On the Effectiveness of Mode Exploration in Bayesian Model Averaging for Neural Networks

John T. Holodnak [1] **Allan B. Wollaber** [1]

Abstract

Multiple techniques for producing calibrated predictive probabilities using deep neural networks in supervised learning settings have emerged that leverage approaches to ensemble diverse solutions discovered during cyclic training or training from multiple random starting points (deep ensembles). However, only a limited amount of work has investigated the utility of exploring the local region around each diverse solution (posterior mode). Using three well-known deep architectures on the CIFAR-10 dataset, we evaluate several simple methods for exploring local regions of the weight space with respect to Brier score, accuracy, and expected calibration error. We consider both Bayesian inference techniques (variational inference and Hamiltonian Monte Carlo applied to the softmax output layer) as well as utilizing the stochastic gradient descent trajectory near optima. While adding separate modes to the ensemble uniformly improves performance, we show that the simple mode exploration methods considered here produce little to no improvement over ensembles without mode exploration.

1. Introduction

Neural networks have been successfully used in many application areas, but perhaps most prominently in image classification. Unfortunately, as networks have become deeper and more complex, their probabilistic outputs have become less calibrated (Guo et al., 2017), and as a result, they do not provide a useful quantification of predictive uncertainty. Many techniques have been proposed to alleviate this problem, including Bayesian Neural Networks (MacKay, 1992; Graves, 2011), deep ensembles (Lakshminarayanan et al., 2017), Monte Carlo dropout (Gal & Ghahramani, 2016), and temperature calibration (Guo et al., 2017).

Wilson & Izmailov (2020), attempt to unify understanding of techniques that aggregate predictions from neural networks with different weights through the lens of the Bayesian Model Average (BMA) (Hoeting et al., 1999)

$$p(y|x; D) = \int p(y|x; \theta)p(\theta|D)d\theta.$$

In the above, $y \in Y = \{1, \ldots, C\}$ is the true label for data point $x \in X$. $D \subset X \times Y$ is the training dataset. For us, $p(y|x; \theta)$ represents the neural network's predicted probabilities for data point x, given specific network weights θ, and $p(\theta|D)$ is the posterior distribution of weights, given the training dataset. Usually, the integral over θ is "approximated" using a single setting of weights, typically obtained from an optimization procedure such as Stochastic Gradient Descent (SGD). As neural networks typically have multimodal posteriors (Müller & Insua, 1998), this approximation is decidedly sub-optimal. Many researchers have proposed methods to approximate the BMA. We now discuss a few of the prominent techniques.

To approximate the BMA, one option is to estimate the integral above using samples from the posterior distribution obtained via Markov Chain Monte Carlo (MCMC) or from a variational approximation to the posterior (often a product of independent Gaussians). Unfortunately, MCMC scales poorly to large datasets and also has difficulty sampling from complicated posteriors in high dimensions. Variational Inference (VI) is more tractable and is implemented in common packages like Tensorflow and PyTorch, but in its typical implementation infers a uni-modal posterior approximation. An alternative is to apply Stochastic Gradient Langevin Dynamics (SGLD), which adds Gaussian noise according to a decaying schedule to SGD and allows sampling from the posterior distribution of parameters (Welling & Teh, 2011). There have been a few recent attempts (see Wenzel et al. (2020); Izmailov et al. (2021)) to better understand the posterior distribution of Bayesian Neural Networks and whether sampling from the posterior results in improved models.

Other not-explicitly-Bayesian techniques can also be viewed as non-trivial approximations to the BMA in that they average predictions over multiple sets of weights, though these

[1]Massachusetts Institute of Technology, Lincoln Laboratory. Correspondence to: John Holodnak <john.holodnak@ll.mit.edu>.

Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning.,

weights are obtained by methods other than sampling from the posterior or approximate posterior. Such techniques include training multiple networks from different starting locations (deep ensembles) (Lakshminarayanan et al., 2017), saving weights from the SGD trajectory with either a traditional or cyclic learning rate schedule (snapshot ensembles) (Huang et al., 2017), training a multiple-input-multiple-output model (Havasi et al., 2021) that at test time produces multiple predictions for an input, or building a Gaussian posterior approximation from the SGD trajectory (Maddox et al., 2019).

