Student: Wei Wu (850023680)

Title: Machine Learning Final Project

Instructor: Dr.Rokach

Date: Aug 1st 2021

# Stochastic Weight Averaging in Low Precision

**Introduction**

In this project, we are going to evaluate an algorithm for training deep neural networks on classification tasks. The algorithm is called SWALP, referring to stochastic weight averaging in low-precision training (Yang et.al, 2019). In addition to the algorithm itself, we would perform a possible improvement on the original algorithm and select another well known one for comparison.

**SWALP**

The stochastic weight averaging algorithm is a proposed approach using the stochastic weight averaging while quantizing all numbers including the gradient accumulator and the velocity vectors during training.

There are two methods in SWALP, the original SWA algorithm and the low-precision method. The stochastic weight averaging algorithm was proposed for improved generalization in deep learning. It takes an average of SGD (stochastic gradient descent) iterates with a modified learning rate schedule has been shown to lead to a wider optima, which also connect with the width of the optimum and generalization performance.

The low-precision method was to shorten the time of training deep neural networks. Because deep Neural Networks usually requires a long time for training. One desirable techniques to fit numbers that are frequently used during the training process into the on-chip memory is *low-precision computation.* Because using fewer bits to represent numbers would reduce both memory consumption and computational cost.

In the paper, they have provided two versions of SWALP algorithm, which is shown in the figure below.

SWALP took advantages from the original SWA algorithm as well as the low-precision method and made the following contributions according to the paper. SWALP outperforms low-precision SGD and is even competitive with full-precision SGD down to 8 bits. We could also find a flat region of the loss surface even more robust to quantization, since in this region large perturbations of the weights do not affect the quality of the solution.

**Dataset**

It was a little difficult to decide which task to be selected. Therefore, at the very beginning the datasets selected were for regression tasks, but later on we finally decided on classification tasks. For this project, we have selected 20 datasets from the classification datasets. Detailed list of the datasets will be presented below. The objective of these datasets is to classify the data points into different classes. Some of them are multiple class problems and some of them are binary class ones. The features in all of the datasets are not labeled with specific names, for examples, the features in abalon set were f1 through f8. The target variable “class” or “label” is the last column of each dataset.

