HWA: Averaging Hyperparameters in Bayesian Neural Networks Leads to Better Generalization.

Anonymous authors
Paper under double-blind review

Abstract

Bayesian Deep Learning presents itself as the most useful tool for adding uncertainty estimation to traditional Deep Learning models that only produce point estimates predictions as outputs. Confidence of the model and the predictions at inference time are left alone. Applying randomness and Bayes Rule to the weights of a deep neural network is a step towards achieving this goal. Current state of the art optimization method for training a Bayesian Neural Network are relatively slow and inefficient, compared to their deterministic counterparts. In this paper, we propose HWA (Hyperparameters Weight Averaging) algorithm that leverages averaging procedure in order to train faster and achieve a better accuracy. We develop our main algorithm using the simple averaging heuristic and demonstrate its effectiveness on the space of the hyperparameters of the neural networks random weights. Numerical applications confirm the empirical benefits of our method.

1 Introduction

While Deep Learning methods have shown increasing efficiency in various domains such as natural language processing, computer vision or robotics, sensible areas including autonomous driving or medical imaging not only require accurate predictions but also uncertainty quantification. In (Neal, 2012), authors develop a bayesian variant of plain feedforward multilayer neural networks in which weights and biases are considered as random variables. For supervised learning tasks, deterministic models are prone to overfitting and are not capable of estimating uncertainty in the training data resulting in making overly confident decisions about the correct class, *i.e.* miscalibration (Guo et al., 2017; Kendall & Gal, 2017). Nevertheless, representing that aforementioned uncertainty is crucial for decision making.

Bayesian methods display a hierarchical probabilistic model that assume a (prior) random distribution over the parameters of the parameters and are useful for assessing the uncertainty of the model via posterior predictive distribution quantification (Blundell et al., 2015; Kingma et al., 2015). Current training methods for Bayesian Neural Networks (BNN) (Neal, 2012) include Variational Inference (Graves, 2011; Hoffman et al., 2013) or BayesByBackprop (Blundell et al., 2015) based on Evidence Lower Bound (ELBO) maximization task. Naturally, Bayesian methods, and in particular BNNs, are thus highly sensitive to the parameters choice of the prior distribution and current state-of-the-art models are not as efficient and robust as traditional deep learning models.

In this paper, we introduce a new *optimization* algorithm to alleviate those challenges. Our main contributions read as follows:

- We introduce Hyperparameter Weight Averaging (HWA), a training algorithm that leverages stochastic averaging techniques (Polyak & Juditsky, 1992) and posterior sampling methods.
- Given the high nonconvexity of the loss landscape, our method finds heuristic explanation from theoretical works on averaging and generalization such as (Keskar et al., 2016; He et al., 2019) and more practical work on Deep Neural Networks (DNN) optimization such as (Izmailov et al., 2018).
- Plots to show how HWA adapts to the curvature and reaches a better testing accuracy (but worst training loss). Plots with hyperparameters landscape and HWA trajectory on PCA subspace.
- We provide numerical examples showcasing the effectiveness of our method on simple and complex supervised classification tasks.

The remaining of the paper is organized as follows. Section 2 presents the related works in the fields of optimization, Variational Inference and posterior sampling. Section 3 introduces the HWA algorithm which is the main contribution of our paper. Section 4 highlights the benefits of our averaging procedure on simple and complex numerical classification tasks. Section 5 concludes our work.

Notations: We denote for all $n > 1$, $[n]$ the set $\{1, \cdots, n\}$.

2 Related Work

Posterior Prediction. Due to the nonconvexity of the loss landscapes involved in modern and complex deep learning tasks, direct sampling from the posterior distribution of the weights is not an option. Depending on the nature and in particular the dimensionality of the problem, Markov Chain Monte Carlo (MCMC) methods have been employed to overcome this intractability issue. By constructing a Markov chain, the samples drawn at convergence are guaranteed to be drawn from the target distribution. Hamiltonian Monte Carlo (HMC) (Neal et al., 2011) or Metropolis Hastings (MH) (Hastings, 1970) are two standard solutions used. Their stochastic gradients counterpart are extensively studied in (Ma et al., 2015).