Intuitively, ensembles are effective when the individual models make diverse but accurate predictions. To this end, ensembles of models corresponding to different posterior modes are likely to be more diverse than ensembles of models all drawn from the same mode. This intuition is supported by the success of deep ensembles (Lakshminarayanan et al., 2017). A recent comparison by Ovadia et al. (2019) shows that deep ensembles are more robust in terms of accuracy and calibration than solutions from explicitly Bayesian techniques such as Stochastic Variational Inference. An interpretation of this result is that it is better to draw one "sample" from each of K modes than to draw K samples from a single mode (Wilson & Izmailov, 2020).

A logical next question is whether it is advantageous to average model predictions *within* posterior modes as well as *between* posterior modes. This question is partially explored by Wilson & Izmailov (2020), Zhang et al. (2020), and Dusenberry et al. (2020). Wilson & Izmailov (2020) construct Gaussian posterior approximations using the SGD trajectory initialized at several stochastic weight averaged neural network solutions[1]. They then average predictions over a few weights sampled from each Gaussian posterior approximation. They find that this approach outperforms deep ensembles as well as ensembles of stochastic weight averaged networks on corrupted CIFAR-10 images. In addition, they show near monotonic improvement in terms of the negative log-likelihood of PreResNet20 as the number of models is increased from one to ten (for each of the methods mentioned above). Zhang et al. (2020) utilize a cyclic learning rate to identify several posterior modes and either Stochastic Gradient Langevin Dynamics (SGLD) or Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) to sample from them. They demonstrate that SGLD and SGHMC, run in a cyclic fashion, are more effective than traditional SGLD or SGHMC, presumably because the cyclic variants sample from multiple modes. Dusenberry et al. (2020) parameterize each weight matrix with a rank one factor and perform variational inference for this relatively small number of weights. They model the posterior distribution as a mixture distribution with a few components, which enables identification of multiple modes and then draw a few weight samples at test time from each mode. They demonstrate results that are comparable to approaches using several times as many parameters on multiple datasets.

[1]Stochastic weight averaging refers to averaging weights from multiple neural networks to arrive at a single network.

The experiments described above demonstrate that the SGD trajectory can be used to explore posterior modes and obtain improved predictive performance either by building a local Gaussian approximation or via adding noise to the gradient and thus the trajectory (SGLD and SGHMC). In this work, we look to better understand whether the SGD trajectory can be used directly to "sample" from a mode and also to examine whether explicitly Bayesian techniques (VI and MCMC) operating on the last layer of the neural network (which is computationally tractable) can capture within-mode model diversity.

To be specific, we perform experiments on CIFAR-10 using two ensemble types (deep and cyclic) and several inference types: SGD; saving samples from the SGD trajectory; variational inference (VI) on the softmax output layer; and MCMC on the softmax output layer. VI on the last layer has been studied previously, for example see (Ovadia et al., 2019). While it was noted to not perform much differently from standard SGD in the context of a *single* model, we include it due to the simplicity of implementing it in Tensorflow and to provide a baseline for the performance of MCMC on the last layer. Because MCMC scales poorly with dataset size, we accelerate it using Bayesian coresets (Huggins et al., 2016; Campbell & Broderick, 2019). We provide more detail on this acceleration in Appendix A. To our knowledge, this is the first use of coresets in the context of accelerating MCMC for neural networks. We are motivated to explore MCMC in the context of deep neural networks by the discussion section in (Zhang et al., 2020) and both (Wenzel et al., 2020) and (Izmailov et al., 2021).

