CIFAR-10, but the image distribution is different, so adapting the model from CIFAR-10 to STL-10 is a commonly used transfer learning benchmark. We provide further details on the architectures and hyperparameters in Appendix H.

Results We visualize the negative log-likelihood for all methods and datasets in Figure 2. On all considered tasks SWAG and SWAG diagonal perform comparably or better than all the considered alternatives, SWAG being best overall. We note that the combination of SWA and temperature scaling presents a competitive baseline. However, unlike SWAG it requires using a validation set to tune the temperature; further, temperature scaling is not effective when the test data distribution differs from train, as we observe in experiments on transfer learning from CIFAR-10 to STL-10.

Next, we analyze the calibration of uncertainty estimates provided by different methods. In Figure 3 we present reliability plots for WideResNet on CIFAR-100, DenseNet-161 and ResNet-152 on ImageNet. The reliability diagrams for all other datasets and architectures are presented in the Appendix E.1. As we can see, SWAG and SWAG-Diagonal both achieve good calibration across the board. The low-rank plus diagonal version of SWAG is generally better calibrated than SWAG-Diagonal. We also present the expected calibration error for each of the methods, architectures and datasets in Tables A.3,4. Finally, in Tables A.9,10 we present the predictive accuracy for all of the methods, where SWAG is comparable with SWA and generally outperforms the other approaches.

5.2 Comparison to ensembling SGD solutions

We evaluated ensembles of independently trained SGD solutions (Deep Ensembles, [36]) on PreResNet-164 on CIFAR-100. We found that an ensemble of 3 SGD solutions has high accuracy (82.1%), but only achieves NLL 0.6922, which is *worse than a single SWAG solution* (0.6595 NLL). While the accuracy of this ensemble is high, SWAG solutions are much better calibrated. An ensemble of 5 SGD solutions achieves NLL 0.6478, which is *competitive with a single SWAG solution, that requires* $5\times$ *less computation to train*. Moreover, we can similarly ensemble independently trained SWAG models; an ensemble of 3 SWAG models achieves NLL of 0.6178.

We also evaluated ensembles of SGD iterates that were used to construct the SWAG approximation (SGD-Ens) for all of our CIFAR models. SWAG has higher NLL than SGD-Ens on VGG-16, but

much lower NLL on the larger PreResNet-164 and WideResNet28x10; the results for accuracy and ECE are analogous.

5.3 Out-of-Domain Image Detection

To evaluate SWAG on out-of-domain data detection we train a WideResNet as described in section 5.1 on the data from five classes of the CIFAR-10 dataset, and then analyze predictions of SWAG variants along with the baselines on the full test set. We expect the outputted class probabilities on objects that belong to classes that were not present in the training data to have high-entropy reflecting the model's high uncertainty in its predictions, and considerably lower entropy on the images that are similar to those on which the network was trained. We plot the histograms of predictive entropies on the in-domain and out-of-domain in Figure A.10 for a qualitative comparison and report the symmetrized KL divergence between the binned in and out of sample distributions in Table 2, finding that SWAG and Dropout perform best on this measure. Additional details are in Appendix E.2.

5.4 Language Modeling with LSTMs

We next apply SWAG to an LSTM network on language modeling tasks on Penn Treebank and WikiText-2 datasets. In Appendix F we demonstrate that SWAG easily outperforms both SWA and NT-ASGD [44], a strong baseline for LSTM training, in terms of test and validation perplexities.

We compare SWAG to SWA and the NT-ASGD method [44], which is a strong baseline for training LSTM models. The main difference between SWA and NT-ASGD, which is also based on weight averaging, is that NT-ASGD starts weight averaging much earlier than SWA: NT-ASGD switches to ASGD (averaged SGD) typically around epoch 100 while with SWA we start averaging after pre-training for 500 epochs. We report test and validation perplexities for different methods and datasets in Table 1.

As we can see, SWA substantially improves perplexities on both datasets over NT-ASGD. Further, we observe that SWAG is able to substantially improve test perplexities over the SWA solution.

Table 1: Validation and Test perplexities for NT-ASGD, SWA and SWAG on Penn Treebank and WikiText-2 datasets.

Method	PTB val	PTB test	WikiText-2 val	WikiText-2 test
NT-ASGD	61.2	58.8	68.7	65.6
SWA	59.1	56.7	68.1	65.0
SWAG	**58.6**	**56.26**	**67.2**	**64.1**

5.5 Regression

Finally, while the empirical focus of our paper is classification calibration, we also compare to additional approximate BNN inference methods which perform well on smaller architectures, including deterministic variational inference (DVI) [60], single-layer deep GPs (DGP) with expectation propagation [7], SGLD [59], and re-parameterization VI [33] on a set of UCI regression tasks. We report test log-likelihoods, RMSEs and test calibration results in Appendix Tables 12 and 13 where it is possible to see that SWAG is competitive with these methods. Additional details are in Appendix G.

6 Discussion

In this paper we developed SWA-Gaussian (SWAG) for approximate Bayesian inference in deep learning. There has been a great desire to apply Bayesian methods in deep learning due to their theoretical properties and past success with small neural networks. We view SWAG as a step towards practical, scalable, and accurate Bayesian deep learning for large modern neural networks.

A key geometric observation in this paper is that the posterior distribution over neural network parameters is close to Gaussian in the subspace spanned by the trajectory of SGD. Our work shows Bayesian model averaging within this subspace can improve predictions over SGD or SWA solutions. Furthermore, Gur-Ari et al. [20] argue that the SGD trajectory lies in the subspace spanned by the eigenvectors of the Hessian corresponding to the top eigenvalues, implying that the SGD trajectory

subspace corresponds to directions of rapid change in predictions. In recent work, Izmailov et al. [26] show promising results from directly constructing subspaces for Bayesian inference.

Acknowledgements

WM, PI, and AGW were supported by an Amazon Research Award, Facebook Research, NSF IIS-1563887, and NSF IIS-1910266. WM was additionally supported by an NSF Graduate Research Fellowship under Grant No. DGE-1650441. DV was supported by the Russian Science Foundation grant no.19-71-30020. We would like to thank Jacob Gardner, Polina Kirichenko, and David Widmann for helpful discussions.

References

[1] Asmussen, S. and Glynn, P. W. (2007). *Stochastic simulation: algorithms and analysis*. Number 57 in Stochastic modelling and applied probability. Springer, New York. OCLC: ocn123113652.

[2] Athiwaratkun, B., Finzi, M., Izmailov, P., and Wilson, A. G. (2019). There are many consistent explanations for unlabeled data: why you should average. In *International Conference on Learning Representations*. arXiv: 1806.05594.

[3] Babichev, D. and Bach, F. (2018). Constant step size stochastic gradient descent for probabilistic modeling. In *Uncertainty in Artificial Intelligence*. arXiv preprint arXiv:1804.05567.

[4] Berger, J. O. (2013). *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media.

[5] Blier, L. and Ollivier, Y. (2018). The Description Length of Deep Learning models. In *Advances in Neural Information Processing Systems*, page 11.

[6] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight Uncertainty in Neural Networks. In *International Conference on Machine Learning*. arXiv: 1505.05424.

[7] Bui, T., Hernández-Lobato, D., Hernandez-Lobato, J., Li, Y., and Turner, R. (2016). Deep gaussian processes for regression using approximate expectation propagation. In *International Conference on Machine Learning*, pages 1472–1481.

[8] Chaudhari, P. and Soatto, S. (2018). Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *International Conference on Learning Representations*. arXiv: 1710.11029.

[9] Chen, T., Fox, E. B., and Guestrin, C. (2014). Stochastic Gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*. arXiv: 1402.4102.

[10] Chen, X., Lee, J. D., Tong, X. T., and Zhang, Y. (2016). Statistical Inference for Model Parameters in Stochastic Gradient Descent. arXiv: 1610.08637.

[11] Coates, A., Ng, A., and Lee, H. (2011). An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 215–223.