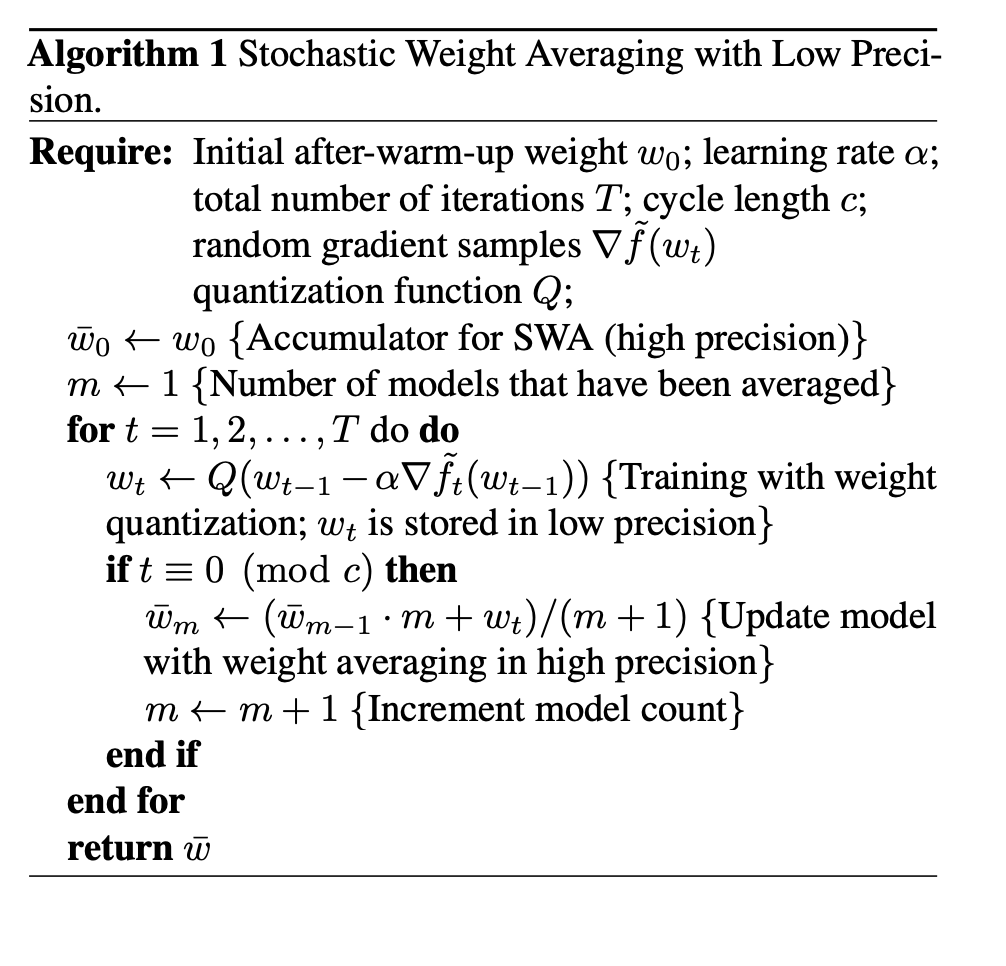


Figure 1: <https://arxiv.org/pdf/1904.11943.pdf>

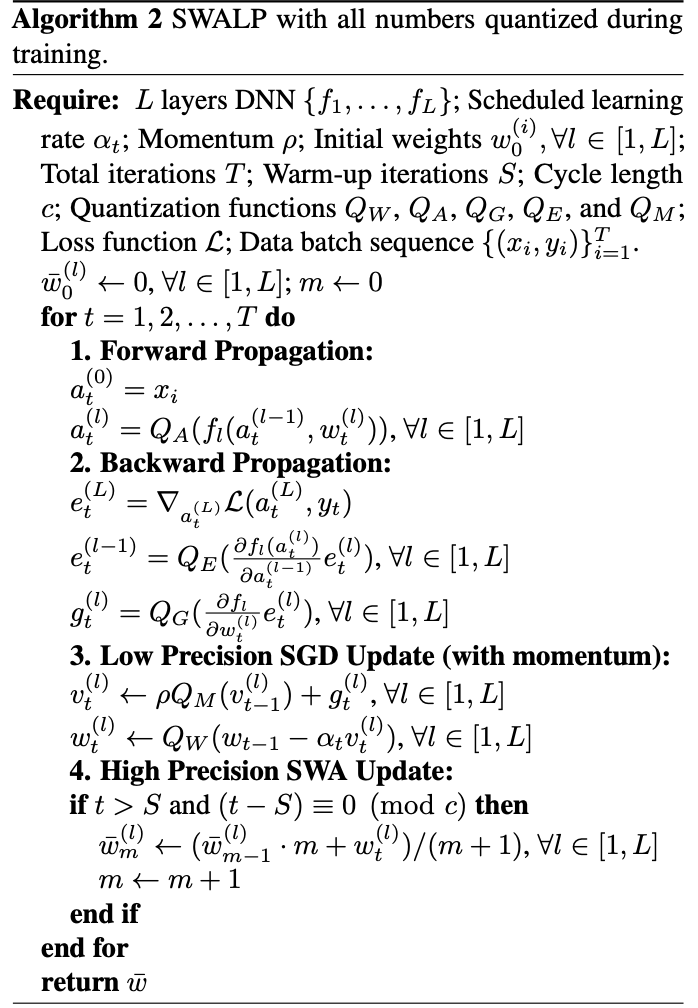


Figure 2: <https://arxiv.org/pdf/1904.11943.pdf>

A list of the datasets chosen are presented below:

1. Abalon.csv
2. acute-inflammation.csv
3. Acute-nephritis.csv
4. Annealing.csv
5. ar4.csv
6. Arrhythmia.csv
7. Audiology-std.csv
8. iris.csv
9. balance-scale.csv
10. Bank.csv
11. baseball.csv
12. Blood.csv
13. breast\_cancer.csv
14. car.csv
15. Cardiotoguraphy-3clases.csv
16. Congressional-voting.csv
17. Contract.csv
18. credit\_approval.csv
19. Ecoli.csv
20. Fertility.csv

**Methods**

The purpose of the project is to implement SWALP algorithm during training to help improve the performance of a neural network model. Then we will modify the SWALP algorithm to try to make it better. Last but not least, we will use SGD ( stochastic gradient descent) algorithm as a comparison with SWALP. The packages used in the project are sklearn package and torch.

**Data Preparation**

First of all, we loaded the datasets into data frames and selected all the features variables to be X and the target variable to be y. Then we took **80 percent** of the dataset to be the training set and **20 percent** to be the test set. Next, we split the training set into 10 folds and selected one of the folds to be the validation set during training.

The split of the training set and the test set performed only once before the process of training, whereas the the separation of the validation set happened inside the cross validation so that different models would get various set of the the training data.

Finally, both the features and the targets variables have been converted to tensor datasets and stored with pytorch DataLoader.

Once the data loaders are ready, we could then build the neural network to classify the data.

**Base Model**

The base model build in this project is quite simple. The Sequence Model is a deep multilayer neural network, including an input layer, a hidden layer and an output layer. The activation between each layer is the “relu” function. Since the datasets are all numerical, we did not put a dense layer before the output the layer and we did not put any activation functions on the output layer either.

Now we would began our process of training the model.

**Stage One: SWALP**

In order to use low precision numbers during training, we define a quantization function Q that rounds a real number to be stored in fewer bits as the author suggested in the paper. In this project, we use block fixed point quantization with stochastic rounding to demonstrate the algorithm as same as used in the paper. As suggested, BFP is preferred over fixed-point because BFP usually has less quantization error caused by overflow and underflow when quantizing DNN models.The quantization methods are implemented in models.py.

For block floating point numbers, all numbers within a block share the same exponent, which is allowed to vary like a floating number, where it has individual exponent.

Suppose we allocate W bits for each number in the block and F bits for the share component. The shared exponent E(w) for a block of numbers to avoid overflow. The formula to compute the shared exponent is :

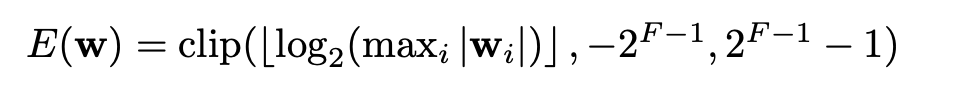


Figure 3:<https://arxiv.org/pdf/1904.11943.pdf>

The presudo code for block floating point quantization method is demonstrated in algorithm 1 above.

Moreover, as discussed in the paper, in order to train DNNs with low-precision storage, we would need to also quantize other numbers during trading. The ways to quantize the weights, activations , back propagation errors and gradient signals are in the same convention. Since SGD (stochastic gradient descent) takes momentum as a parameters, in order to use momentum during training, we need to store the velocity tensors in low-precision, then we modified the SGD update as follows:

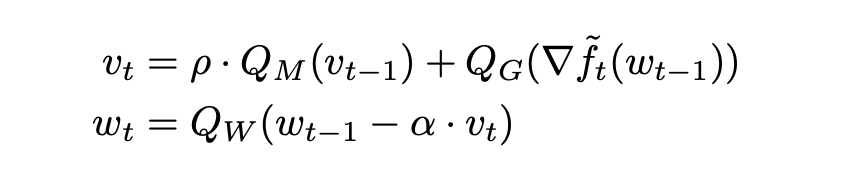


Figure 4: https://arxiv.org/pdf/1904.11943.pdf

Qm, Qg, and Qw are quantizers for momentum, gradients and weights. For simplicity, the quantizer for momentum and gradients are both set to 8 bits. Detailed steps of this process is presented in Algorithm 2 above.

