Averaging Weights Leads to Wider Optima and Better Generalization

Liad Nahum, Lishay Aben Sour

May 2021

1 Sage 1 - Selecting the algorithm for evaluation

We chose the paper "Averaging Weights Leads to Wider Optima and Better Generalization" by Pavel Izmailov Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, Andrew Gordon Wilson [1].

1.1 Paper Summery

Deep neural networks are typically trained by optimizing a loss function with an SGD variant. Typically, the learning rate of those networks is decaying until convergence. In this paper is presented an algorithm SWA - Stochastic Weight Averaging, which argues that simple averaging of multiple points along the trajectory of SGD, with a cyclical or constant learning rate, leads to better generalization than conventional training with almost no computational overhead.

The pseudo-code of SWA can be seen in Figure 1. The algorithm works as follows:

1. First, we start with a pre-trained model with weights, $\hat{w}$, that was trained using conventional SGD, up to 1 *Budget* or up to a reduced percent of the *Budget*. While *Budget* is defined by the authors as the number of epochs required to train the model with the conventional SGD procedure. (In our implementation of SWA, we trained the model using the conventional SGD, up to 0.75 *Budget*)

2. Then, we continue training, using a cyclic learning rate or a constant learning rate schedule for n iterations.

   Using the cyclic learning rate we have several hyper-parameters: α_1, α_2 and the cycle length, c. In each cycle, we start from a relatively high learning rate α_1 and decrease gradually to α_2. We refer this procedure as $\alpha(i)$. Using the constant learning rate, $\alpha(i) = \alpha_1$. And $c = 1$.

   The main difference between the two approaches is that the individual proposals of SGD with a cyclical learning rate schedule are in general

Algorithm 1 Stochastic Weight Averaging

Require:
weights $\hat{w}$, LR bounds α_1, α_2,
cycle length c (for constant learning rate $c = 1$), number of iterations n
Ensure: w_{SWA}
$w \leftarrow \hat{w}$ {Initialize weights with $\hat{w}$}
$w_{\text{SWA}} \leftarrow w$
for $i \leftarrow 1, 2, \ldots, n$ **do**
  $\alpha \leftarrow \alpha(i)$ {Calculate LR for the iteration}
  $w \leftarrow w - \alpha \nabla \mathcal{L}_i(w)$ {Stochastic gradient update}
  if $\mathrm{mod}(i, c) = 0$ **then**
    $n_{\text{models}} \leftarrow i/c$ {Number of models}
    $w_{\text{SWA}} \leftarrow \frac{w_{\text{SWA}} \cdot n_{\text{models}} + w}{n_{\text{models}} + 1}$ {Update average}
  end if
end for
{Compute BatchNorm statistics for w_{SWA} weights}

Figure 1: SWA algorithm

more accurate than the proposals of a fixed-learning rate SGD. After making a large step, SGD with a cyclical learning rate spends several epochs fine-tuning the resulting point with a decreasing learning rate. SGD with a fixed learning rate on the other hand is always making steps of relatively large sizes, exploring more efficiently than with a cyclical learning rate, but the individual proposals are worse.

Using those kind of learning rates values we get a higher spread on the test error surfaces of a Preactivation as can be seen in Figure 2.

3. At each iteration, i, we apply the chosen learning rate schedule and then apply a stochastic gradient update on the weights of the **pre-trained model**. At the end of a cycle, we update the weights of our **swa model** to be the average of the swa model's weights so far, w_{swa} and the weights of the pre-trained model, updated during the cycle, w.

1.2 Advantages of the Algorithm

- According to the paper the Stochastic Weight Averaging (SWA) algorithm relatively easy to implement and use with every type of network due to the fact that the main part of the deep learning containing the Stochastic Gradient Descend (SGD) doesn't change, and all needed to be done in order to implement SWA, is to add the weights averaging part at the end (can be added as a pre-implemented library).