[12] Dasgupta, S. and Hsu, D. (2007). On-Line Estimation with the Multivariate Gaussian Distribution. In Bshouty, N. H. and Gentile, C., editors, *Twentieth Annual Conference on Learning Theory.*, volume 4539, pages 278–292, Berlin, Heidelberg. Springer Berlin Heidelberg.

[13] Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. A. (2018). Essentially No Barriers in Neural Network Energy Landscape. In *International Conference on Machine Learning*, page 10.

[14] Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian Approximation. In *International Conference on Machine Learning*.

[15] Gal, Y., Hron, J., and Kendall, A. (2017). Concrete Dropout. In *Advances in Neural Information Processing Systems*. arXiv: 1705.07832.

[16] Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. (2018). Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, pages 7587–7597.

[17] Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. (2018). Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, pages 8789–8798.

[18] Graves, A. (2011). Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356.

[19] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On Calibration of Modern Neural Networks. In *International Conference on Machine Learning*. arXiv: 1706.04599.

[20] Gur-Ari, G., Roberts, D. A., and Dyer, E. (2019). Gradient descent happens in a tiny subspace.

[21] Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

[22] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *CVPR*. arXiv: 1512.03385.

[23] Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *Advances in Neural Information Processing Systems*.

[24] Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. In *CVPR*. arXiv: 1608.06993.

[25] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

[26] Izmailov, P., Maddox, W. J., Kirichenko, P., Garipov, T., Vetrov, D., and Wilson, A. G. (2019). Subspace inference for bayesian deep learning. *arXiv preprint arXiv:1907.07504*.

[27] Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. (2018). Averaging weights leads to wider optima and better generalization. *Uncertainty in Artificial Intelligence (UAI)*.

[28] Kendall, A. and Gal, Y. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In *Advances in Neural Information Processing Systems*, Long Beach.

[29] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2017). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *International Conference on Learning Representations*. arXiv: 1609.04836.

[30] Khan, M. E., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. (2018). Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam. In *International Conference on Machine Learning*. arXiv: 1806.04854.

[31] Kingma, D. P., Salimans, T., and Welling, M. (2015a). Variational Dropout and the Local Reparameterization Trick. *arXiv:1506.02557 [cs, stat]*. arXiv: 1506.02557.

[32] Kingma, D. P., Salimans, T., and Welling, M. (2015b). Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583.

[33] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. In *International Conference on Learning Representations*.

[34] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, page 201611835.

[35] Kuleshov, V., Fenner, N., and Ermon, S. (2018). Accurate Uncertainties for Deep Learning Using Calibrated Regression. In *International Conference on Machine Learning*, page 9.

[36] Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems*.

[37] Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018). Visualizing the Loss Landscape of Neural Nets. In *Advances in Neural Information Processing Systems*. arXiv: 1712.09913.

[38] Loshchilov, I. and Hutter, F. (2019). Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*. arXiv: 1711.05101.

[39] Louizos, C. and Welling, M. (2017). Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*.

[40] MacKay, D. J. C. (1992a). Bayesian Interpolation. *Neural Computation*.

[41] MacKay, D. J. C. (1992b). A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472.

[42] MacKay, D. J. C. (2003). *Information theory, inference, and learning algorithms*. Cambridge University Press, Cambridge, UK ; New York.

[43] Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic Gradient Descent as Approximate Bayesian Inference. *JMLR*, 18:1–35.

[44] Merity, S., Keskar, N. S., and Socher, R. (2017). Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.

[45] Molchanov, D., Ashukha, A., and Vetrov, D. (2017). Variational dropout sparsifies deep neural networks. *arXiv preprint arXiv:1701.05369*.

[46] Mukhoti, J. and Gal, Y. (2018). Evaluating Bayesian Deep Learning Methods for Semantic Segmentation.

[47] Müller, U. K. (2013). Risk of bayesian inference in misspecified models, and the sandwich covariance matrix. *Econometrica*, 81(5):1805–1849.

[48] Naeini, M. P., Cooper, G. F., and Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, pages 2901–2907.

[49] Neal, R. M. (1996). *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer New York, New York, NY.

[50] Niculescu-Mizil, A. and Caruana, R. (2005). Predicting good probabilities with supervised learning. In *International Conference on Machine Learning*, pages 625–632, Bonn, Germany. ACM Press.

[51] Nikishin, E., Izmailov, P., Athiwaratkun, B., Podoprikhin, D., Garipov, T., Shvechikov, P., Vetrov, D., and Wilson, A. G. (2018). Improving stability in deep reinforcement learning with weight averaging.

[52] Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of Stochastic Approximation by Averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855.

[53] Ritter, H., Botev, A., and Barber, D. (2018a). Online Structured Laplace Approximations For Overcoming Catastrophic Forgetting. In *Advances in Neural Information Processing Systems*. arXiv: 1805.07810.

[54] Ritter, H., Botev, A., and Barber, D. (2018b). A Scalable Laplace Approximation for Neural Networks. In *International Conference on Learning Representations*.

[55] Ruppert, D. (1988). Efficient Estimators from a Slowly Convergent Robbins-Munro Process. Technical Report 781, Cornell University, School of Operations Report and Industrial Engineering.

[56] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252. arXiv: 1409.0575.

[57] Sagun, L., Evci, U., Guney, V. U., Dauphin, Y., and Bottou, L. (2018). Empirical Analysis of the Hessian of Over-Parametrized Neural Networks. In *International Conference on Learning Representations Workshop Track*. arXiv: 1706.04454.

[58] Vaart, A. W. v. d. (1998). *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge.

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

[60] Wu, A., Nowozin, S., Meeds, E., Turner, R. E., Hernández-Lobato, J. M., and Gaunt, A. L. (2019). Fixing variational bayes: Deterministic variational inference for bayesian neural networks. In *Inernational Conference on Learning Representations*. arXiv preprint arXiv:1810.03958.

[61] Yang, G., Zhang, T., Kirichenko, P., Bai, J., Wilson, A. G., and De Sa, C. (2019). Swalp: Stochastic weight averaging in low precision training. In *International Conference on Machine Learning*, pages 7015–7024.

[62] Yazici, Y., Foo, C.-S., Winkler, S., Yap, K.-H., Piliouras, G., and Chandrasekhar, V. (2019). The Unusual Effectiveness of Averaging in GAN Training. In *International Conference on Learning Representations*. arXiv: 1806.04498.

[63] Zagoruyko, S. and Komodakis, N. (2016). Wide Residual Networks. In *BMVC*. arXiv: 1605.07146.

[64] Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. (2017). Noisy Natural Gradient as Variational Inference. *arXiv:1712.02390 [cs, stat]*. arXiv: 1712.02390.

[65] Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. (2019). Cyclical stochastic gradient mcmc for bayesian deep learning. *arXiv preprint arXiv:1902.03932*.

A Asymptotic Normality of SGD

Under conditions of decaying learning rates, smoothness of gradients, and the existence of a full rank stationary distribution, martingale based analyses of stochastic gradient descent [e.g., 1, Chapter 8] show that SGD has a Gaussian limiting distribution. That is, in the limit as the time step goes to infinity, $t^{1/2}(\theta_t - \theta^*) \to \mathcal{N}(0, \mathcal{H}(\theta)^{-1}\mathbb{E}(\nabla \log p(\theta) \nabla \log p(\theta)^T)\mathcal{H}(\theta)^{-1}))$, where $\mathcal{H}(\theta)^{-1}$ is the inverse of the Hessian matrix of the log-likelihood and $\mathbb{E}(\nabla \log p(\theta) \nabla \log p(\theta)^T)$ is the covariance of the gradients and θ^* is a stationary point or minima. Note that these analyses date back to Ruppert [55] and Polyak and Juditsky [52] for Polyak-Ruppert averaging, and are still popular in the analysis of stochastic gradient descent.