**Stage Two: Improvement of SWALP**

In the paper, the authors started with a regular low-precision SGD as a pre-trained model. SWALP then continues to run low-precision SGD while averaging the current model weight into an accumulator. Since SGD, or the stochastic gradient descent algorithm is one of the optimizing methods towards the optimal solutions, one possible way to improve the performance of SWALP might be to use another optimize algorithm.

The stochastic gradient descent algorithm is one of the traditional method for optimizing. Although it had been proved that it performed well in a lot of models and data bases, it would also take a longer time to reach to the optimal point. Therefore, in order to speed up, we could combine the original SGD with the parameter momentum, which is the method used in the paper and in this project. The role of momentum is to reduce the impact of fluctuation when converging., thus it would converge faster.

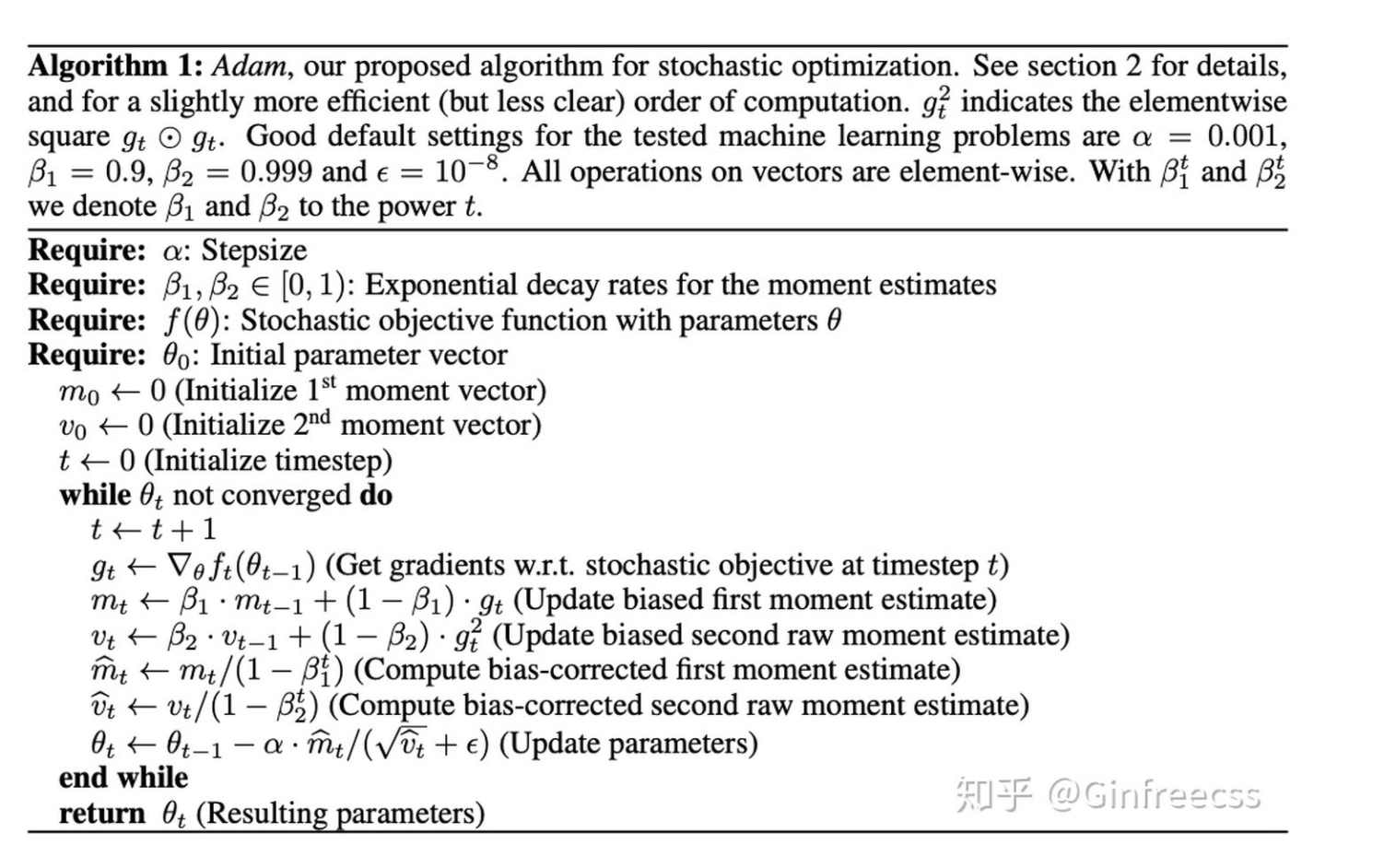


Figure 5: <https://arxiv.org/pdf/1412.6980.pdf>

However, because of a new addition to the hyper parameters, the initialization of the values would influence the convergence of the model.

Then after researching for various optimizing algorithms, a commonly used algorithm called Adaptive moment estimation (Adam) might be a good choice.

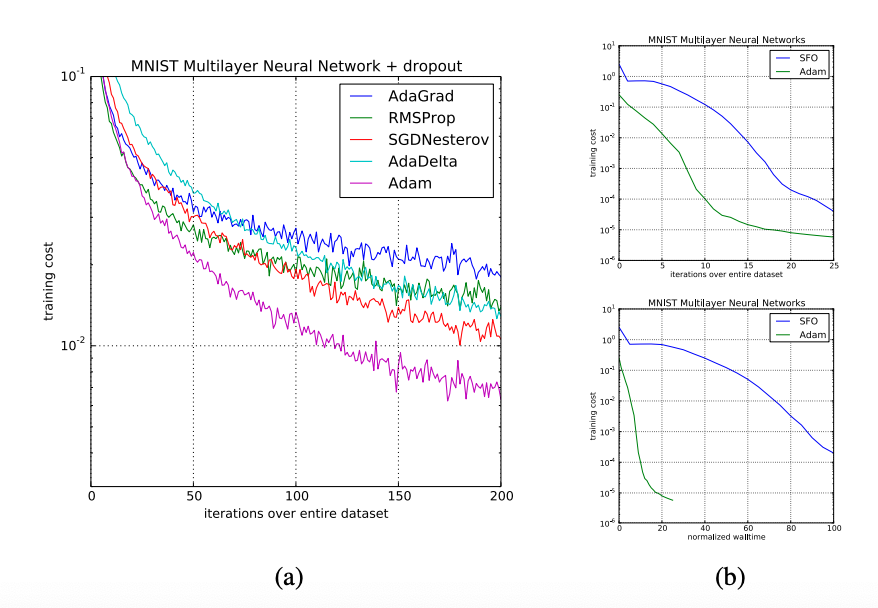


Figure 6: <https://arxiv.org/pdf/1412.6980.pdf>

"Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments (P.KingMa, Lei Ba. 2015) .” It was experimented on Deep Multilayer Neural Network and Convolutional Neural Network and showed better convergence than other methods.