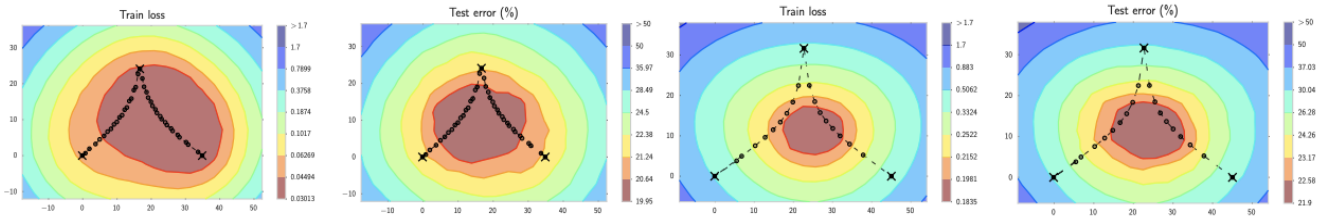

Figure 2: The L2-regularized cross-entropy train loss and test error surfaces of a Preactivation ResNet-164 on CIFAR100 in the plane containing the first, middle and last points (indicated by black crosses) in the trajectories with (left two) cyclical and (right) constant learning rate schedules

Figure 3: L2-regularized cross-entropy train loss and test error as a function of a point on the line connecting SWA and SGD solutions on CIFAR-100 using VGG16.

- Improves generalization, SWA leads to much wider solutions - in SGD the weights values lies near the boundary of a wide flat region of the train loss surface graph. Further, the loss is very steep near SGD. With SWA the weights of the solution are closer to the minimum point of the loss surface graph and creates a more wider graph which implies of finding wider solution and will be more robustness against generalized data, As can be seen in Figure 3.
- Except for calculation the averages which take slightly more time than in regular network the rest of the implementation has no computational time overhead, and therefore can easily be considered as an improvement to any neural network.

1.3 Disadvantages of the Algorithm

- The training loss for SWA is often slightly worse than for SGD suggesting that SWA solution is not a local optimum of the loss.
- During training, the algorithm must maintain a copy of the running average of DNN weights.
- Despite SWA is different from Fast Geometric Ensembling (FGE) by that FGE uses ensemble of different models and the SWA corresponds to a single model, the results of the SWA does not improve the results of the FGE at some of the data sets.

2 Stage 2 - Suggesting an improvement

In the paper, the authors applied several versions of the SWA algorithm. They ran the whole model for 1 *Budget*, 1.25 *Budgets* and *1.5 Budgets* number of epochs. The best results were given by the version of *1.5 Budgets*. While in each version, they trained the model for 0.75 *Budget* using the conventional SGD, and the rest using the SWA algorithm. So, we implemented the SWA 1.5 *Budget* version while 0.75 *Budget* we trained by SGD and the rest by SWA.

Our improvement is in running time. We let the whole algorithm run for only 1 *Budget* but we start the SWA phase earlier. We train the network using SGD for only 50% the state of art amount of epochs instead of 75% which was suggested by the paper and then start the SWA phase. And instead of using the same learning rate for the SWA phase and for the SGD phase as was implemented in paper's experiments, we chose a higher learning rate for the SWA phase. The thing we hoped to achieve by doing that is that the relatively higher learning rate will let the network get to the minimum error rate in the error surface faster than with the regular learning rate and thus will have the ability to compensate on the lower amount of epochs, but will also achieve a lover error than the one we can get SGD thanks to the weights averaging that will hopefully despite the large steps a higher learning rate takes in the error surface, will eventually be averaged to the minimum of the error rate together will the ability to find a wider solution. We wanted to achieve the advantages of the original SWA with the relatively low running time of the SGD algorithm.

In Figure 4, we can see an example for our expectations regarding the error surface with the improvement algorithm. In addition, we can see that since we use averaging it is reasonable to have this marked circle since the last value of the weights will be the middle of the circle and not one of the points on it. Furthermore it should save time in the convergence because the higher learning rate will get us to the circle faster than the SGD.