Mandt et al. [43], Chen et al. [10], and Babichev and Bach [3] all use the same style of analyses, but for different purposes. We will test the specific assumptions of Mandt et al. [43] in the next section. Finally, note that the technical conditions are essentially the same conditions as for the Bernstein von Mises Theorem [e.g., 58, Chapter 10] which implies that the asymptotic posterior will also be Gaussian.

It may be counter-intuitive that, as we show in Section 4, SWAG captures the geometry of the objective correctly. One might even expect SWAG estimates of variance to be inverted, as gradient descent would oscillate more in the sharp directions of the objective. To gain more intuition about SGD dynamics we visualize SGD trajectory on a quadratic problem. More precisely, we define a 2-dimensional quadratic function $f(x, y) = (x + y)^2 + 0.05 \cdot (x - y)^2$ shown in Figure 4. We then run SGD to minimize this function.

It turns out that the gradient noise plays a crucial role in forming the SGD stationary distribution. If there is no noise in the gradients, we are in the full gradient descent regime, and optimization either

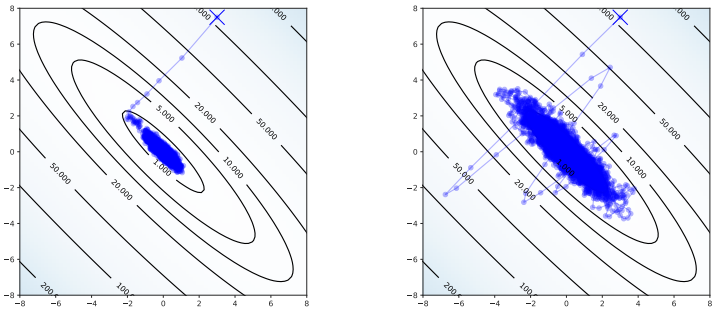


Figure 4: Trajectory of SGD with isotropic Gaussian gradient noise on a quadratic loss function. **Left**: SGD without momentum; **Right**: SGD with momentum.

converges to the optimizer, or diverges to infinity depending on the learning rate. However, when we add isotropic Gaussian noise to the gradients, SGD converges to the correct Gaussian distribution, as we visualize in the left panel of Figure 4. Furthermore, adding momentum affects the scale of the distribution, but not its shape, as we show in the right panel of Figure 4. These conclusions hold as long as the learning rate in SGD is not too large.

The results we show in Figure 4 are directly predicted by theory in Mandt et al. [43]. In general, if the gradient noise is not isotropic, the stationary distribution of SGD would be different from the exact posterior distribution. Mandt et al. [43] provide a thorough empirical study of the SGD trajectory for convex problems, such as linear and logistic regression, and show that SGD can often provide a competitive baseline on these problems.

A.1 Other Considerations for Gaussian Approximation

Given the covariance matrix $A = \mathcal{H}(\theta)^{-1}\mathbb{E}(\nabla \log p(\theta) \nabla \log p(\theta)^T)\mathcal{H}(\theta)^{-1}$, Chen et al. [10] show that a batch means estimator of the iterates (similar to what SWAG uses) themselves will converge to A in the limit of infinite time. We tried batch means based estimators but saw no improvement; however, it could be interesting to explore further in future work.

Intriguingly, the covariance A is the same form as sandwich estimators [see e.g. 47, for a Bayesian analysis in the model mis-specification setting], and so $A = \mathcal{H}(\theta)^{-1}$ under model well-specification [47, 10]. We then tie the covariance matrix of the iterates back to the well known Laplace approximation, which uses $\mathcal{H}(\theta)^{-1}$ as its covariance as described by MacKay [42, Chapter 28], thereby justifying SWAG theoretically as a sample based Laplace approximation.

Finally, in Chapter 4 of Berger [4] constructs an example (Example 10) of fitting a Gaussian approximation from a MCMC chain, arguing that it empirically performs well in Bayesian decision theoretic contexts. Berger [4] give the explanation for this as the Bernstein von Mises Theorem providing that in the limit the posterior will itself converge to a Gaussian. However, we would expect that even in the infinite data limit the posterior of DNNs would converge to something very non-Gaussian, with connected modes surrounded by gorges of zero posterior density [17]. One could use this justification for fitting a Gaussian from the iterates of SGLD or SGHMC instead.

B Do the assumptions of Mandt et al. [43] hold for DNNs?

In this section, we investigate the results of Mandt et al. [43] in the context of deep learning. Mandt et al. [43] uses the following assumptions:

1. Gradient noise at each point θ is $\mathcal{N}(0, C)$.
2. C is independent of θ and full rank.
3. The learning rates, η, are small enough that we can approximate the dynamics of SGD by a continuous-time dynamic described by the corresponding stochastic differential equation.

4. In the stationary distribution, the loss is approximately quadratic near the optima, i.e. approximately $(\theta - \theta^*)^\top \mathcal{H}(\theta)(\theta - \theta^*)$, where $\mathcal{H}(\theta^*)$ is the Hessian at the optimum; further, the Hessian is assumed to be positive definite.

Assumption 1 is motivated by the central limit theorem, and Assumption 3 is necessary for the analysis in Mandt et al. [43]. Assumptions 2 and 4 may or may not hold for deep neural networks (as well as other models). Under these assumptions, Theorem 1 of Mandt et al. [43] derives the optimal constant learning rate that minimizes the KL-divergence between the SGD stationary distribution and the posterior[6]:

$$\eta^* = 2\frac{B}{N}\frac{d}{tr(C)}, \tag{2}$$

where N is the size of the dataset, d is the dimension of the model, B is the minibatch size and C is the gradient noise covariance.

We computed Equation 2 over the course of training for two neural networks in Figure A.5a, finding that the predicted optimal learning rate was an order of magnitude larger than what would be used in practice to explore the loss surface in a reasonable time (about 4 compared to 0.1).

We now focus on seeing how Assumptions 2 and 4 fail for DNNs; this will give further insight into what portions of the theory do hold, and may give insights into a corrected version of the optimal learning rate.

B.1 Assumption 2: Gradient Covariance Noise.

In Figure A.5b, the trace of the gradient noise covariance and thus the optimal learning rates *are* nearly constant; however, the total variance is much too small to induce effective learning rates, probably due to over-parameterization effects inducing non full rank gradient covariances as was found in Chaudhari and Soatto [8]. We note that this experiment is not sufficient to be fully confident that C is independent of the parameterization near the local optima, but rather that $tr(C)$ is close to constant; further experiments in this vein are necessary to test if the diagonals of C are constant. The result that $tr(C)$ is close to constant suggests that a constant learning rate could be used for sampling in a stationary phase of training. The dimensionality parameter in Equation 2 could be modified to use the number of effective parameters or the rank of the gradient noise to reduce the optimal learning rate to a feasible number.

To estimate $tr(C)$ from the gradient noise we need to divide the estimated variance by the batch size (as $V(\hat{g}(\theta)) = BC(\theta)$), for a correct version of Equation 2. From Assumption 1 and Equation 6 of Mandt et al. [43], we see that

$$\hat{g}(\theta) \approx g(\theta) + \frac{1}{\sqrt{B}}\nabla g(\theta), \nabla g(\theta) \sim N(0, C(\theta)),$$

where B is the batch size. Thus, collecting the variance of $\hat{g}(\theta)$ (the variance of the stochastic gradients) will give estimates that are upscaled by a factor of B, leading to a cancellation of the batch size terms:

$$\eta \approx \frac{2}{N}\frac{d}{tr(V(\hat{g}(\theta)))}.$$

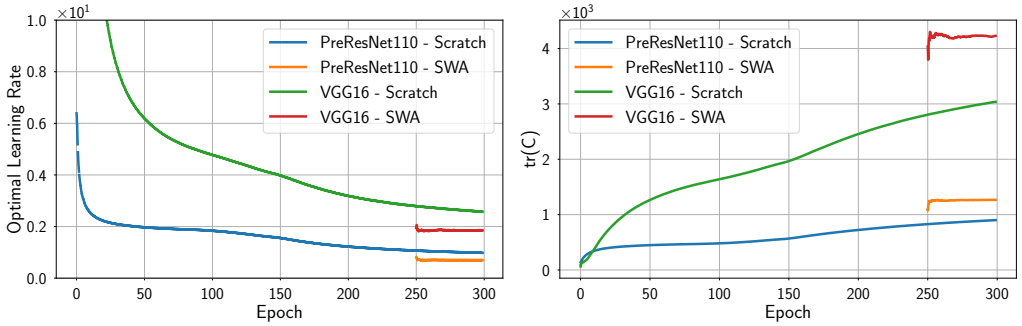
To include momentum, we can repeat the analysis in Sections 4.1 and 4.3 of Mandt et al. [43] finding that this also involves scaling the optimal learning rate but by a factor of μ, the momentum term.[7] This gives the final optimal learning rate equation as

$$\eta \approx \frac{2\mu}{N}\frac{d}{tr(V(\hat{g}(\theta)))}. \tag{3}$$

In Figure 5b, we computed $tr(C)$ for VGG-16 and PreResNet-164 on CIFAR-100 beginning from the start of training (referred to as from scratch), as well as the start of the SWAG procedure (referred to in the legend as SWA). We see that $tr(C)$ is never quite constant when trained from scratch, while for a period of constant learning rate near the end of training, referred to as the stationary phase,

[6] An optimal diagonal preconditioner is also derived; our empirical work applies to that setting as well. A similar analysis with momentum holds as well, adding in only the momentum coefficient.

[7] Our experiments used $\mu = 0.1$ corresponding to $\rho = 0.9$ in PyTorch's SGD implementation.



(a) Optimal learning rate. (b) $tr(C)$

Figure 5: Gradient variance norm and computed optimal learning rates for VGG-16 and PreResNet-164. The computed optimal learning rates are always too large by a factor of 10, while the gradient variance stabilizes over the course of training.

$tr(C)$ is essentially constant throughout. This discrepancy is likely due to large gradients at the very beginning of training, indicating that the stationary distribution has not been reached yet.

Next, in Figure 5a, we used the computed $tr(C)$ estimate for all four models and Equation 3 to compute the optimal learning rate under the assumptions of Mandt et al. [43], finding that these learning rates are not constant for the estimates beginning at the start of training and that they are too large (1-3 at the minimum compared to a standard learning rate of 0.1 or 0.01).

B.2 Assumption 4: Hessian Eigenvalues at the Optima

To test assumption 4, we used a GPU-enabled Lanczos method from GPyTorch [16] and used restarting to compute the minimum eigenvalue of the train loss of a pre-trained PreResNet-164 on CIFAR-100. We found that even at the end of training, the minimum eigenvalue was -272 (the maximum eigenvalue was 3580 for comparison), indicating that the Hessian is not positive definite. This result harmonizes with other work analyzing the spectra of the Hessian for DNN training [37, 57]. Further, Garipov et al. [17] and Draxler et al. [13] argue that the loss surfaces of DNNs have directions along which the loss is completely flat, suggesting that the loss is nowhere near a positive-definite quadratic form.

C Further Geometric Experiments

In Figure 6 we present plots analogous to those in Section 4 for PreResNet-110 and VGG-16 on CIFAR-10 and CIFAR-100. For all dataset-architecture pairs we see that SWAG is able to capture the geometry of the posterior in the subspace spanned by SGD trajectory.

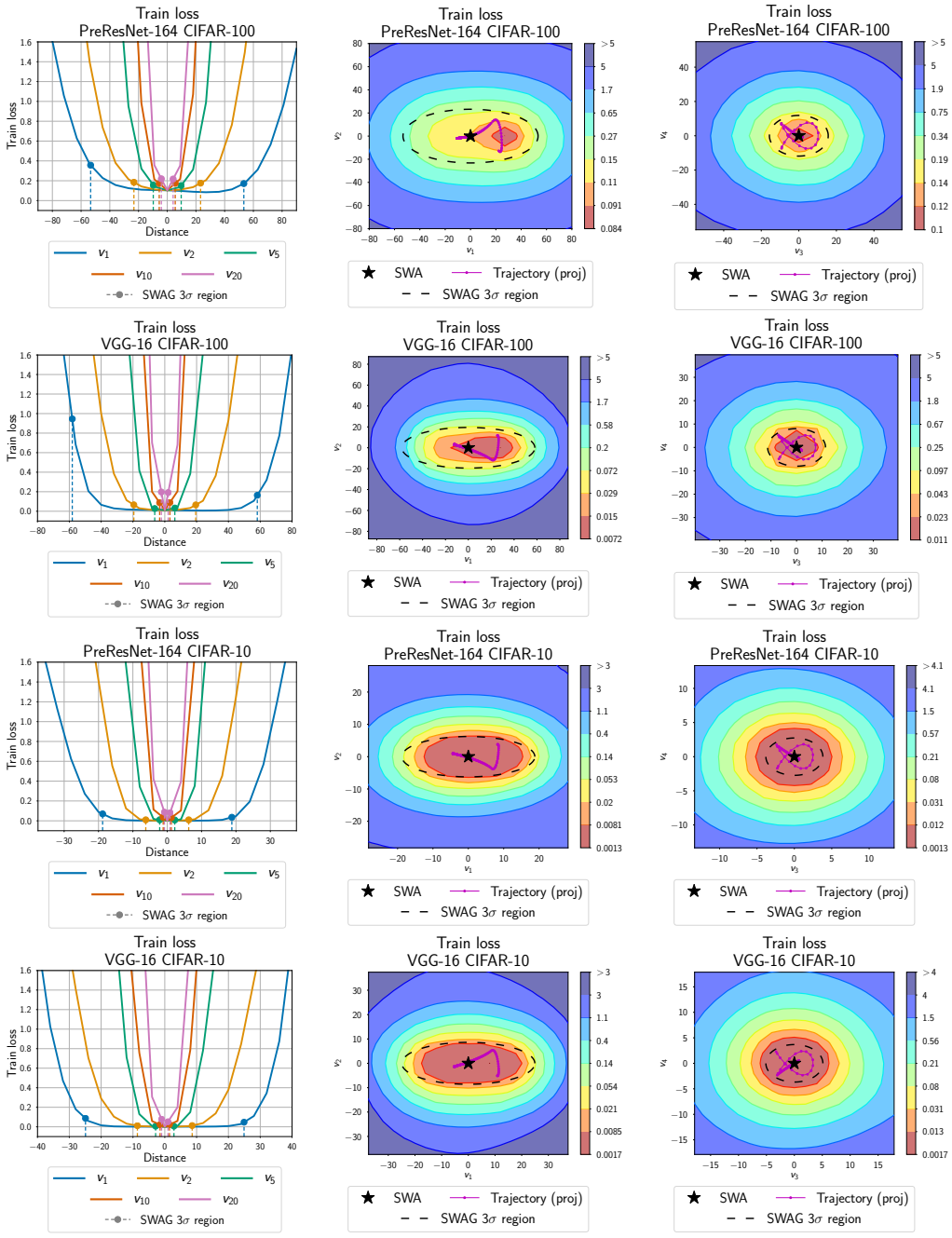
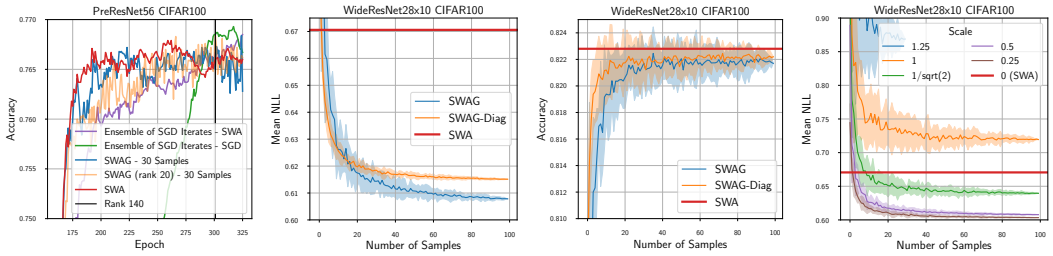


Figure 6: **Left:** Posterior-density cross-sections along the rays corresponding to different eigenvectors of the SWAG covariance matrix. **Middle:** Posterior-density surface in the plane spanned by eigenvectors of SWAG covariance matrix corresponding to the first and second largest eigenvalues and (**Right:**) the third and fourth largest eigenvalues. Each row in the figure corresponds to an architecture-dataset pair indicated in the title of each panel.