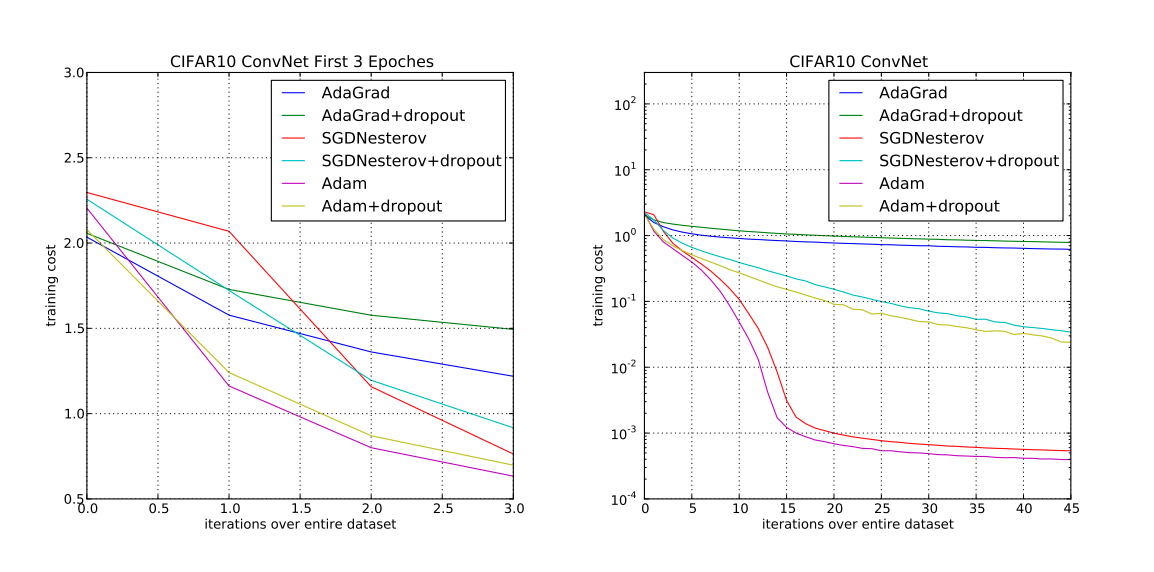


Figure 7: https://arxiv.org/pdf/1412.6980.pdf

Hence, as shown in the figure on the right, since Adam had a better performance than SGD on the Multilayer Neural Network, we suppose it might improve the SWALP’s performance if we use Adam as a pretrained model instead of SGD.

**Stage Three: Comparison with SGD**

We select SGD to be a baseline to compare with SWALP and SWALP with Adam. Below is the detailed information about the SGD algorithm.

In the project, both SGD and Adam are provided by **torch.optim** package. We did not specify a stage for the evaluation of SGD. Instead, we compared SGDwith SWALP at Stage One and with SWALP at stage two. The algorithm is measured by the accuracy of predictions on the dataset. The performance results are stored as SGD error in sheet 3 and sheet 4.



Figure 8: https://zhuanlan.zhihu.com/p/91736992

**Evaluations**

We use a external10-fold cross validation as an evaluation protocol to separate between Training and Test sets and an internal 3-fold cross validation for hyper parameters optimization.

The separation between the datasets was mentioned above in the data preparation part. The implementation of the k-fold split is in the **get\_k\_fold** function.

For the hyper parameters optimization, we use Random Search to select the best parameters for the algorithm. All the values of each parameter were put in a dictionary. During the process, we randomly select a value from each and pass to the algorithm. The measurement metrics of the performance was the accuracy of the training set. The hyper parameters for the algorithms selected were **momentum** and **batch\_size**. The dimension of the hidden layer was also selected because it was difficult to determine.

In addition to the cross validation procedure, more evaluating measurements of the algorithms are listed as the following:

1. Accuracy, under the assumption the classification is the Class with the highest probability

2. TPR, truth positive rate

3. FPR, false positive ratePrecision

4. AUC, area under the ROC curve

5.Area under the Precision Recall Curve

6. Training Time

7. Inference Time

Detailed results of the algorithms are reported in the excel file.

**Result**

In this section, we present the partial results achieved with the algorithms. Due to the fact that there are a lot evaluation metrics, we will not demonstrate the entire results in the report.

Detailed performance will be presented in an Excel file in the repository.