Our contributions are as follows:

- We perform a detailed set of experiments using multiple runs of the same approximation methods across different network architectures of increasing complexity using the same dataset and learning rate schedule.
- We show that a few epochs of training at a small constant learning rate after using the decaying learning rate schedule from (Huang et al., 2017; Zhang et al., 2020) is effective at refining the SGD solution, but the trajectory is not useful for defining a diverse set of models within the mode.
- We provide the first, to our knowledge, experimental analysis of the effectiveness of MCMC (applied to the output layer), accelerated by a data reduction technique called Bayesian coresets, at estimating the BMA.

- We demonstrate that Bayesian inference on the last layer typically provides little to no improvement to the Brier score over SGD. Last layer VI and last layer MCMC produce similar results overall.

2. Approximation Methods

In this section, we describe the methods we use to approximate the Bayesian Model Average. For all approximation methods, except as noted below, we use a learning rate that decays according to

$$r_b = \frac{r_0}{2}\cos\left(\frac{\pi * b}{B}\right) + \frac{r_0}{2},$$

where b is the batch counter, B is the total number of batches across all epochs, and r_0 is the initial learning rate. This allows the model to initially take large steps and then converge towards a mode. This learning rate is also used in (Huang et al., 2017; Zhang et al., 2020).

The methods considered are as follows:

- *Deep ensembles*: We train M models from different random starting points. We approximate the BMA with

$$p(y|x; D) \approx \frac{1}{M}\sum_{j=1}^{M} p(y|x; \theta_j),$$

where θ_j are the parameters obtained by SGD for model j, $1 \leq j \leq M$.

- *Cyclic ensembles*: We run SGD for M "cycles," meaning that the starting point of optimization for model j is the final set of parameters for model $j-1$. This is identical to snaphsot ensembles (Huang et al., 2017), apart from the fact that we keep the solution from the end of each cycle, rather than only the last several. We approximate the BMA as for deep ensembles.

- *Deep (cyclic) ensembles - SGD trajectory*: We also consider applying SGD, first with the decaying learning rate and then with a small constant learning rate for the last K epochs, and saving samples from the trajectory (when in the constant learning rate stage). We denote the weights obtained from the last K epochs of training each model j as θ_{jk}, $1 \leq k \leq K$. We consider two variants in which we save a single set of weights from the final epoch (SGD-t1) or the weights from the end of each of the last K epochs (SGD-tK). The BMA for SGD-t1 is then

$$p(y|x; D) \approx \frac{1}{M}\sum_{j=1}^{M} p(y|x; \theta_{jK}),$$

and the BMA for SGD-tK is

$$p(y|x; D) \approx \frac{1}{MK}\sum_{j=1}^{M}\sum_{k=1}^{K} p(y|x; \theta_{jk}).$$

We view SGD-t1 as a means to obtain a more refined single solution and SGD-tK as a way to explore the mode of the likelihood. SGD-tK can be viewed as a snapshot version of the Gaussian approximation method from (Maddox et al., 2019), except that instead of constructing a Gaussian using the SGD iterates, we use the raw iterates themselves. Maddox et al. (2019) note the similarity to SGLD as well.

- *Deep (cyclic) ensembles - Last layer VI*: We train M models from random starting points or the end of each cycle of a cyclic learning rate schedule, performing Bayesian inference via stochastic variational inference for the softmax output layer only. The BMA is then approximated as

$$p(y|x; D) \approx \frac{1}{MK}\sum_{j=1}^{M}\sum_{k=1}^{K} p(y|x; \theta_j^*, (\theta_j^{'})_k),$$

where θ_j^* are the deterministic weights obtained by SGD for model j and $(\theta_j^{'})_k$ are sampled from the variational approximation to the posterior distribution of the weights in the last layer of model j.

- *Deep (cyclic) ensembles - Last layer MCMC*: As in last layer VI, except that we use the variational approximations as a starting point to identify a so-called Bayesian coreset, and then run MCMC on the coreset for the parameters associated with the softmax output layer only. See Appendix A. After warmup, we extract K samples from the Markov chains. The BMA is approximated as for last layer VI, except that $(\theta_j^{'})_k$ are sampled from the posterior distribution of the weights in the last layer of model j (assuming the Markov Chain has converged).