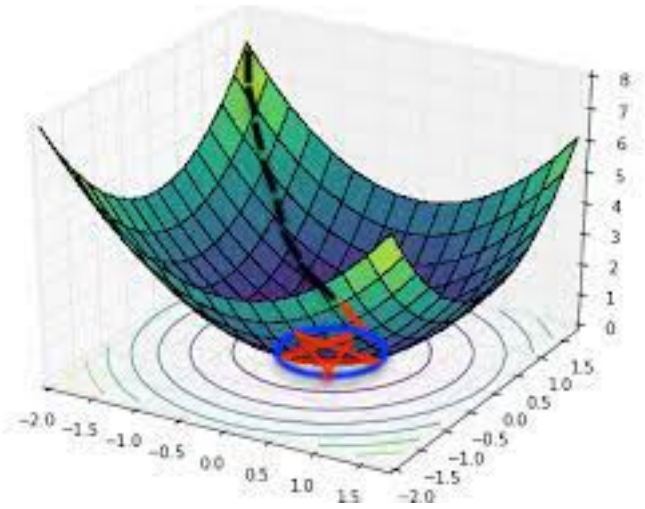

Figure 4: The black lines shows the SGD convergent into the min error location. The red lines shows the larger steps made by the higher learning rate of the SWA which will make the network create points at the area surrounding the minimum error rate and create the blue circle, with the averaging the result should be the middle of the circle which is the wanted result.

3 Stage 3: Select a well-known algorithm for comparison

We chose to compare the SWA algorithm with the SGD (Stochastic Gradient Descent) conventional algorithm for the loss function optimization. In Figure 5 is described SGD pseudo code as presented in [3].

4 Stage 4: Evaluating the algorithms you selected in stages 1, 2 and 3

4.1 Versions

SWA is based on averaging the samples proposed by SGD using a learning rate schedule that allows exploration of the region of weight space corresponding to high-performing networks. The algorithm has two main versions using cyclic learning rate or a constant learning rate schedules. According to the paper, each version has its own advantages but the results are similar. We used the version with the constant learning rate.

In addition, recall that *Budget* is defined as the number of epochs required to train the model with the conventional SGD procedure. And recall that the

Algorithm 2 Stochastic Gradient Descent

Require: Training Set $\mathcal{T}$; Learning Rate η; Normal Distribution Std: σ.
Ensure: Model Parameter $\boldsymbol{\theta}$

Initialize parameter with Normal distribution $\boldsymbol{\theta} \sim N(0, \sigma^2)$
Initialize convergence $tag = False$
while $tag == False$ **do**
  Shuffle the training set $\mathcal{T}$
  for each data instance $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{T}$ **do**
    Compute gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; (\mathbf{x}_i, \mathbf{y}_i))$ on the training instance $(\mathbf{x}_i, \mathbf{y}_i)$
    Update variable $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; (\mathbf{x}_i, \mathbf{y}_i))$
  end for
  if convergence condition holds **then**
    $tag = True$
  end if
end while
Return model variable $\boldsymbol{\theta}$

Figure 5: SGD pseudo code

authors had several versions of their SWA algorithm while the versions were different in the overall number of epochs: SWA 1 *Budget*, SWA 1.25 *Budgets* and SWA 1.5 *Budgets*. We used the version of SWA 1.5 *Budgets* because it achieved the best results.

4.2 Data

In the paper, they used only convolution neural network models on images datasets. The evaluation on those datasets is very expensive in terms of running time and memory consumption. Therefore, we had hard time in finding matching datasets for our project and in evaluating them. Eventually, we succeeded to evaluate 11 different datasets using Google Collaboratory utilities and resources. Among them: CIFAR10, CIFAR100, EMNIST, FashionMNIST, KMNIST, MNIST, QMNIST, STL10, USPS, SVHN, SEMEION. We think this is enough for analysis and conclusions because those datasets are very used and diverse from each other. The datasets are taken from the torchvision datasets `https://pytorch.org/vision/stable/datasets.html` . In practice, the deep learning model we used for our evaluation is VGG16.