(a) (b) (c) (d)

Figure 7: **(a)** 30 samples of SWAG with a rank 20 covariance matches the SWA result over the course of training for PreResNet56 on CIFAR-100. SWAG with a rank of 140, SWAG with a rank of 20, and SWA all outperform ensembles of SGD iterates from the SWA procedure and from a standard SGD training path. **(b)** NLL and **(c)** accuracy by number of samples for WideResNet on CIFAR-100 for SWAG, SWAG-Diag, and SWA. 30 samples is adequate for stable accuracies and NLLs. **(d)** NLL by number of samples for different scales for WideResNet on CIFAR-100 for SWAG, SWAG-Diag, and SWA. Scales beneath 1 perform better, with 0.5 and 0.25 best.

D Hyper-Parameters and Limitations

In this section, we discuss the hyper-parameters in SWAG, as well as some current theoretical limitations.

D.1 Rank of Covariance Matrix

We now evaluate the effect of the covariance matrix rank on the SWAG approximation. To do so, we trained a PreResNet56 on CIFAR-100 with SWAG beginning from epoch 161, and evaluated 30 sample Bayesian model averages obtained at different epochs; the accuracy plot from this experiment is shown in Figure 7 (a). The rank of each model after epoch 161 is simply $\min(epoch - 161, 140)$, and 30 samples from even a low rank approximation reach the same predictive accuracy as the SWA model. Interestingly, both SWAG and SWA outperform ensembles of a SGD run and ensembles of the SGD models in the SWA run.

D.2 Number of Samples in the Forwards Pass

In most situations where SWAG will be used, no closed form expression for the integral $\int f(y)q(\theta|y)d\theta$, will exist. Thus, Monte Carlo approximations will be used; Monte Carlo integration converges at a rate of $1/\sqrt{K}$, where K is the number of samples used, but practically good results may be found with very few samples (e.g. Chapter 29 of MacKay [42]).

To test how many samples are needed for good predictive accuracy in a Bayesian model averaging task, we used a rank 20 approximation for SWAG and then tested the NLL on the test set as a function of the number of samples for WideResNet28x10 [63] on CIFAR-100.

The results from this experiment are shown in Figure 7 (b, c), where it is possible to see that about 3 samples will match the SWA result for NLL, with about 30 samples necessary for stable accuracy (about the same as SWA for this network). In most of our experiments, we used 30 samples for consistency. In practice, we suggest tuning this number by looking at a validation set as well as the computational resources available and comparing to the free SWA predictions that come with SWAG.

D.3 Dependence on Learning Rate

First, we note that the covariance, Σ, estimated using SWAG, is a function of the learning rate (and momentum) for SGD. While the theoretical work of Mandt et al. [43] suggests that it is possible to optimally set the learning rate, our experiments in Appendix B show that currently the assumptions of the theory do not match the empirical reality in deep learning. In practice the learning rate can be chosen to maximize negative log-likelihood on a validation set. In the linear setting as in Mandt et al. [43], the learning rate controls the scale of the asymptotic covariance matrix. If the optimal

learning rate (Equation 2) is used in this setting, the covariance matches the true posterior. To attempt to disassociate the learning rate from the covariance in practice, we rescale the covariance matrix when sampling by a constant factor for a WideResNet on CIFAR-100 shown in Figure 7 (d).

Over several replications, we found that a scale of 0.5 worked best, which is expected because the low rank plus diagonal covariance incorporates the variance twice (once for the diagonal and once from the low rank component).

D.4 Necessity of Batch Norm Updates

One possible slowdown of SWAG at inference time is in the usage of updated batch norm parameters. Following Izmailov et al. [27], we found that in order for the averaging and sampling to work well, it was necessary to update the batch norm parameters of networks after sampling a new model. This is shown in Figure 8 for a WideResNet on CIFAR-100 for two independently trained models.

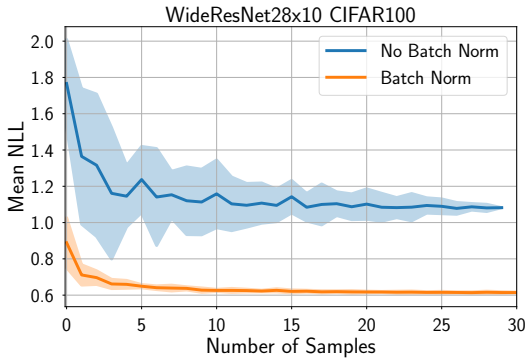


Figure 8: NLL by number of samples for SWAG with and without batch norm updates after sampling. Updating the batch norm parameters after sampling results in a significant improvement in NLL.

D.5 Usage in Practice

From our experimental findings, we see that given an equal amount of training time, SWAG typically outperforms other methods for uncertainty calibration. SWAG additionally does not require a validation set like temperature scaling and Platt scaling (e.g. Guo et al. [19], Kuleshov et al. [35]). SWAG also appears to have a distinct advantage over temperature scaling, and other popular alternatives, when the target data are from a different distribution than the training data, as shown by our transfer learning experiments.

Deep ensembles [36] require several times longer training for equal calibration, but often perform somewhat better due to incorporating several independent training runs. Thus SWAG will be particularly valuable when training time is limited, but inference time may not be. One possible application is thus in medical applications when image sizes (for semantic segmentation) are large, but predictions can be parallelized and may not have to be instantaneous.

E Further Classification Uncertainty Results

E.1 Reliability Diagrams

We provide the additional reliability diagrams for all methods and datasets in Figure 9. SWAG consistently improves calibration over SWA, and performs on par or better than temperature scaling. In transfer learning temperature scaling fails to achieve good calibration, while SWAG still provides a significant improvement over SWA.

E.2 Out-of-Domain Image Detection

Next, we evaluate the SWAG variants along with the baselines on out-of-domain data detection. To do so we train a WideResNet as described in Section H on the data from five classes of the

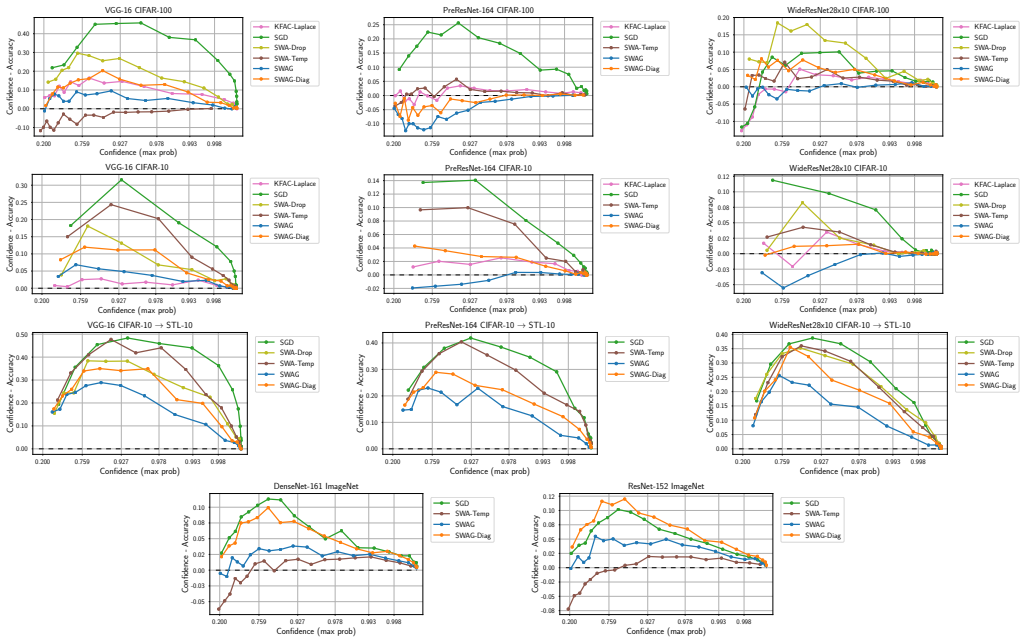


Figure 9: Reliability diagrams (see Section 5.1) for all models and datasets. The dataset and architecture are listed in the title of each panel.

