Closing the K-FAC Generalisation Gap Using Stochastic Weight Averaging

Xingchen Wan, Diego Granziol, Stefan Zohren, Stephen Roberts
Machine Learning Research Group, University of Oxford
`{xwan, diego, zohren, sjrob}@robots.ox.ac.uk`

Abstract

In this paper we present a novel adaptation of Stochastic Weight Averaging (SWA) in the context of the second order optimisation of deep neural networks. We propose to combine SWA with Kronecker-factored Approximate Curvature (K-FAC), and we empirically show, by testing on various neural network architectures on the CIFAR-100 data-set, that we close some of the generalisation gaps with negligible additional computational cost and no negative impact on the efficiency of the algorithm. We also find that the SWA variant of K-FAC outperforms different variants of SGD and Adam in test accuracy.

1 Introduction

The unprecedented success of deep learning in a wide range of tasks has motivated the development of increasingly sophisticated network architectures, which in turn necessitates more accurate and efficient optimisation algorithms for training. To this end, second-order optimisation techniques have been suggested as an efficient alternative to the conventional first-order optimisation techniques such as the Stochastic Gradient Descent (SGD): most remarkably, by leveraging the local curvature information, second-order optimisers can make much more progress per iteration compared to SGD [1]. Recent algorithms, such as the Kronecker-factored Approximate Curvature (K-FAC) [7] and its variants [3], also use various approximations to the surrogate of the Hessian matrix used in second-order updates to reduce the computational cost to a level only several times more expensive than SGD updates. While these recent works have greatly advanced the proposition of the usage of second-order optimisers in neural network training, the effective leveraging of second-order information in generalisation remains noticeably elusive.

In this paper, we propose a novel adaptation of the Stochastic Weight Averaging (SWA) algorithm [5] in the context of second-order optimisation. Theoretically, we argue SWA improves generalisation via weight reduction in second-order optimisation and beyond. Experimentally, in various network architectures and on the CIFAR-100 data-set, we demonstrate that with negligible additional computational cost and no impact on the convergence efficiency, SWA is capable of improving the generalisation performance of second-order optimiser in level that is similar to, if not more than, the first-order counterparts. We also show that our proposed algorithm performs competitively to SGD, Adam and their respective SWA variants.

2 Background

Network Training as an Optimisation Problem Consider input $\boldsymbol{x} \in \mathbb{R}^{d_x}$, output $\boldsymbol{y} \in \mathbb{R}^{d_y}$ and a family of prediction functions parameterised by a weight vector $\boldsymbol{w} \in \mathbb{R}^P$: $\mathcal{H} := \{h(\cdot; \boldsymbol{w})\}$ where each prediction function $h(\cdot;\cdot) : \mathbb{R}^{d_x} \times \mathbb{R}^P \to \mathbb{R}^{d_y}$ has a loss $l(h(\boldsymbol{x}; \boldsymbol{w})\boldsymbol{y}) : \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \to \mathbb{R}$, the true risk is given by the expectation integral:

$$R_{true}(\boldsymbol{w}) = \int_{\mathbb{R}^{d_x} \times \mathbb{R}^{d_y}} l(h(\boldsymbol{x};\boldsymbol{w}), \boldsymbol{y}) dP(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}[l(h(\boldsymbol{x};\boldsymbol{w}), \boldsymbol{y})] \tag{1}$$

where the expectation $\mathbb{E}$ is with respect to the underlying data-generating distribution to which we cannot access directly. For a data-set of with N training input-output samples, we only have access to the empirical risk R_{emp}, given by:

$$R_{emp}(w) = \frac{1}{N} \sum_{i=1}^{N} l(h(\boldsymbol{x}_i;\boldsymbol{w}), \boldsymbol{y}_i) \tag{2}$$

and the minimisation of (2) is considered to be the practical optimisation problem of network training.

First- and Second-order Methods With R_{emp} in (2) as the objective function and without considering additional features such as regularisation and momentum, the k-th iteration of first-order optimisers is generally given in the form of:

$$\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k - \alpha_k \nabla L_k(\boldsymbol{w}) \tag{3}$$

where α is the learning rate and $\nabla L(\boldsymbol{w})$ is, for example, the full gradient, the batch gradient or the stochastic gradient, depending on the choice of the optimiser. On the other hand, for second-order optimisers, the iteration is generally in the form of:

$$\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k - \alpha_k \bar{\boldsymbol{H}}^{-1} \nabla L_k(\boldsymbol{w}) \tag{4}$$

where for the general case of the optimisation of a non-convex objective function, $\bar{\boldsymbol{H}}$ is some positive-definite surrogate of the Hessian $\boldsymbol{H}$. Examples of such a surrogate include the Generalised Gauss-Newton Approximation used in Hessian-free optimisation methods [6] and the Fisher Information Matrix (FIM) $\boldsymbol{F}$ used in K-FAC [7].