Variational Inference (VI). When tackling an optimization problem, exact posterior sampling may be computationally involved and not even required. variational inference was proposed in (Graves, 2011), in the particular case of BNNs, in order to fit a Gaussian variational posterior approximation over the weights of neural networks. Through a simple reparameterization trick (Blundell et al., 2015), several methods have emerged to train BNNs leveraging the ease of use and implementation of VI (Kingma et al., 2015; Blundell et al., 2015; Molchanov et al., 2017). Though, those methods appear to be inefficient for large-scale datasets and newer methods were proposed to alleviate this issue such as the use of normalizing flows (Louizos & Welling, 2017), deterministic VI (Wu et al., 2018) or dropout VI (Gal & Ghahramani, 2016).

Stochastic Averaging. Averaging methods include the seminal papers of (Polyak, 1990) and (Ruppert, 1988), both based on the combination of past iterates along a stochastic approximation trajectory. For nonconvex loss objectives, this averaging procedure has been adapted to Stochastic Gradient Descent (SGD) trajectory (Zhou & Cong, 2017). In particular, in modern deep learning examples, Izmailov et al. (2018) develops a novel method that averages snapshots of a deep neural networks (DNN) through the iterations and shows empirical benefits leading to a better generalization. Those experimental discoveries are then backed by theoretical understanding of the DNN loss landscape and the impact of averaging successive iterates in (Keskar et al., 2016; He et al., 2019).

3 Hyperparameters Averaging in Bayesian Neural Networks

In this section, we introduce the basic concepts of Bayesian Neural Networks and their corresponding loss function which plays a key role in this paper. From an optimization perspective, we review the Stochastic Weight Averaging (SWA) (Izmailov et al., 2018) averaging procedure, which can be seen as an approximation of the mean trajectory of the SGD iterates and introduce our method, namely HWA. We then discuss the uncertainty estimation prediction of such method and how our proposed extra step combining *posterior sampling* and *optimization* can lead to a better generalization of the trained model on test sets.

3.1 Bayesian Neural Networks and ELBO Maximization

Let $((x_i, y_i), i \in [n])$ be i.i.d. input-output pairs and $w \in \mathcal{W} \subseteq \mathbb{R}^d$ be a latent variable. When conditioned on the input data $x = (x_i, i \in [n])$, the joint distribution of $y = (y_i, i \in [n])$ and w is given by:

$$p(y, w|x) = \pi(w) \prod_{i=1}^{n} p(y_i|x_i, w) \ . \tag{1}$$

In the particular case of BNN, this likelihood function is parametrized by a multilayer neural network, which can be convolutional or not. The latent variables w are thus the weights and the biases of the model and are considered as latent (and random) variables. Training of such hierarchical models implies sampling from the posterior distribution of the weights w conditioned on the data (x, y) and noted $p(w|y, x)$. In most cases, this posterior distribution $p(w|y, x)$ is intractable and is approximated using a family of parametric distributions, $\{q(w, \boldsymbol{\theta}), \boldsymbol{\theta} \in \Theta\}$. The variational inference (VI) problem (Blei et al., 2017) boils down to minimizing the Kullback-Leibler (KL) divergence between $q(w, \boldsymbol{\theta})$ and the posterior distribution $p(w|y,x)$:

$$\min_{\boldsymbol{\theta}\in\Theta} \mathcal{L}(\boldsymbol{\theta}) := \mathrm{KL}\left(q(w;\boldsymbol{\theta})\,||\,p(w|y,x)\right) := \mathbb{E}_{q(w;\boldsymbol{\theta})}\big[\log\big(q(w;\boldsymbol{\theta})/p(w|y,x)\big)\big] \ . \tag{2}$$

Using (1), we decompose $\mathcal{L}(\boldsymbol{\theta}) = n^{-1}\sum_{i=1}^{n}\mathcal{L}_i(\boldsymbol{\theta}) + \text{const.}$ where:

$$\mathcal{L}_i(\boldsymbol{\theta}) := -\mathbb{E}_{q(w;\boldsymbol{\theta})}\big[\log p(y_i|x_i,w)\big] + \frac{1}{n}\mathbb{E}_{q(w;\boldsymbol{\theta})}\big[\log q(w;\boldsymbol{\theta})/\pi(w)\big] \ . \tag{3}$$

Directly optimizing the finite sum objective function in (2) can be difficult. First, with $n \gg 1$, evaluating the objective function $\mathcal{L}(\boldsymbol{\theta})$ requires a full pass over the entire dataset. Second, for some complex models, the expectations in (3) can be intractable even if we assume a simple parametric model for $q(w;\boldsymbol{\theta})$.