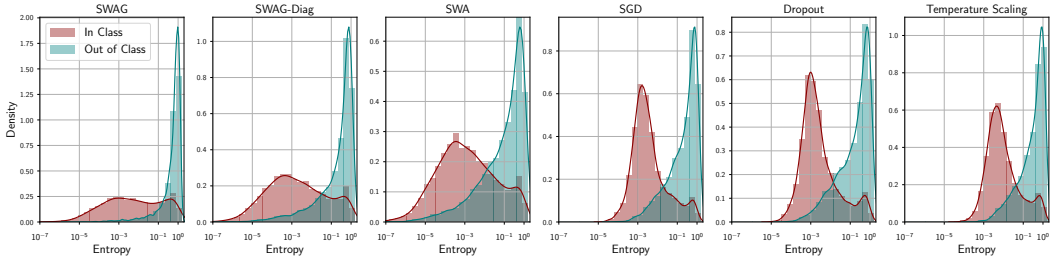


Figure 10: In and out of sample entropy distributions for WideResNet28x10 on CIFAR5 + 5.

Table 2: Symmetrized, discretized KL divergence between the distributions of predictive entropies for data from the first and last five classes of CIFAR-10 for models trained only on the first five classes. The entropy distributions for SWAG are more different than the baseline models.

Method	JS-Distance
SWAG	**3.31**
SWAG-Diag	2.27
MC Dropout	3.04
SWA	1.68
SGD (Baseline)	3.14
SGD + Temp. Scaling	2.98

CIFAR-10 dataset, and then analyze their predictions on the full test set. We expect the outputted class probabilities on objects that belong to classes that were not present in the training data to have high-entropy reflecting the model's high uncertainty in its predictions, and considerably lower entropy on the images that are similar to those on which the network was trained.

To make this comparison quantitative, we computed the symmetrized KL divergence between the binned in and out of sample distributions in Table 2, finding that SWAG and Dropout perform best on this measure. We plot the histograms of predictive entropies on the in-domain (classes that were

Table 3: ECE for various versions of SWAG, temperature scaling, and MC Dropout on CIFAR-10 and CIFAR-100.

	CIFAR-10	CIFAR-10	CIFAR-10	CIFAR-100	CIFAR-100	CIFAR-100
Model	VGG-16	PreResNet-164	WideResNet28x10	VGG-16	PreResNet-164	WideResNet28x10
SGD	0.0483 ± 0.0022	0.0255 ± 0.0009	0.0166 ± 0.0007	0.1870 ± 0.0014	0.1012 ± 0.0009	0.0479 ± 0.0010
SWA	0.0408 ± 0.0019	0.0203 ± 0.0010	0.0087 ± 0.0002	0.1514 ± 0.0032	0.0700 ± 0.0056	0.0684 ± 0.0022
SWAG-Diag	0.0267 ± 0.0025	0.0082 ± 0.0008	$\mathbf{0.0047} \pm 0.0013$	0.0819 ± 0.0021	0.0239 ± 0.0047	0.0322 ± 0.0018
SWAG	0.0158 ± 0.0030	$\mathbf{0.0053} \pm 0.0004$	0.0088 ± 0.0006	0.0395 ± 0.0061	0.0587 ± 0.0048	$\mathbf{0.0113} \pm 0.0020$
KFAC-Laplace	0.0094 ± 0.0005	0.0092 ± 0.0018	0.0060 ± 0.0003	0.0778 ± 0.0054	$\mathbf{0.0158} \pm 0.0014$	0.0379 ± 0.0047
SWA-Dropout	0.0284 ± 0.0036	0.0162 ± 0.0000	0.0094 ± 0.0014	0.1108 ± 0.0181	*	0.0574 ± 0.0028
SWA-Temp	0.0366 ± 0.0063	0.0172 ± 0.0010	0.0080 ± 0.0007	$\mathbf{0.0291} \pm 0.0097$	0.0175 ± 0.0037	0.0220 ± 0.0007
SGLD	$\mathbf{0.0082} \pm 0.0012$	0.0251 ± 0.0012	0.0192 ± 0.0007	0.0424 ± 0.0029	0.0363 ± 0.0008	0.0296 ± 0.0008

Table 4: ECE on ImageNet.

Model	DenseNet-161	ResNet-152
SGD	0.0545 ± 0.0000	0.0478 ± 0.0000
SWA	0.0509 ± 0.0000	0.0605 ± 0.0000
SWAG-Diag	0.0459 ± 0.0000	0.0566 ± 0.0000
SWAG	0.0204 ± 0.0000	0.0279 ± 0.0000
SWA-Temp	$\mathbf{0.0190} \pm 0.0000$	$\mathbf{0.0183} \pm 0.0000$

Table 5: ECE on CIFAR10 to STL 10.

Model	VGG-16	PreResNet-164	WideResNet28x10
SGD	0.2149 ± 0.0027	0.1758 ± 0.0000	0.1561 ± 0.0000
SWA	0.2082 ± 0.0056	0.1739 ± 0.0000	0.1413 ± 0.0000
SWAG-Diag	0.1719 ± 0.0075	0.1312 ± 0.0000	0.1241 ± 0.0000
SWAG	$\mathbf{0.1463} \pm 0.0075$	$\mathbf{0.1110} \pm 0.0000$	$\mathbf{0.1017} \pm 0.0000$
SWA-Dropout	0.1803 ± 0.0024		0.1421 ± 0.0000
SWA-Temp	0.2089 ± 0.0055	0.1646 ± 0.0000	0.1371 ± 0.0000

trained on) and out-of-domain (classes that were not trained on) in Figure A.10 for a qualitative comparison.

Table 2 shows the computed symmetrized, discretized KL distance between in and out of sample distributions for the CIFAR5 out of sample image detection class. We used the same bins as in Figure 10 to discretize the entropy distributions, then smoothed these bins by a factor of 1e-7 before calculating $KL(\text{IN}||\text{OUT}) + KL(\text{OUT}||\text{IN})$ using the `scipy.stats.entropy` function. We can see even qualitatively that the distributions are more distinct for SWAG and SWAG-Diagonal than for the other methods, particularly temperature scaling.

E.3 Tables of ECE, NLL, and Accuracy.

We provide test accuracies (Tables 9,10,11) and negative log-likelihoods (NLL) (Tables 6,7,8) all methods and datasets. We observe that SWAG is competitive with SWA, SWA with temperature scaling and SWA-Dropout in terms of test accuracy, and typically outperforms all the baselines in terms of NLL. SWAG-Diagonal is generally inferior to SWAG for log-likelihood, but outperforms SWA.

In Tables 3,4,5 we additionally report expected calibration error [ECE, 48], a metric of calibration of the predictive uncertainties. To compute ECE for a given model we split the test points into 20 bins based on the confidence of the model, and we compute the absolute value of the difference of the average confidence and accuracy within each bin, and average the obtained values over all bins. Please refer to [48, 19] for more details. We observe that SWAG is competitive with temperature scaling for ECE. Again, SWAG-Diagonal achieves better calibration than SWA, but using the low-rank plus diagonal covariance approximation in SWAG leads to substantially improved performance.