Kronecker-Factored Approximate Curvature (K-FAC) Noting that $\boldsymbol{w} \in \mathbb{R}^P$ where P is the number of parameters in the neural network and that $\bar{\boldsymbol{H}} \in \mathbb{R}^{P \times P}$, inverting $\bar{\boldsymbol{H}}$ (or equivalently in the context of K-FAC, $\boldsymbol{F}$) is intractable[1] considering that modern neural network typically contains ten to hundred millions of parameters and this is one of the key obstacles limiting the application of second-order optimisers in neural network training. To avoid the explicit inversion of FIM, K-FAC uses a block diagonal approximation to the FIM where each block $\boldsymbol{F}_i$ represents the i-th layer of the network. K-FAC then proceeds to approximate $\boldsymbol{F}_i$ as a Kronecker product:

$$\boldsymbol{F}_i = \mathbb{E}[\nabla L(\boldsymbol{w}) \nabla L(\boldsymbol{w})^T] \approx \boldsymbol{A} \otimes \boldsymbol{B} \tag{5}$$

where $\boldsymbol{A} = \mathbb{E}(\boldsymbol{h}\boldsymbol{h}^T)$ and $\boldsymbol{B} = \mathbb{E}(\boldsymbol{\delta}\boldsymbol{\delta}^T)$ are smaller matrices that are much easier to compute ($\boldsymbol{h}$ and $\boldsymbol{\delta}$ are the input and gradient of the i-th layer, respectively). By virtue of the Kronecker product identity $(\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1}$, K-FAC allows the previously intractable inversion to be approximated, thereby allowing the second-order updates in (4) to be executed efficiently [3, 7].

Stochastic Weight Averaging (SWA) SWA [5] was proposed as a modification to the conventional SGD procedure. The history of averaging weights in gradient methods is extensive [2, 8]. It is required for all the proofs of convergence of SGD in order to reduce the noise and stop oscillating around the minimum, and such effect can under certain step size schemes give similar behaviour in second-order methods [1]. SWA uses a constant or cyclical learning rate schedule to explore the weight space, and then takes a average of the weights obtained at the end of each epoch instead of converging to a single point in the weight space. SWA is shown to improve SGD generalisation performance, as it has been argued that SGD often converge to the boundary of the set of high-performing solutions [5] whereas by taking averages, SWA is capable of converging to the centre of that set which is more robust to the shifts between training and testing distributions. Furthermore, since the only additional computational effort of is to keep a running average of the SGD weights during the training, there is little extra computational cost involved.

[1]It has computational complexity $\mathcal{O}(P^3)$.

3 SWA in Second-order Optimisation

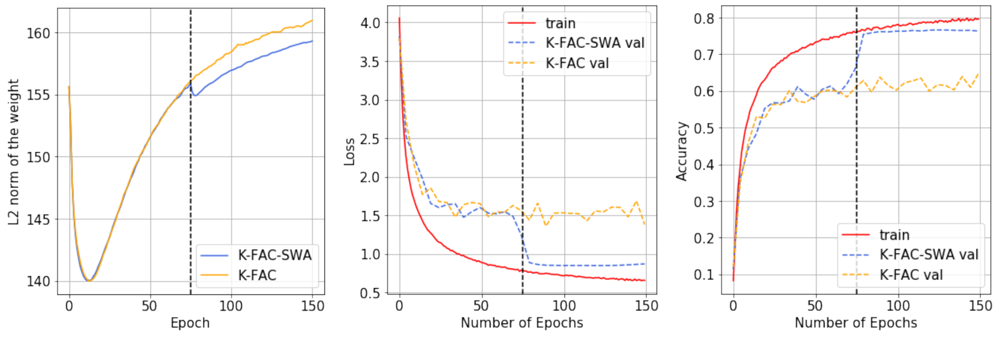
We hypothesise that in addition to the aforementioned mechanisms, SWA might also improve the generalisation performance through weight reduction. Denoting the averaged SWA solution in the weight space as the *SWA point*, we argue that SWA reduces the norm of the weight, with the following theorem:

Theorem 1. *The $L2$ norm of the SWA point in the weight space that averages over T epochs, is smaller than or equal to the Césaro mean of their $L2$ norms.*

$$\left(\frac{1}{T}\sum_{i=1}^{T}\boldsymbol{w}_i\right)^2 \leq \frac{1}{T}\sum_{i=1}^{T}\boldsymbol{w}_i^2 \tag{6}$$

Proof. See Appendix. □

With Theorem 1 established, we note that weight-reducing techniques, such as $L2$ regularisation and weight decay, have been already known to improve generalisation [10]. This mechanism of action is not optimiser-specific, and we find its application in the context of second-order optimisation to be particularly desirable, as it could potentially improve generalisation without much additional implementation complexity or efficiency impact. To empirically validate our claims, we run an experiment to investigate the changes in the $L2$ norm of the weights during the course of K-FAC optimisation with and without SWA. We use a constant learning rate of 0.03 for 150 epochs and for K-FAC-SWA, we activate SWA from the 75th epoch onward. We perform the experiment on CIFAR-100 data-set with Preactivation ResNet110 (PreResNet110) [4]. The result of the experiment, as presented in Figure 1, further validates our claim as SWA is shown to reduce the $L2$ norm of weight and the test loss while improving the test accuracy.



(a) $L2$ norm of weights (b) Train/validation loss (c) Train/validation acc.

Figure 1: Effect of SWA on PreResNet110 CIFAR-100 network performance. The vertical dashed line denotes the epoch from which SWA is activated.

4 Experiments

Experimental Setup We investigate the performance of K-FAC, SGD with Momentum and Adam on CIFAR-100 (45,000 training samples and 5,000 test samples) with PreResNet110 [4] and VGG16 [9] networks, both with and without SWA. To improve training performance, we apply batch normalisation to all experiments and regularise with a weight decay parameter of 5×10^{-4}. We further use a batch size of 128 for all experiment runs.

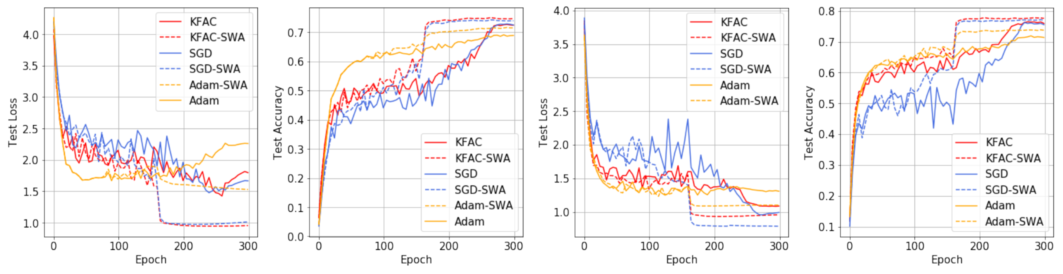
Learning Rate Schedule For all experiments without SWA, we use the following learning rate schedule for the learning rate at the t-th epoch:

$$\alpha_t = \begin{cases} \alpha_0, & \text{if } \frac{t}{T} \leq 0.5 \\ \alpha_0[1 - \frac{(1-r)(\frac{t}{T}-0.5)}{0.4}] & \text{if } 0.5 < \frac{t}{T} \leq 0.9 \\ \alpha_0 r, & \text{otherwise} \end{cases} \tag{7}$$

where α_0 is the initial learning rate, $T = 300$ is the total number of epochs budgeted and $r = 0.01$ is the learning rate ratio. For all experiments with SWA, we start computing the weight averaging from the 161st epoch and the learning rate after SWA is activated is set to be a constant value of $0.5\alpha_0$.

Hyperparameter Tuning For all SGD and K-FAC runs, we set the momentum parameter to be 0.9 whereas for Adam, we set $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon = 10^{-8}$, their default values. On K-FAC, we set the frequency of inversion of FIM to be once per 100 iterations, and we conduct a grid-search over different combinations of initial learning rates $\alpha_0 = \{0.1, 0.03, 0.01, 0.003\}$ and damping coefficients $d = \{0.1, 0.03, 0.01, 0.003\}$. On SGD and Adam, we search over different initial learning rates $\alpha_0 = \{0.1, 0.03, 0.01, 0.003\}$ and $\alpha_0 = \{0.005, 0.001, 0.0005\}$, respectively.