Solutions simply include using SGD (Bottou & Bousquet, 2008) where the gradient of the individual ELBO (3) is computed using Automatic Differentiation (Kucukelbir et al., 2017). The final update goes in the opposite direction of that gradient up to a learning rate factor. In the sequel, we develop an improvement over baseline SGD, invoking averaging virtue of several successive snapshots of the gradients. The method, called Hyperparameters Weight Averaging (HWA), aims at improving the generalization property of the trained model on unseen data.

3.2 Averaging iterates through hyperparmeter loss landscapes snapshots.

We now recall the classical SWA method developed in (Izmailov et al., 2018). Consider a deterministic deep neural network, the idea behind the Stochastic Weight Averaging procedure is to run several iterates of the classical SGD procedure, starting from a pre-trained model. At each timestep noted T_{avg}, the model estimate is equal to the average of the last T_{avg} iterates. Empirically, a constant and large learning rate is more efficient given the exploration virtue that it implies.

After establishing the connectivity between several modes (point estimate of minimal loss) of the same deep neural network (after different training procedure) in (Garipov et al., 2018), being able to average all those iterates probably traversing several models or at least model estimates that belong to low loss region would make the resulting trained model more robust and thus generalize better to unseen data. Several theoretical paper such as (He et al., 2019) or (Keskar et al., 2016) provide an attempt at explaining this phenomena.

Hyperparameters Weight Averaging: Based on the probabilistic model developed above, the loss function (2) is defined on the space of the hyperparameters, *i.e.* the mean and the variance of the variational candidate distribution.

Regarding the parameterization choice of the variational candidate $q(w;\boldsymbol{\theta})$, we chose for simplicity a scalar mean μ_ℓ depending on the layer $\ell \in [L]$ and constant between each neuron. Classically, the covariance of this variational distribution is diagonal, see (Kirkpatrick et al., 2017; Blundell et al., 2015), yet can be too restrictive. We follow the direction taken in (Maddox et al., 2019), where the covariance of $q(w,\boldsymbol{\theta})$ is a low-rank plus diagonal posterior approximation matrix.

As a result, the averaging procedure practically occurs on the set of hyperparameters and requires updating the mean and the variance of the variational candidate distribution, at iteration $k+1$, if k, the iteration index, is a multiple of the cycle length c, as below:

$$\begin{aligned}\mu_\ell^{HWA} &= \frac{n_{\text{m}}\mu_\ell^{HWA} + \mu_\ell^{k+1}}{n_{\text{m}}+1}\\ \sigma^{HWA} &= \frac{n_{\text{m}}\sigma^{HWA} + (\mu_\ell^{k+1})^2}{n_{\text{m}}+1} - (\mu_\ell^{HWA})^2\end{aligned} \tag{4}$$

where for all $\ell \in [L]$, μ_ℓ^{k+1} and σ^{k+1} are both obtained via Stochastic Variational Inference.

Algorithm 1 HWA: Hyperparameters Weight Averaging

```
1:  Input: Trained hyperparameters $\hat{\mu}_\ell$ and $\hat{\sigma}$. LR bounds $\gamma_1$ and $\gamma_2$. Cycle length $c$.
2:  Initialize the hyperparameters of the weights and $\mu_\ell = \hat{\mu}_\ell$ and $\mu_\ell^{HWA} = \mu_\ell$.
3:  for $k = 0, 1, ...$ do
4:    $\gamma \leftarrow \gamma(k)$ (Cyclical LR for the iteration)
5:    SVI updates:
6:      $\mu_\ell^{k+1} \leftarrow \mu_\ell^k - \gamma_k \nabla \mathcal{L}(\mu_\ell^k)$
7:      $\sigma^{k+1} \leftarrow \sigma^k - \gamma_k \nabla \mathcal{L}(\sigma^k)$
8:    if mod$(k, c) = 0$ then
9:          $n_{\mathrm{m}} \leftarrow k/c$    (Number of models to average over)
10:       $\mu_\ell^{HWA} \leftarrow \frac{n_{\mathrm{m}} \mu_\ell^{HWA} + \mu_\ell^{k+1}}{n_{\mathrm{m}}+1}$
11:       $\sigma^{HWA} \leftarrow \frac{n_{\mathrm{m}} \sigma^{HWA} + (\mu_\ell^{k+1})^2}{n_{\mathrm{m}}+1} - (\mu_\ell^{HWA})^2$
12:   end if
13: end for
14: Return hyperparameters $(\{\mu_\ell^{HWA}\}_{l=1}^L, \sigma^{HWA})$.
```

