Bayesian Machine Learning - Project Report

Thomas Lemercier **Clement Wang**

Abstract

This report presents an in-depth analysis of our work on the paper "Averaging Weights Leads to Wider Optima and Better Generalization" [1], as part of the Bayesian Machine Learning course in the MVA program. This paper proposes a novel training procedure, which is supposed to improve generalization performance over conventional training methods, with minimal additional overhead. The report provides a comprehensive review of the paper, our understanding of the proposed method, and the results of our experiments and analyses. Noteworthy is the inclusion of a **fresh array of experiments**, absent in the original paper, which assesses the method's efficacy across synthetic regression datasets and a graph dataset. Our results indicate that SWA may not be a feasible or efficient approach in the experimental settings we explored. Our implementation can be found here: https://github.com/ThomasLEMERCIER/BayesianML-SWA.

1 Introduction

The pursuit of improved convergence, stability, and accuracy in deep learning training procedures is contingent upon a thorough understanding of the complex loss surfaces within multilayer networks. Recent studies [2] have uncovered the connectivity of local optima discovered by Stochastic Gradient Descent (SGD), revealing that these optima can be connected by simple curves of near-constant loss. Building upon these findings, the authors of the original paper introduced Fast Geometric Ensembling (FGE), a method that utilizes high-frequency cyclical learning rates with SGD to construct high-performing ensembles efficiently. In this report, we focus on the novel approach proposed in the paper, Stochastic Weight Averaging (SWA), which involves averaging the points traversed by SGD with a cyclical or high constant learning rate. SWA demonstrates significant potential for training deep neural networks, offering improved generalization and minimal overhead compared to standard SGD training.

The experiments conducted in the original paper show that SWA converges to solutions in flatter regions of the training loss, providing a more centered alternative to SGD's tendency to locate points on the periphery of good weight sets. Notably, SWA solutions exhibit distinct characteristics from local optima, implying a broader understanding of optimization landscapes in deep learning. In this report, we present our work on the paper "Averaging Weights Leads to Wider Optima and Better Generalization" by Izmailov et al., 2019, and provide a comprehensive review of the proposed method, our understanding of it, and the results of our experiments and analyses.

2 Stochastic Weight Averaging

The concept of stochastic weight averaging involves sampling various weights along the trajectory of the stochastic gradient descent algorithm and using these sample points to compute a new weight vector w_{swa}, being the mean of all the sample points.

More formally, given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i \in \{1,\cdots,n\}}$ and a loss function $\ell(x, y)$, we first pretrain a neural network using any optimization procedure, resulting in the obtained weight $\hat{w}$.

Next, we perform multiple sequential fine-tuning of this network using a cyclical learning rate. The proposed learning rate schedule follows a cyclical linear pattern as shown in Figure 1.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

The formulation of such a learning rate schedule is as follows:

$$\begin{cases} \eta(t) = \eta_{max} - c(t)\frac{\eta_{min}-\eta_{max}}{T-1} \\ c(t) = \text{mod}(t, T) \end{cases}$$

Here, t denotes the current epoch starting from 0, T denotes the period of the cycle, and η_{max} and η_{min} represent the maximum and minimum learning rates, respectively.

At each minimum of the learning rate, we update a weight vector such that at the end of the training, this weight vector corresponds to the mean of all the weight vectors at the minimum of the learning rate. This is done sequentially to reduce the memory overhead of this method, using only one additional weight vector, as shown below:

$$w_{swa}^{n} = \frac{1}{n}\sum_{i=1}^{n} w_i = \frac{n-1}{n(n-1)}\sum_{i=1}^{n-1} w_i + \frac{1}{n}w_n = \frac{(n-1)w_{swa}^{n-1} + w_n}{n}$$

In summary, the SWA algorithm can be outlined as follows:

Algorithm 1: Stochastic weight averaging

Inputs.

1. A dataset: $\mathcal{D} = \{(x_i, y_i)\}_{i\in\{1,\cdots,n\}}$
2. A loss function: $\ell(x, y)$
3. A pretrained weight vector: $\hat{w}$
4. The learning rate schedule: $\eta_{max}, \eta_{min}, T$
5. The number of training epochs: N

Stochastic weight averaging.

$w \leftarrow \hat{w}$ {Initialize the weight for optimization}
$w_{swa} \leftarrow 0$ {Initialize the SWA weight vector}
$n \leftarrow 0$ {Initialize the number of model used in the SWA model}
for $t = 0$ **to** $N - 1$ **do**
  $\eta \leftarrow \eta_{max} - c(t)\frac{\eta_{min}-\eta_{max}}{T-1}$ {Compute the learning rate}
  $w \leftarrow w - \eta\nabla\ell(\mathcal{D})$ {Stochastic gradient descent}
  if $mod(t + 1, T)$ **then**
    $w_{swa} \leftarrow \frac{nw_{swa}+w}{n+1}$ {Update the SWA weight vector}
    $n \leftarrow n + 1$ {Update the number of model used in the SWA model}

Output.