Table 6: NLL on CIFAR10 and CIFAR100.

Dataset	CIFAR-10			CIFAR-100		
Model	VGG-16	PreResNet-164	WideResNet28x10	VGG-16	PreResNet-164	WideResNet28x10
SGD	0.3285 ± 0.0139	0.1814 ± 0.0025	0.1294 ± 0.0022	1.7308 ± 0.0137	0.9465 ± 0.0191	0.7958 ± 0.0089
SWA	0.2621 ± 0.0104	0.1450 ± 0.0042	0.1075 ± 0.0004	1.2780 ± 0.0051	0.7370 ± 0.0265	0.6684 ± 0.0034
SWAG-Diag	0.2200 ± 0.0078	0.1251 ± 0.0029	0.1077 ± 0.0009	1.0163 ± 0.0032	0.6837 ± 0.0186	0.6150 ± 0.0029
SWAG	0.2016 ± 0.0031	$\mathbf{0.1232} \pm 0.0022$	0.1122 ± 0.0009	0.9480 ± 0.0038	$\mathbf{0.6595} \pm 0.0019$	$\mathbf{0.6078} \pm 0.0006$
KFAC-Laplace	0.2252 ± 0.0032	0.1471 ± 0.0012	0.1210 ± 0.0020	1.1915 ± 0.0199	0.7881 ± 0.0025	0.7692 ± 0.0092
SWA-Dropout	0.2328 ± 0.0049	0.1270 ± 0.0000	0.1094 ± 0.0021	1.1872 ± 0.0524		0.6500 ± 0.0049
SWA-Temp	0.2481 ± 0.0245	0.1347 ± 0.0038	$\mathbf{0.1064} \pm 0.0004$	1.0386 ± 0.0126	$\mathbf{0.6770} \pm 0.0191$	0.6134 ± 0.0023
SGLD	$\mathbf{0.2001} \pm 0.0059$	0.1418 ± 0.0005	0.1289 ± 0.0009	0.9699 ± 0.0057	0.6981 ± 0.0052	0.678 ± 0.0022
SGD-Ens	0.1881 ± 0.002	0.1312 ± 0.0023	0.1855 ± 0.0014	$\mathbf{0.8979} \pm 0.0065$	0.7839 ± 0.0046	0.7655 ± 0.0026

Table 7: NLL on ImageNet.

Model	DenseNet-161	ResNet-152
SGD	0.9094 ± 0.0000	0.8716 ± 0.0000
SWA	0.8655 ± 0.0000	0.8682 ± 0.0000
SWAG-Diag	0.8559 ± 0.0000	0.8584 ± 0.0000
SWAG	$\mathbf{0.8303} \pm 0.0000$	$\mathbf{0.8205} \pm 0.0000$
SWA-Temp	0.8359 ± 0.0000	0.8226 ± 0.0000

Table 8: NLL when transferring from CIFAR10 to STL10.

Model	VGG-16	PreResNet-164	WideResNet28x10
SGD	1.6528 ± 0.0390	1.4790 ± 0.0000	1.1308 ± 0.0000
SWA	1.3993 ± 0.0502	1.3552 ± 0.0000	1.0047 ± 0.0000
SWAG-Diag	1.2258 ± 0.0446	1.0700 ± 0.0000	0.9340 ± 0.0000
SWAG	$\mathbf{1.1402} \pm 0.0342$	$\mathbf{0.9706} \pm 0.0000$	$\mathbf{0.8710} \pm 0.0000$
SWA-Dropout	1.3133 ± 0.0000		0.9914 ± 0.0000
SWA-Temp	1.4082 ± 0.0506	1.2228 ± 0.0000	0.9706 ± 0.0000

Table 9: Accuracy on CIFAR-10 and CIFAR-100.

Dataset	CIFAR-10			CIFAR-100		
Model	VGG-16	PreResNet-164	WideResNet28x10	VGG-16	PreResNet-164	WideResNet28x10
SGD	93.17 ± 0.14	95.49 ± 0.06	96.41 ± 0.10	73.15 ± 0.11	78.50 ± 0.32	80.76 ± 0.29
SWA	93.61 ± 0.11	96.09 ± 0.08	$\mathbf{96.46} \pm 0.04$	74.30 ± 0.22	$\mathbf{80.19} \pm 0.52$	82.40 ± 0.16
SWAG-Diag	$\mathbf{93.66} \pm 0.15$	96.03 ± 0.10	96.41 ± 0.05	74.68 ± 0.22	80.18 ± 0.50	$\mathbf{82.40} \pm 0.09$
SWAG	93.60 ± 0.10	96.03 ± 0.02	96.32 ± 0.08	$\mathbf{74.77} \pm 0.09$	79.90 ± 0.50	82.23 ± 0.19
KFAC-Laplace	92.65 ± 0.20	95.49 ± 0.06	96.17 ± 0.00	72.38 ± 0.23	78.51 ± 0.05	80.94 ± 0.41
SWA-Dropout	93.23 ± 0.36	$\mathbf{96.18} \pm 0.00$	96.39 ± 0.09	72.50 ± 0.54		82.30 ± 0.19
SWA-Temp	93.61 ± 0.11	96.09 ± 0.08	96.46 ± 0.04	74.30 ± 0.22	80.19 ± 0.52	82.40 ± 0.16
SGLD	93.55 ± 0.15	95.55 ± 0.04	95.89 ± 0.02	74.02 ± 0.30	80.09 ± 0.05	80.94 ± 0.17

Table 10: Accuracy on ImageNet.

Model	DenseNet-161	ResNet-152
SGD	77.79 ± 0.00	78.39 ± 0.00
SWA	$\mathbf{78.60} \pm 0.00$	78.92 ± 0.00
SWAG-Diag	78.59 ± 0.00	78.96 ± 0.00
SWAG	78.59 ± 0.00	$\mathbf{79.08} \pm 0.00$
SWA-Temp	78.60 ± 0.00	78.92 ± 0.00

F Language Modeling

We evaluate SWAG using standard Penn Treebank and WikiText-2 benchmark language modeling datasets. Following [44] we use a 3-layer LSTM model with 1150 units in the hidden layer and an embedding of size 400; we apply dropout, weight-tying, activation regularization (AR) and temporal

Table 11: Accuracy when transferring from CIFAR-10 to STL-10.

Model	VGG-16	PreResNet-164	WideResNet28x10
SGD	$\mathbf{72.42} \pm 0.07$	75.56 ± 0.00	76.75 ± 0.00
SWA	71.92 ± 0.01	$\mathbf{76.02} \pm 0.00$	$\mathbf{77.50} \pm 0.00$
SWAG-Diag	72.09 ± 0.04	75.95 ± 0.00	77.26 ± 0.00
SWAG	72.19 ± 0.06	75.88 ± 0.00	77.09 ± 0.00
SWA-Dropout	71.45 ± 0.11		76.91 ± 0.00
SWA-Temp	71.92 ± 0.01	76.02 ± 0.00	77.50 ± 0.00

Table 12: Unnormalized test log-likelihoods on small UCI datasets for proposed methods, as well as direct comparisons to the numbers reported in deterministic variational inference (DVI, Wu et al. [60]) and Deep Gaussian Processes with expectation propagation (DGP1-50, Bui et al. [7]), and variational inference (VI) with the re-parameterization trick [32]. * denotes reproduction from [60]. Note that SWAG wins on two of the six datasets, and that SGD serves as a strong baseline throughout.