Algorithm 1 develops the main method of our paper. Stochastic Variational update is executed Line 6. The stochastic averaging procedure happens every c iterations, and consists in computing the weight sum between the latest model estimate and the running average noted using the superscript HWA.

Several hyperparameters are worth highlighting here. The standard learning rate γ_k plays a key role and is either equal to a constant or a cyclical learning rate. The cycle length c which monitors the number of times snapshots of the model estimates are being averaged is also of utmost importance and needs careful tuning.

3.3 Through the Lens of Loss Landscapes

PCA plot of the loss landscape and visualization of HWA iterates.

4 Numerical Experiments

In this section, we provide experiments on classification tasks with various neural network architectures and datasets to demonstrate the effectiveness of our hyperparameter averaging method, namely HWA.

Methods. We consider two baselines. The first one is the standard BayesByBackprop method developed in (Blundell et al., 2015). This optimization is achieving state-of-the-art results when training Bayesian Neural Network. The second method is the Stochastic Gradient Langevin Dynamics (SGLD) optimization method introduced in (Welling & Teh, 2011). The algorithms are initialized at the same point and the results are averaged over 5 repetitions.

Datasets. We compare different algorithms on *MNIST* (LeCun & Cortes, 2010) and *CIFAR10* (Krizhevsky et al., 2009) datasets.

Network architectures. We train a Bayesian variant of LeNet-5 convolutional neural network on the MNIST dataset. The training set is composed of $n = 55\,000$ handwritten digits, 28×28 images. Each image is labelled with its corresponding number (from zero to nine). Under the prior distribution π, see (1), the weights are assumed independent and identically distributed according to $\mathcal{N}(0, 1)$. We also assume that $q(\cdot; \boldsymbol{\theta}) \equiv \mathcal{N}(\mu, \sigma^2 \mathbf{I})$. The variational posterior parameters are thus $\boldsymbol{\theta} = (\mu, \sigma)$ where $\mu = (\mu_\ell, \ell \in [d])$ where d is the number of weights in the neural network. We use the re-parametrization as $w = t(\boldsymbol{\theta}, z) = \mu + \sigma z$ with $z \sim \mathcal{N}(0, I)$.

We train here the Bayesian variant of the VGG neural network introduced in (Simonyan & Zisserman, 2014) on the CIFAR-10 dataset. The latter dataset is composed of $n = 60\,000$ handwritten digits, 32×32 colour images in 10 classes, with $6\,000$ images per class. As in the previous example, the weights are assumed independent and identically distributed according to $\mathcal{N}(0, \mathbf{I})$. Standard hyperparameters values found in the literature, such as the annealing constant or the number of MC samples, were used for the benchmark methods. For better efficiency and lower variance, the Flipout estimator (Wen et al., 2018) is used.

Results.