- The SWA weight vector: w_{swa}

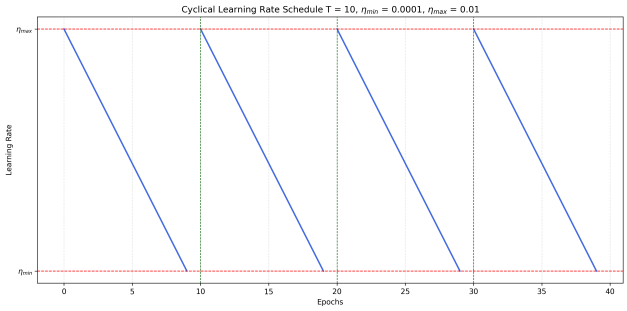

Figure 1: Cyclical learning rate schedule

3 Theoretical Insights: Relationship to Ensembling

This study marks the authors' second foray into the domain of loss landscape and generalization. Their initial work [2] introduced a methodology similar to the one presented here. Rather than deriving

the average weight vector from sampled points along the stochastic gradient descent trajectory, the authors utilized these sampled points to construct an ensemble model using techniques like prediction averaging or voting.

Let's delve into a comparative analysis of these two approaches. Ensembling entails averaging the function of each network:

$$f_{\text{ensemble}} = \frac{1}{n}\sum_{i=1}^{n} f(w_i)$$

where $f(w_i)$ represent the value function of network i. Linearizing f at w_{SWA}, we get

$$f(w_i) = f(w_{\text{SWA}}) + \langle \nabla f(w_{\text{SWA}}), \Delta_i \rangle + O(\|\Delta_i\|^2)$$

where $\Delta_i = w_i - w_{\text{SWA}}$. Summing it up, we arrive at:

$$f_{\text{ensemble}} = O(\Delta^2)$$

Based on this analysis, we can infer that the SWA algorithm approximates ensembling without increasing the number of parameters.

4 Experiments

4.1 Synthetic Datasets

Our initial experiments aimed to evaluate the proposed method on synthetic datasets. We examined a total of three synthetic datasets:

1. **2D Dataset** (refer to Figure 2): This dataset is defined by the following function:
$$f(x, y) = 2\cos(\pi xy) - 0.5x + 0.08y + \sin(\pi x) - \sin(\pi y) + \xi\mathcal{N}(0, 1)$$
2. The other three datasets, also known as the **Friedman datasets**, are defined in the following papers [3], [4].

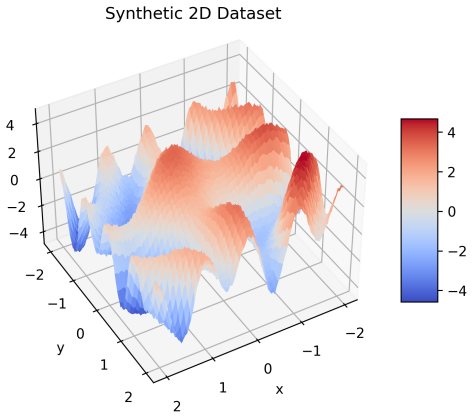

Figure 2: 2D Synthetic dataset

These synthetic datasets serve as straightforward regression datasets. The architecture we employed for training on these datasets is a basic multi-layer perceptron.

4.2 Image classification

Similar to the study discussed in the paper, we evaluated our implementation using image classification datasets.

Due to limitations in computational power, we restricted our implementation to two small CNN architectures:

- A basic CNN network: comprising 3 convolutional layers with max-pooling in between and two linear layers at the end.
- A MobileNet V2 network [5]: recognized for its robust performance despite having a low number of parameters.

4.3 Node classification

The main contribution of this report centers on our experimentation with the proposed method applied to a novel data domain: graphs.

To evaluate the proposed method's performance with graph data, we employed one of the datasets introduced by [6], specifically the CLUSTER dataset. This dataset pertains to node classification, comprising 10,000 training samples, each node having 6 input features, and 6 classes for classification.

In this endeavor, we adopted one of the most established graph neural network architectures, the Graph Convolutional Network (GCN), following the architectural principles outlined in the reference paper by [6].

5 Results

5.1 Synthetic Datasets

The results for synthetic datasets are summarized in Table 1. For each dataset, the metric used is the mean square error (MSE), which is also the loss used during the optimization procedure. To clarify the designation, the "pretrained model" corresponds to the model trained using a classical optimization method until convergence (here, gradient descent using the AdamW optimizer), the "SWA model" corresponds to the model obtained using the proposed method, and the "Final model" corresponds to the model obtained at the very end of the training, the last point of the SWA procedure.