3. Experiments

In this section, we describe experiments on CIFAR-10 to evaluate the methods described in the previous section.

We train networks based on VGG13, ResNet20, and DenseNet40 on CIFAR-10. On each of the convolutional bases, we add two dense layers of size 128 and 10, with ReLU and softmax activations, respectively. We report the performance of the BMA approximation methods discussed above. More details on training and our implementations are available in Appendix B. In Figure 1, we display the Brier score of the ensemble models on the test set as the number of models in the ensemble increases from one to sixteen. The Brier score is a proper scoring rule for probabilistic forecasts. To be specific, the Brier score is defined as

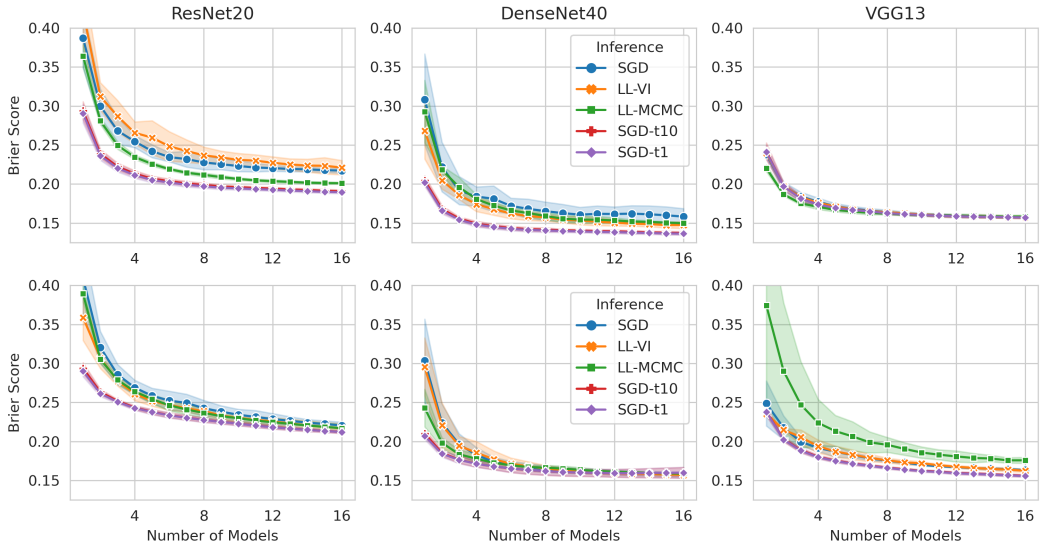

Figure 1. Brier scores against number of models for different approximations to the BMA. The top row shows deep ensembles, while the bottom row shows cyclic ensembles. The columns show three different network architectures. Each point represents the mean of five runs of the approximation method. The colored region covers one standard deviation around the mean.

$BS = \frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}(y_{ic} - p_{ic})^2$, where y_{ic} is one if item i is a member of class c and zero otherwise and p_{ic} is the predicted probability for class c on item i. Note that across convolutional bases, ensemble types (deep or cyclic), and approximation methods, increasing the size of the ensemble decreases the Brier score.

Overall, last layer VI and last layer MCMC produce results not much different from SGD, the main exceptions being ResNet20 with deep ensembles, where last layer MCMC outperforms last layer VI and SGD, and VGG13 with cyclic ensembles, where last layer MCMC performs worse than last layer VI and SGD. On the other hand, SGD-t1 and SGD-t10 perform at least as well as the other methods and considerably better on Resnet20 and DenseNet40 with deep ensembles. Interestingly, SGD-t1 and SGD-t10 produce almost identical results, perhaps indicating that SGD by itself is ineffective at exploring modes in the likelihood. In Figures 3 and 4 in the Appendix, we provide similar plots, except for accuracy and expected calibration error. Accuracy increases with the number of models almost monotonically for all approximation methods, while expected calibration error, in some cases, reaches a minimum for a small number of models (around three) and then increases. Deep ensembles overall have better (lower) Brier scores in Figure 1 (top row) than the cyclic ensembles (bottom row), which is primarily due to their higher accuracies (Figure 3).