4.1 TRAINING BAYESIAN LENET-5 FOR MNIST (LECUN ET AL., 1998) CLASSIFICATION

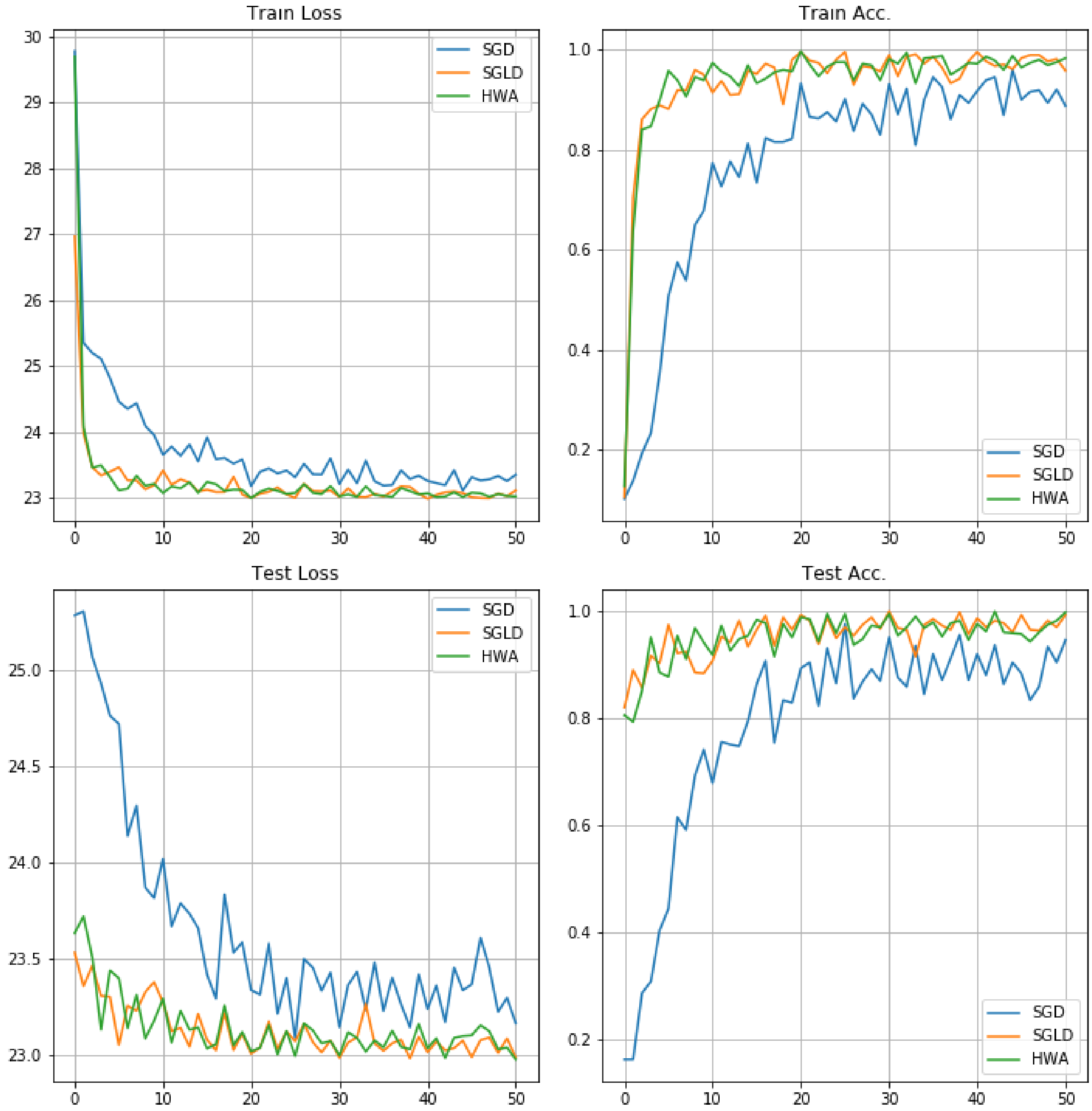

Figure 1: Comparison for Bayesian LeNet CNN architecture on MNIST dataset.