As the table illustrates, the proposed method does not seem to yield any substantial improvements. The only dataset (Friedman 1) where the proposed method demonstrates a benefit, the gain is only 0.33% compared to the final model and 3% compared to the pretrained model.

Dataset	Pretrained	Final	SWA
2D	0.21922	**0.214709**	**0.214709**
FriedMan 1	0.173177	0.168545	**0.167995**
FriedMan 3	0.047658	**0.047594**	0.047604

Table 1: Synthetic regression metrics

5.2 MNIST, CIFAR10, CIFAR100

Our attempts to replicate the findings of the original paper were unsuccessful: we observed that SWA did not yield improvements over standard training. In instances where it did exhibit better performance, we attribute it solely to random fluctuations (Table 2).

	Test loss				Test accuracy (%)			
DNN	Pretrained	Final	Ensemble	SWA	Pretrained	Final	Ensemble	SWA
MNIST								
CNN	0.0473	0.0446	**0.0445**	**0.0445**	98.49	98.50	**98.51**	**98.51**
CIFAR10								
CNN	1.1884	1.1813	**1.1804**	1.1806	58.11	58.54	58.58	**58.64**
MobileNetV2	0.6191	**0.6065**	0.6078	0.6079	78.43	**79.01**	78.94	78.84
CIFAR100								
MobileNetV2	2.1301	**2.1207**	2.1215	2.1254	43.38	43.50	**43.51**	43.43

Table 2: Image classification metrics

In the case of CIFAR100, it's evident that the network hadn't had sufficient time to converge fully to a valley, making SWA even less pertinent in this context.

5.3 CLUSTER dataset

Unsurprisingly, the results on graph data follow the trend observed in other experiments. Although the SWA model achieves the lowest loss on the test set, this difference can be attributed to the randomness inherent in training, with a relative difference to both the final and pretrained models of less than 0.0004% and 0.07%, respectively.

	Test loss			Test accuracy (%)		
DNN	Pretrained	Final	SWA	Pretrained	Final	SWA
CLUSTER						
GCN	1.660918	1.659778	**1.659771**	**30.50014**	30.4911	30.4946

Table 3: Node classification metrics

5.4 Loss surface visualization

The visualization procedure to plot the loss surface over a 2D representation of the space of weights is defined in [2]. We will employ the same representation to gain a deeper insight into the behavior of SWA.

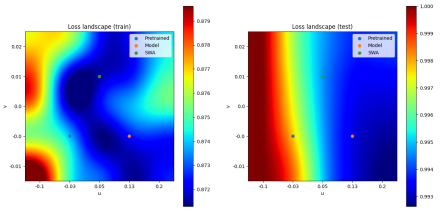

Figure 3: Loss surface of MobileNet on CIFAR100

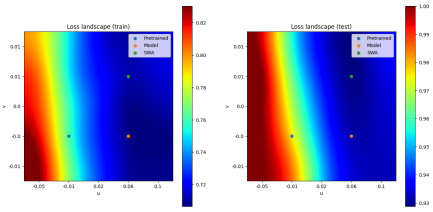

Figure 4: Loss surface of a CNN on MNIST

In these figures, the term "Pretrained" refers to the model trained with Adam until convergence, "Model" indicates the latest trained version (including SWA), and SWA represents the model with weights corresponding to SWA.

We observed that one issue with SWA is that the loss surface differs between the training set and the test set. This discrepancy is particularly noticeable on CIFAR100 due to the dataset's higher complexity compared to MNIST.

SWA weights may not necessarily locate a broader region of loss minimization. Additionally, it's essential to note that this visualization provides a simplified representation of the loss surface as we reduce the space of weights, spanning over 1 billion parameters, into a mere 2D representation. In reality, we anticipate the loss surface to be highly non-convex, which raises questions about the relevance of the SWA approach.

6 Conclusion

In conclusion, our experimentation with Stochastic Weight Averaging (SWA) across various types of data, including images, tabular regression, and graphs, did not yield practical results. Despite the appealing concept of SWA potentially approximating ensembling, our experiments were unable to replicate the findings of the original paper. Instead, we observed minimal to zero performance improvements with SWA, which we attribute possibly to random fluctuations. Overall, our findings suggest that SWA may not be a practical or effective approach in our experimental contexts.

References

[1] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization, 2019.

[2] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns, 2018.

[3] Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, 19(1):1–67, 1991.

[4] Leo Breiman. Bagging predictors. *Machine learning*, 24:123–140, 1996.

[5] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[6] Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.