We view these results not as evidence that exploring modes is not useful, but as evidence that the simple techniques investigated here are not sufficient to produce within-mode model diversity.

4. Conclusion

In this paper, we performed a detailed set of experiments using several approximation methods for the BMA on three different network architectures.

One of our main goals was to answer whether simple mode exploration techniques could be leveraged to improve the BMA. The answer to this largely seems to be negative. Bayesian inference in the last layer (using either VI or MCMC) is not effective at improving Brier score over the baseline provided by SGD. In addition, while it is helpful to refine the SGD solution obtained from the decreasing learning rate schedule with a few epochs of training with a constant learning rate, it does not appear helpful to save multiple solutions from those epochs. It is possible that using a larger learning rate could make this more effective.

We believe more work is still needed to identify the best ways to explore modes for Bayesian model averaging. An interesting direction would be to combine mode identification with the subspace inference technique from (Izmailov et al., 2020) as a way to identify subspaces in which one could feasibly run MCMC. Combined with the Bayesian coresets idea discussed in Appendix A, this approach could be scalable to large datasets.

References

Campbell, T. and Broderick, T. Bayesian coreset construction via greedy iterative geodesic ascent. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 698–706. PMLR, 10–15 Jul 2018.

Campbell, T. and Broderick, T. Automated scalable Bayesian inference via Hilbert coresets. *J. Mach. Learn. Res.*, 20(1):551588, January 2019. ISSN 1532-4435.

Dusenberry, M., Jerfel, G., Wen, Y., Ma, Y., Snoek, J., Heller, K., Lakshminarayanan, B., and Tran, D. Efficient and scalable Bayesian neural nets with rank-1 factors. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2782–2792. PMLR, 13–18 Jul 2020.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

Graves, A. Practical variational inference for neural networks. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pp. 23482356, Red Hook, NY, USA, 2011. Curran Associates Inc. ISBN 9781618395993.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 13211330. JMLR.org, 2017.

Havasi, M., Jenatton, R., Fort, S., Zhe Liu, J., Snoek, J., Lakshminarayanan, B., Dai, A. M., and Tran, D. Training independent subnetworks for robust prediction. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

Hoeting, J. A., Madigan, D., Raftery, A. E., and Volinsky, C. T. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–401, 1999. ISSN 08834237.

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. Snapshot ensembles: Train 1, get M for free. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Huggins, J. H., Campbell, T., and Broderick, T. Coresets for scalable Bayesian logistic regression. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 40874095, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Izmailov, P., Maddox, W. J., Kirichenko, P., Garipov, T., Vetrov, D., and Wilson, A. G. Subspace inference for bayesian deep learning. In Adams, R. P. and Gogate, V. (eds.), *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pp. 1169–1179. PMLR, 22–25 Jul 2020.

Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. What are bayesian neural network posteriors really like? *arXiv preprint arXiv:2104.14421*, 2021.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

MacKay, D. J. C. A practical Bayesian framework for backpropagation networks. *Neural Comput.*, 4 (3):448472, May 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.3.448.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. A simple baseline for Bayesian uncertainty in deep learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Müller, P. and Insua, D. R. Issues in Bayesian analysis of neural network models. *Neural Computation*, 10(3): 749–770, 1998. doi: 10.1162/089976698300017737.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pp. 681688, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

Wenzel, F., Roth, K., Veeling, B., Swiatkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. How good is the Bayes posterior in deep neural networks really? In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10248–10259. PMLR, 13–18 Jul 2020.

Wilson, A. G. and Izmailov, P. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.

Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. Cyclical stochastic gradient MCMC for Bayesian deep learning. *International Conference on Learning Representations*, 2020.