4.3 Hyperparameter Optimization

The hyper-parameters we chose are: initial learning rate of the SGD phase and number of epochs. The initial learning rate is a very significant parameter for all kind of SGD algorithms. A learning rate that is too big may lead to missing the optimum while a too small learning rate may lead to a very slow convergence. In addition, number of epochs is a significant hyper-parameters for

deep learning models, because as can be seen in the experiments, during the epochs, the metrics values change. It is important to set a number of epochs so the running will not stop before convergence but also won't over-fit.

The three algorithms we are evaluating strongly depend on the SGD algorithm, and there is a strong relation between the hyper-parameters of each algorithm. Therefore, we apply the hyper-parameters optimization only on the SGD algorithm and then we determined the initial learning rate and the number of epochs of SWA and of the improved algorithm using determined relations. In order to find the best hyper-parameters for SGD, we used Random Search optimization. In each outer k-fold, we apply random search for 50 times. In each time, we randomly choose a pair of [lr_init, num_epochs] out of the following space $lr\_init = range(0.01, 0.1, 0.01)$ and $num\_epochs = range(50, 300, 25)$. Then, we used an inner fold with $k = 3$, evaluated the model and take an average of the accuracy results among the 3 folds. Finally, after 50 trials, we choose the hyper-parameters which gain the best accuracy average. We do this process for all 10 outer folds and eventually choose the best hyper-parameters out of all outer and inner folds.

With the SGD best hyper-parameters, we determined the ones for SWA and the improved algorithm. For the SGD algorithm, the number of epochs is 1 *Budget*. For the SWA algorithm, we trained the net for 1.5 *Budgets* of the number of epochs chosen by the random search optimization for SGD. In the SWA, We ran the SGD part of the algorithm for 0.75 *Budget* and then started to average the weights for the rest of the epochs until 1.5 *Budgets*. The learning rate remains the same as the learning rate chosen for the SGD for both part of the training. For the improved algorithm, we ran in total 1 *Budget* which is split for 0.5 of the budget as the SGD and the rest 0.5 budget as the SWA with a higher learning rate than the learning rate chosen by the SGD random search.

Finally, to evaluate the results on all the algorithms with the best hyper-parameters we evaluated each algorithm using 10 folds with the same best hyper-parameters as determined by the random search and the relation explained above. Those results are reported in the attached file "results.xlsx".

4.4 Performance metrics for evaluation

All datasets we evaluated are multi-class. Therefore, we calculated the metrics in one-vs-rest approach. All the results are reported in the file "results.xlsx". For each dataset(appears in different color in the file), we calculated the metrics using 3 different algorithms: SGD which is the well-known algorithm for comparison that we chose, SWA which is the suggestion of the authors and our suggested improved algorithm.

5 Stage 5: Statistical significance testing of the results

We chose the accuracy (ACC) metric for our statistical significance testing. The results for each dataset are reported in Figure 6. We applied two kinds of tests on the Accuracy metric: First, Friedman test. Second, if according to the Friedman test the results are statistically significant, we applied a Post-Hoc test which is called tukey test. As can be seen, in all datasets except for 'SEMEION', the results were statistically significant, so we also applied the Post-Hoc test on them.

Conclusions:

- SWA vs SGD: at 5 of 11 the SWA improved the SGD algorithm and the hypothesis of having the same distribution of the accuracy metric was rejected. In the 'KMNIST' data-set the improvement we got was the most significant and had a mean difference of 0.0029. Furthermore, in this data-set the H_0 hypothesis was rejected with confidence of 99.9% (i.e with 0.001 p-value). On the other hand in the 'STL10' data-set, for example, the SWA decreased the accuracy on a mean difference of -0.0194 which implies that despite the ability of the algorithm to improve the state of art at some cases as was shown in the article, there are cases that it doesn't improve.