4.2 Training Bayesian VGG for CIFAR10 (Krizhevsky et al., 2009) classification

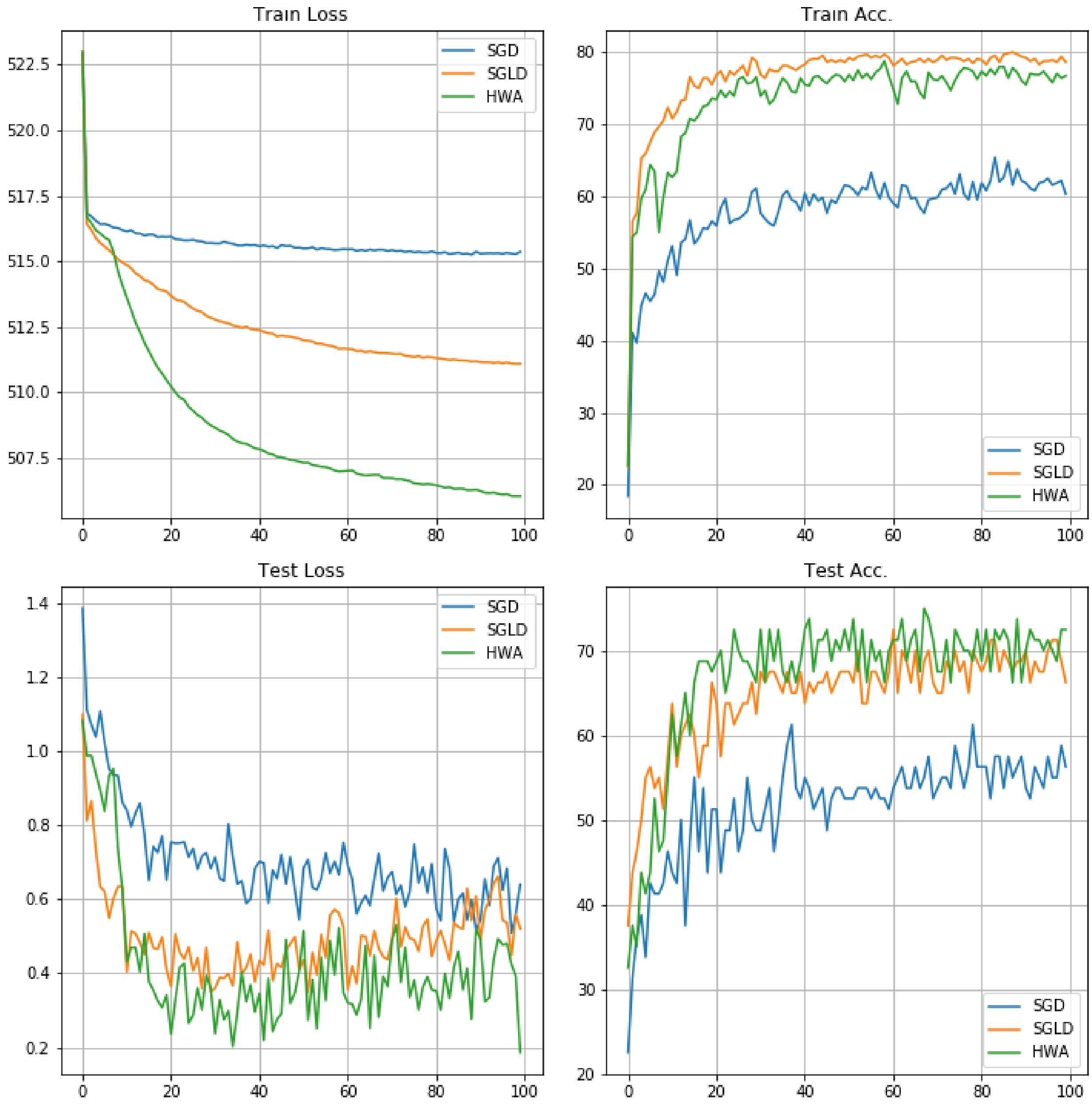

Figure 2: Comparison for Bayesian VGG CNN architecture on CIFAR-10 dataset.

5 Conclusion

References

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

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

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pp. 161–168, 2008.

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

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

Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pp. 2348–2356, 2011.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

Haowei He, Gao Huang, and Yang Yuan. Asymmetric valleys: Beyond sharp and flat local minima. In *Advances in Neural Information Processing Systems*, pp. 2553–2564, 2019.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

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

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pp. 5574–5584, 2017.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems*, pp. 2575–2583, 2015.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

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

Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *The Journal of Machine Learning Research*, 18(1):430–474, 2017.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. *arXiv preprint arXiv:1703.01961*, 2017.

Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient mcmc. In *Advances in Neural Information Processing Systems*, pp. 2917–2925, 2015.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pp. 13153–13164, 2019.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. *arXiv preprint arXiv:1701.05369*, 2017.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

Boris T Polyak. A new method of stochastic approximation type. *Avtomat. i Telemekh*, (7):98:107, 1990.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

David Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

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

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

Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*, 2018.

Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E Turner, José Miguel Hernández-Lobato, and Alexander L Gaunt. Deterministic variational inference for robust bayesian neural networks. *arXiv preprint arXiv:1810.03958*, 2018.

Fan Zhou and Guojing Cong. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. *arXiv preprint arXiv:1708.01012*, 2017.

A Appendix

You may include other additional sections here.