A. Bayesian Coresets for Neural Networks

Bayesian coresets (Huggins et al., 2016; Campbell & Broderick, 2019) is a technique to identify sparse weights for the training dataset so that the weighted log-likelihood is close to the true log-likelihood. That is, find weights $w_n \geq 0$ such that

$$\sum_{i=1}^{N} \ln(p(y_n|\theta)) \approx \sum_{i=1}^{N} w_n \ln(p(y_n|\theta)),$$

with the constraint that at most $N* << N$ weights are non-zero. Campbell & Broderick (2018) and Campbell & Broderick (2019) discuss optimization techniques to solve the problem, which require computing weighted inner products of log-likelihood functions

$$\left\langle \ln(p(y_{n_i}|\theta)), \ln(p(y_{n_j}|\theta)) \right\rangle_{\pi},$$

where π is an approximation to the posterior distribution. To make this computation practical, the authors utilize random projections to project the log-likelihood functions into a finite dimensional space and obtain a quick approximation to the posterior distribution to use as the weighting function (via Laplace approximation, variational inference, informed prior, etc.).

Unfortunately, MCMC also scales poorly with the number of parameters. As a result, we only run MCMC on the last layer of the neural network.

Recall that in our work, we are attempting to explore multiple posterior modes using either deep or cyclic ensembles. Our goal is to use MCMC to "explore" each mode. As such, we need to find a coreset for each posterior mode. We use variational inference to approximate the posterior distribution of the weights in the last layer, for each mode. We project the likelihood functions into a 5,000 dimensional space and solve the optimization problem using the technique from (Campbell & Broderick, 2018), using a maximum coreset size of 1,500. Finally, we run the No U-Turn Sampler (NUTS) as implemented in Tensorflow Probability with M chains (one chain initialized in each mode). We run all M chains simultaneously on the same GPU. We use 10,000 warm-up iterations and then draw 1,000 samples. We thin the samples by a factor of 100 to obtain 10 samples per mode. These samples are used in combination with the deterministic weights from the rest of the network to obtain 10 predictions per mode.

B. CIFAR-10 details

We use stochastic gradient descent with no momentum as our optimizer. We use 80% of the dataset for training and reserve the remainder for validation to control early stopping. We train all models (except for SGD trajectory) for 50 epochs using early stopping, with patience set to 10 epochs. The learning rate decreases for each batch. We use a batch size of 32, so there are 1,250 batches per epoch. For SGD trajectory, we run the decreasing learning rate schedule for at most 40 epochs (50,000 batches), then run $K = 10$ epochs (12,500 batches) with a constant learning rate of 0.0001. For SGD trajectory, if we run into the early stopping criterion, we immediately jump to the constant learning rate stage. The learning rate schedules are shown in Figure 2. We set the initial learning rate to $r_0 = 0.1$. We experimented with larger initial learning rates, but settled on one that worked well for all architectures[2].

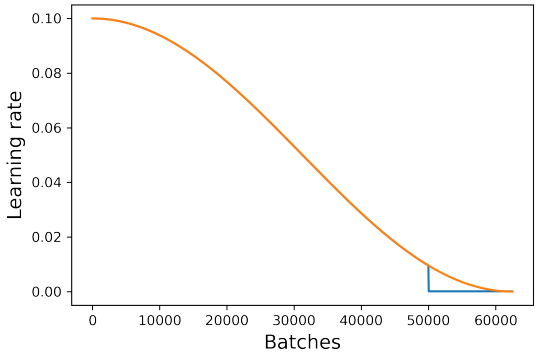

Figure 2. Learning rate schedules.

We use existing implementations of ResNet20[3], VGG13[3], and DenseNet40[4].

C. Acknowledgements

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Air Force.