Experimental Results Figure 2 shows the test performance of the various optimisers on the CIFAR-100 data-set against the number of epochs and Table 1 summarises the Top-1 test accuracy and the test loss at the end of training. We observe that SWA improves the test accuracy and reduces the test loss of all optimisers in both architectures without compromising the efficiency: it increases the test accuracy of SGD by 1.2-2%, Adam by 2.5-3.5% and K-FAC by 1.3-2.4%. SWA improves K-FAC by a larger margin than SGD, and K-FAC-SWA achieves the best test accuracy in both architectures although K-FAC is not necessarily always the best-performing in non-SWA optimisers. Another observation is that in terms of test loss, K-FAC-SWA does not necessarily perform much better than SGD-SWA (for the case of PreResNet110, much worse). This seems to imply that despite of K-FAC-SWA achieving better test accuracy, SGD-SWA still produces a better calibrated model at the end of the training.



(a) VGG16 loss (b) VGG16 acc. (c) PreResNet110 loss (d) PreResNet110 acc.

Figure 2: Test Performance by Number of Epochs

Table 1: Comparison of the Top-1 Test Accuracy and the Test Loss

Optimiser	Top-1 Accuracy		Test Loss	
	VGG16	PreResNet110	VGG16	PreResNet110
K-FAC	72.7	76.5	1.79	1.09
K-FAC-SWA	**75.1**	**77.8**	**0.955**	0.958
SGD	72.9	76.3	1.67	0.993
SGD-SWA	74.9	77.5	1.01	**0.787**
Adam	69.2	71.9	2.26	1.31
Adam-SWA	71.7	75.4	1.53	1.10

5 Conclusion

In this paper we investigate the application of SWA in the context of second-order optimisation, we find it to improve generalization performance and by a larger margin than for SGD in terms of test accuracy. We further give a new interpretation of the mechanism of action of SWA. In the future, we look forward to further validating the results presented in this paper with more extensive experiments on different optimisers, hyperparameters, data-sets and network architectures. We also hope this work will be the starting point leading to better understanding and leveraging of the weight reduction techniques, including but not limited to SWA, in second-order optimisation.

References

[1] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[2] J. C. Duchi. Introductory lectures on stochastic optimization. *The Mathematics of Data*, 25:99, 2018.

[3] T. George, C. Laurent, X. Bouthillier, N. Ballas, and P. Vincent. Fast approximate natural gradient descent in a Kronecker factored eigenbasis. In *Advances in Neural Information Processing Systems*, pages 9550–9560, 2018.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[6] J. Martens. Deep learning via Hessian-free optimization. In *ICML*, volume 27, pages 735–742, 2010.

[7] J. Martens and R. Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015.

[8] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[9] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[10] G. Zhang, C. Wang, B. Xu, and R. Grosse. Three mechanisms of weight decay regularization. *arXiv preprint arXiv:1810.12281*, 2018.

A Appendix

Proof of Theorem 1

The $L2$ norm of the SWA point in the weight space that averages over T epochs, is smaller than or equal to the Césaro mean of their $L2$ norms.

$$\left(\frac{1}{T}\sum_{i=1}^{T}\boldsymbol{w}_i\right)^2 \leq \frac{1}{T}\sum_{i=1}^{T}\boldsymbol{w}_i^2 \tag{8}$$

Proof. For $T = 2$

$$\frac{(\boldsymbol{w}_1+\boldsymbol{w}_2)^2}{4} \leq \frac{\boldsymbol{w}_1^2+\boldsymbol{w}_2^2}{2} \tag{9}$$

$$0 \leq (\boldsymbol{w}_1-\boldsymbol{w}_2)^2 \tag{10}$$

Assume for $T = n$, for $T = n+1$ i.e. $(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{w}_i)^2 \leq \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{w}_i^2$

$$\left(\frac{1}{n+1}\sum_{i=1}^{n+1}\boldsymbol{w}_i\right)^2 \leq \frac{1}{n}\sum_{i=1}^{n+1}\boldsymbol{w}_i^2 + \sum_{i=1}^{n}\boldsymbol{w}_i^2 \tag{11}$$

$$\left(\sum_{i=1}^{n}\boldsymbol{w}_i\right)^2 + 2\boldsymbol{w}_{n+1}\sum_{i=1}^{n}\boldsymbol{w}_i + \boldsymbol{w}_{n+1}^2 \leq n\sum_{i=1}^{n}\boldsymbol{w}_i^2 + \sum_{i=1}^{n}\boldsymbol{w}_i^2 + (n+1)\boldsymbol{w}_{n+1}^2 \tag{12}$$

$$0 \leq \sum_{i=1}^{n}(\boldsymbol{w}_i-\boldsymbol{w}_{n+1})^2 \tag{13}$$

and by mathematical induction Theorem 1 follows. □