dataset	N	D	SGD	SWAG	DVI*	DGP1-50*	VI*	SGLD*	PBP*
boston	506	13	-2.536 $\pm$ 0.240	-2.469 $\pm$ 0.183	-2.41 $\pm$ 0.02	**-2.33** $\pm$ 0.06	-2.43 $\pm$0.03	-2.40 $\pm$ 0.05	-2.57 $\pm$ 0.09
concrete	1030	8	**-3.02** $\pm$ 0.126	-3.05 $\pm$ 0.1	-3.06 $\pm$ 0.01	-3.13 $\pm$ 0.03	-3.04 $\pm$0.02	-3.08 $\pm$ 0.03	-3.16 $\pm$ 0.02
energy	768	8	-1.736 $\pm$ 1.613	-1.679 $\pm$ 1.488	**-1.01** $\pm$ 0.06	-1.32 $\pm$ 0.03	-2.38 $\pm$0.02	-2.39 $\pm$ 0.01	-2.04 $\pm$ 0.02
naval	11934	16	6.567 $\pm$ 0.185	**6.708** $\pm$ **0.105**	6.29 $\pm$ 0.04	3.60 $\pm$ 0.33	5.87 $\pm$0.29	3.33 $\pm$ 0.01	3.73 $\pm$ 0.01
yacht	308	6	-0.418 $\pm$ 0.426	**-0.404** $\pm$ **0.418**	-0.47 $\pm$ 0.03	-1.39 $\pm$ 0.14	-1.68 $\pm$0.04	-2.90 $\pm$ 0.01	-1.63 $\pm$ 0.02
power	9568	4	-2.772 $\pm$ 0.04	-2.775 $\pm$ 0.038	-2.80 $\pm$ 0.00	-2.81 $\pm$ 0.01	**-2.66** $\pm$ 0.01	-2.67 $\pm$ 0.00	-2.84 $\pm$ 0.01

activation regularization (TAR) techniques. We follow [44] for specific hyper-parameter settings such as dropout rates for different types of layers. We train all models for language modeling tasks and evaluate validation and test perplexity. For SWA and SWAG we pre-train the models using standard SGD for 500 epochs, and then run the model for 100 more epochs to estimate the mean θ_{SWA} and covariance Σ in SWAG. For this experiment we introduce a small change to SWA and SWAG: to estimate the mean θ_{SWA} we average weights after each mini-batch of data rather than once per epoch, as we found more frequent averaging to greatly improve performance. After SWAG distribution is constructed we sample and ensemble 30 models from this distribution. We use rank-10 for the low-rank part of the covariance matrix of SWAG distribution.

G Regression

For the small UCI regression datasets, we use the architecture from Wu et al. [60] with one hidden layer with 50 units, training for 50 epochs (starting SWAG at epoch 25) and using 20 repetitions of 90/10 train test splits. We fixed a single seed for tuning before using 20 different seeds for the results in the paper.

We use SGD[8], manually tune learning rate and weight decay, and use batch size of $N/10$ where N is the dataset size. All models predict heteroscedastic uncertainty (i.e. output a variance). In Table 12, we compare subspace inference methods to deterministic VI (DVI, Wu et al. [60]) and deep Gaussian processes with expectation propagation (DGP1-50 Bui et al. [7]). SWAG outperforms DVI and the other methods on three of the six datasets and is competitive on the other three despite its vastly reduced computational time (the same as SGD whereas DVI is known to be 300x slower). Additionally, we note the strong performance of well-tuned SGD as a baseline against the other approximate inference methods, as it consistently performs nearly as well as both SWAG and DVI.

Finally, in Table 12, we compare the calibration (coverage of the 95% credible sets of SWAG and 95% confidence regions of SGD) of both SWAG and SGD. Note that neither is ever too over-confident (far beneath 95% coverage) and that SWAG is considerably better calibrated on four of the six datasets.

H Classification Experimental Details and Parameters

In this section we describe all of the architectures and hyper-parameters we use in Sections 5.1, E.2.

[8]Except for concrete where we use Adam due to convergence issues.

Table 13: Calibration on small-scale UCI datasets. Bolded numbers are those closest to 0.95 %the predicted coverage).

	N	D	SGD	SWAG
boston	506	13	0.913 ± 0.039	$\mathbf{0.936} \pm 0.036$
concrete	1030	8	0.909 ± 0.032	$\mathbf{0.930} \pm 0.023$
energy	768	8	0.947 ± 0.026	$\mathbf{0.951} \pm 0.027$
naval	11934	16	$\mathbf{0.948} \pm 0.051$	0.967 ± 0.008
yacht	308	6	0.895 ± 0.069	$\mathbf{0.898} \pm 0.067$
power	9568	4	$\mathbf{0.956} \pm 0.006$	0.957 ± 0.005

On ImageNet we use architecture implementations and pre-trained weights from `https://github.com/pytorch/vision/tree/master/torchvision`. For the experiments on CIFAR datasets we adapted the following implementations:

- VGG-16: `https://github.com/pytorch/vision/blob/master/torchvision/models/vgg.py`
- Preactivation-ResNet-164: `https://github.com/bearpaw/pytorch-classification/blob/master/models/cifar/preresnet.py`
- WideResNet28x10: `https://github.com/meliketoy/wide-resnet.pytorch/blob/master/networks/wide_resnet.py`

For all datasets and architectures we use the same piecewise constant learning rate schedule and weight decay as in Izmailov et al. [27], except we train Pre-ResNet for 300 epochs and start averaging after epoch 160 in SWAG and SWA. For all of the methods we are using our own implementations in PyTorch. We describe the hyper-parameters for all experiments for each model:

SWA We use the same hyper-parameters as Izmailov et al. [27] on CIFAR datasets. On ImageNet we used a constant learning rate of 10^{-3} instead of the cyclical schedule, and averaged 4 models per epoch. We adapt the code from `https://github.com/timgaripov/swa` for our implementation of SWA.

SWAG In all experiments we use rank $K = 20$ and use 30 weight samples for Bayesian model averaging. We re-use all the other hyper-parameters from SWA.

KFAC-Laplace For our implementation we adapt the code for KFAC Fisher approximation from `https://github.com/Thrandis/EKFAC-pytorch` and implement our own code for sampling. Following [54] we tune the scale of the approximation on validation set for every model and dataset.

MC-Dropout In order to implement MC-dropout we add dropout layers before each weight layer and sample 30 different dropout masks for Bayesian model averaging at inference time. To choose the dropout rate, we ran the models with dropout rates in the set $\{0.1, 0.05, 0.01\}$ and chose the one that performed best on validation data. For both VGG-16 and WideResNet28x10 we found that dropout rate of 0.05 worked best and used it in all experiments. On PreResNet-164 we couldn't achieve reasonable performance with any of the three dropout rates, which has been reported from the work of He et al. [22]. We report the results for MC-Dropout in combination with both SWA (SWA-Drop) and SGD (SGD-Drop) training.

Temperature Scaling For SWA and SGD solutions we picked the optimal temperature by minimizing negative log-likelihood on validation data, adapting the code from `https://github.com/gpleiss/temperature_scaling`.

SGLD We initialize SGLD from checkpoints pre-trained with SGD. We run SGLD for 100 epochs on WideResNet and for 150 epochs on PreResNet-156. We use the learning rate schedule of [59]:

$$\eta_t = \frac{\eta_0}{(\eta_1 + t)^{0.55}}.$$

We tune constants a, b on validation. For WideResNet we use $a = 38.0348$, $b = 13928.7$ and for PreResNet we use $a = 40.304$, $b = 15476.4$; these values are selected so that the initial learning rate is 0.2 and final learning rate is 0.1. We also had to rescale the noise in the gradients by a factor of $5 \cdot 10^{-4}$ compared to [59]. Without this rescaling we found that even with learning rates on the scale of 10^{-7} SGD diverged. We note that noise rescaling is commonly used with stochastic gradient MCMC methods (see e.g. the implementation of [65]).

On CIFAR datasets for tuning hyper-parameters we used the last 5000 training data points as a validation set. On ImageNet we used 5000 of test data points for validation. On the transfer task for CIFAR10 to STL10, we report accuracy on all 10 STL10 classes even though frogs are not a part of the STL10 test set (and monkeys are not a part of the CIFAR10 training set).