[2] VGG13 did not work well with $r_0 = 0.5$, for example.
[3] https://github.com/gahaalt/resnets-in-tensorflow2
[4] https://github.com/titu1994/DenseNet

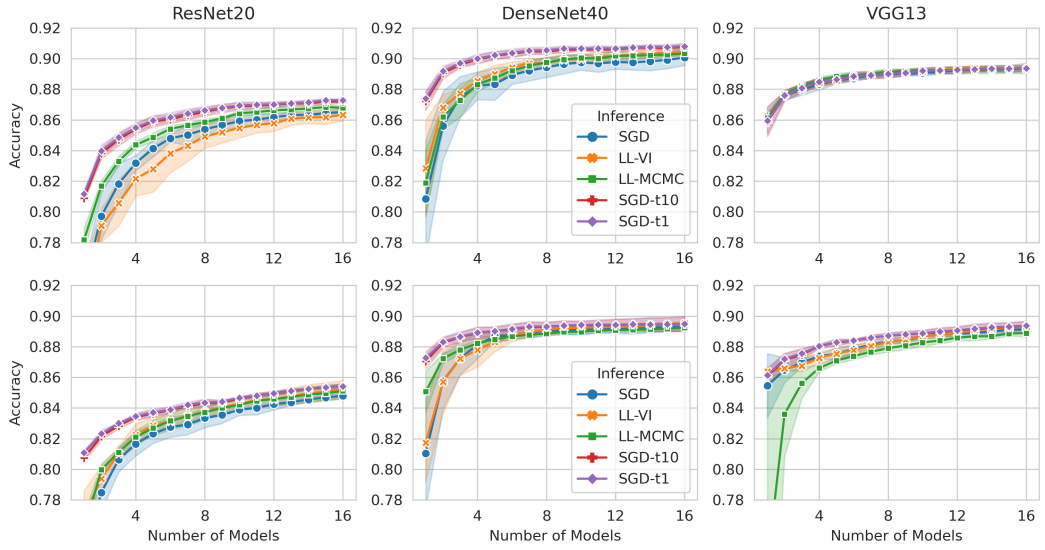

Figure 3. Accuracy of models using different approximations to the BMA. The top row shows deep ensembles; the bottom row shows cyclic ensembles. The columns show three different network architectures. Each point represents the mean of five runs of the approximation method. The colored regions cover one standard deviation around the mean.

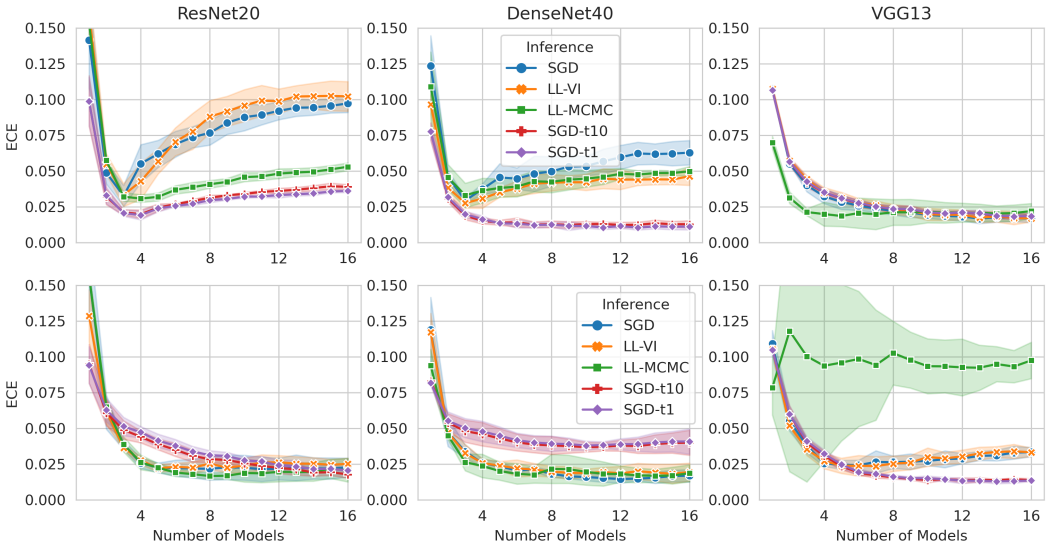

Figure 4. Expected calibration errors of models using different approximations to the BMA. The top row shows deep ensembles; the bottom row shows cyclic ensembles. The columns show three different network architectures. Each point represents the mean of five runs of the approximation method. The colored regions cover one standard deviation around the mean.