- Improved vs both SGD and SWA: at 4 out of the 11 data-sets tested the improved algorithm is better than both SGD and SWA and rejected the hypothesis of the algorithms to be the same. The highest improvement on the mean difference was 0.0129 and the highest confidence of rejecting the H_0 hypothesis was 99.9% in the 'USPS' data-set. At the rest of the data-sets the improvement algorithm decreased the performance and at the 'SEMEION' data-set all three hypothesis were not rejected which means that on this data-set all three algorithms had similar performance distribution.

- Improved vs SWA: at 5 out of 11, the improved algorithm we created gave higher accuracy than the SWA .

- Improved vs SGD: at 5 out of 11, the improved algorithm's accuracy is higher than SGD. Those results regarding our improvement algorithm indicates that our improvement, like the SWA algorithm shown in the paper, improved the results on some of the data-sets and confirms the logic in our expectations regarding its performance was sometime correct and like the idea of SWA itself should be considered when training neural networks.

```
Dataset MNIST
Different distributions (reject H0)
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
 group1  group2 meandiff p-adj  lower   upper  reject
-----------------------------------------------------
Improved    SGD   0.0003 0.001  0.0002  0.0003   True
Improved    SWA  -0.0004 0.001 -0.0004 -0.0003   True
     SGD    SWA  -0.0006 0.001 -0.0007 -0.0006   True
-----------------------------------------------------

Dataset QMNIST
Different distributions (reject H0)
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
 group1  group2 meandiff p-adj   lower   upper  reject
-----------------------------------------------------
Improved    SGD  -0.0008  0.001  -0.001 -0.0006   True
Improved    SWA  -0.0001 0.7024 -0.0003  0.0001  False
     SGD    SWA   0.0007  0.001  0.0005  0.0009   True
-----------------------------------------------------

Dataset STL10
Different distributions (reject H0)
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
 group1  group2 meandiff p-adj  lower   upper  reject
-----------------------------------------------------
Improved    SGD   0.0294 0.001  0.0253  0.0334   True
Improved    SWA     0.01 0.001  0.0059   0.014   True
     SGD    SWA  -0.0194 0.001 -0.0235 -0.0153   True
-----------------------------------------------------

Dataset USPS
Different distributions (reject H0)
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
 group1  group2 meandiff p-adj  lower   upper  reject
-----------------------------------------------------
Improved    SGD  -0.0011 0.001 -0.0016 -0.0007   True
Improved    SWA  -0.0129 0.001 -0.0133 -0.0124   True
     SGD    SWA  -0.0118 0.001 -0.0122 -0.0113   True
-----------------------------------------------------

Dataset SVHN
Different distributions (reject H0)
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
 group1  group2 meandiff p-adj  lower   upper  reject
-----------------------------------------------------
Improved    SGD  -0.0017  0.001 -0.002 -0.0015   True
Improved    SWA   0.0004 0.0029 0.0001  0.0007   True
     SGD    SWA   0.0022  0.001 0.0019  0.0024   True
-----------------------------------------------------

Dataset SEMEION
Same distributions (fail to reject H0)
```

(a)

```
Dataset CIFAR10
Different distributions (reject H0)
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
 group1  group2 meandiff p-adj  lower   upper  reject
-----------------------------------------------------
Improved    SGD   0.0061  0.001 0.0056  0.0066   True
Improved    SWA   0.0055  0.001  0.005   0.006   True
     SGD    SWA  -0.0005 0.0266 -0.001 -0.0001   True
-----------------------------------------------------

Dataset CIFAR100
Different distributions (reject H0)
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
 group1  group2 meandiff p-adj lower  upper  reject
-----------------------------------------------------
Improved    SGD   0.0071 0.001 0.0067 0.0074   True
Improved    SWA   0.0092 0.001 0.0088 0.0095   True
     SGD    SWA   0.0021 0.001 0.0018 0.0024   True
-----------------------------------------------------

Dataset EMNIST
Different distributions (reject H0)
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
 group1  group2 meandiff p-adj   lower   upper  reject
-----------------------------------------------------
Improved    SGD  -0.0002  0.001 -0.0003 -0.0001   True
Improved    SWA  -0.0001 0.0139 -0.0002    -0.0   True
     SGD    SWA   0.0001  0.001     0.0  0.0002   True
-----------------------------------------------------

Dataset FashionMNIST
Different distributions (reject H0)
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
 group1  group2 meandiff p-adj   lower  upper  reject
-----------------------------------------------------
Improved    SGD   0.0019  0.001  0.0016 0.0022   True
Improved    SWA   0.0015  0.001  0.0012 0.0019   True
     SGD    SWA  -0.0004 0.0305 -0.0007   -0.0   True
-----------------------------------------------------

Dataset KMNIST
Different distributions (reject H0)
 Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
 group1  group2 meandiff p-adj  lower   upper  reject
-----------------------------------------------------
Improved    SGD   -0.003 0.001 -0.0036 -0.0024   True
Improved    SWA  -0.0001   0.9 -0.0007  0.0005  False
     SGD    SWA   0.0029 0.001  0.0023  0.0035   True
-----------------------------------------------------
```

(b)

Figure 6: Statistical significance testing for each dataset

DNN (Budget)	SGD	FGE (1 Budget)	SWA		
			1 Budget	1.25 Budgets	1.5 Budgets
		CIFAR-100			
VGG-16 (200)	72.55 ± 0.10	74.26	73.91 ± 0.12	74.17 ± 0.15	74.27 ± 0.25
		CIFAR-10			
VGG-16 (200)	93.25 ± 0.16	93.52	93.59 ± 0.16	93.70 ± 0.22	93.64 ± 0.18

Figure 7: True Positive rate (%) of SWA, SGD and FGE methods on CIFAR-100 and CIFAR-10 datasets for different training budgets.

6 Stage 6: Summary and Conclusions

We implemented the project in python, using pytorch library for deep learning. The authors attached a GitHub repository [2], which contains a limited implementation of the SWA algorithm. We extended this implementation according to the project's instructions. Our implementation is available on `https://github.com/LiadNahum1/FinalProject`.

Experimental Conclusions: In conclusion, we successfully implemented the algorithm suggested in the paper. We reported evaluations on 11 well-known and publicly used data-sets, on three algorithms: conventional SGD, SWA and our improved SWA. During our evaluations, we reported several metrics results as requested in the assignment and also applied statistical significance tests on the accuracy metric. As shown, at some datasets the SWA indeed improved accuracy compared to SGD and at some of the datasets, our improvement produced higher results than both SWA and SGD.

In addition, in our implementation, we reproduce the results of the paper on the CIFAR10 and CIFAR100 data-sets for the 1.5 *Budget* version referring the TPR metric. By our results, for CIFAR100 dataset, the SGD algorithm gave average of 61.9, the SWA gave average of 72.84 while in the paper the results were similar having the True Positive Rate of 72.56 in the SGD and 74.27 in the SWA as can be seen in Figure 7. For CIFAR10 dataset, the SGD algorithm gave average of 92.8, the SWA gave average of 92.6 while in the paper the results were similar as can also be seen in Figure 7.

References

[1] Pavel Izmailov, Dmitrii Podoprikhin,Timur Garipov, Dmitry Vetrov, Andrew Gordon Wilson *Averaging Weights Leads to Wider Optima and Better Generalization*.2019. https://arxiv.org/pdf/1803.05407.pdf

[2] https://github.com/timgaripov/swa

[3] Jiawei Zhang. *Gradient Descent based Optimization Algorithms for Deep Learning Models Training*. 2019.

[4] https://pytorch.org/vision/stable/datasets.html

[5] https://www.reneshbedre.com/blog/anova.html

[6] https://machinelearningmastery.com/nonparametric-statistical-significance-tests-in-python/