| Dataset Name | Algorithm Name | Cross\_Validation | Accuracy | Algorithm Name | Cross Validation | Accuracy |
| --- | --- | --- | --- | --- | --- | --- |
| Abalon | SWALP | 79.8561 | Train\_accuracy: 73.4412 Test\_accuracy: 64.8681 | SWALP-with Adam | 76.9784 | Train Accuracy: 72.0624 Test\_accuracy: 63.7890 |
| acute inflammation | SWALP | 100.0 | Train\_accuracy: 100.0 Test\_accuracy: 95.8333 | SWALP-with Adam | 100.0 | Train\_accuracy: 100.0 Test\_accuracy: 95.8333 |
| Acute nephritis | SWALP | 100.0 | Train\_accuracy: 100.0 Test\_accuracy: 95.8333 | SWALP-with Adam | 100.0 | Train\_accuracy: 100.0 Test\_accuracy: 95.8333 |
| Annealing | SWALP | 86.4406 | Train\_accuracy: 55.3672 Test\_accuracy: 51.9774 | SWALP-with Adam | 99.1525 | Train Accuracy: 92.9379 Test\_accuracy: 79.0960 |
| Ar4 | SWALP | 85.7142 | Train\_accuracy: 80.9524 Test\_accuracy: 76.1905 | SWALP-with Adam | 85.7412 | Train Accuracy: 80.9524 Test\_accuracy: 80.9524 |
| Arrhythmia | SWALP | 100.0 | Train\_accuracy: 100.0 Test\_accuracy: 46.6667 | SWALP-with Adam | 100.0 | Train Accuracy: 78.8889 Test\_accuracy: 36.6667 |
| Audiology-std | SWALP | 100.0 | Train\_accuracy: 100.0 Test\_accuracy: 58.9744 | SWALP-with Adam | 100.0 | Train Accuracy: 100.0 Test\_accuracy: 51.2821 |
| Iris | SWALP | 100.0 | Train\_accuracy: 100.0 Test\_accuracy: 80.0 | SWALP-with Adam | 100.0 | Train Accuracy: 100.0 Test\_accuracy: 76.6667 |
| Balance-scale | SWALP | 100.0 | Train\_accuracy: 100.0 Test\_accuracy: 93.4959 | SWALP-with Adam | 100.0 | Train Accuracy: 100.0 Test\_accuracy: 91.8699 |
| Bank | SWALP | 100.0 | Train\_accuracy: 94.9446 Test\_accuracy: 84.4961 | SWALP-with Adam | 99.5016 | Train Accuracy: 94.4075 Test\_accuracy: 83.9424 |
| Ozone | SWALP | 100.0 | Train\_accuracy: 100.0 Test\_accuracy: 97.8304 | SWALP-with Adam | 100.0 | Train Accuracy: 98.7179 Test\_accuracy: 97.2387 |
| Blood | SWALP | 79.5918 | Train\_accuracy: 77.8912 Test\_accuracy: 68.0272 | SWALP-with Adam | 90.8163 | Train Accuracy: 81.9728 Test\_accuracy: 71.4286 |
| breast\_cancer | SWALP | 100.0 | Train\_accuracy: 100.0 Test\_accuracy: 68.4211 | SWALP-with Adam | 100.0 | Train Accuracy: 100.0 Test\_accuracy: 70.1754 |
| Car | SWALP | 99.8263 | Train\_accuracy = 99.9421. Test\_accuracy = 96.8750 | SWALP-with Adam | 100.0 | Train Accuracy: 96.667 Test\_accuracy: 90.7246 |
| Cardiotocography-3clases | SWALP | 100.0 | Train\_accuracy = 99.7636. Test\_accuracy = 87.4704 | SWALP-with Adam | 99.6453 | Train Accuracy: 89.0071 Test\_accuracy: 81.3239 |
| Congressional-voting | SWALP | 70.6896 | Train\_accuracy = 67.2414. Test\_accuracy = 49.4253 | SWALP-with Adam | 70.6896 | Train Accuracy: 67.2414 Test\_accuracy: 54.0230 |
| Contract | SWALP | 98.4693 | Train\_accuracy = 89.9660. Test\_accuracy = 45.5782 | SWALP-with Adam | 94.3877 | Train Accuracy: 79.2517 Test\_accuracy: 48.6395 |
| credit\_approval | SWALP | 100.0 | Train\_accuracy = 99.6377. Test\_accuracy = 76.8116 | SWALP-with Adam | 100.0 | Train Accuracy: 100.0 Test\_accuracy: 75.3623 |
| Ecoli | SWALP | 100.0 | Train\_accuracy = 99.2424. Test\_accuracy = 78.7879 | SWALP-with Adam | 100.0 | Train Accuracy: 100.0 Test\_accuracy: 80.3030 |
| Fertility | SWALP | 100.0 | Train\_accuracy = 100.0. Test\_accuracy = 83.3333 | SWALP-with Adam | 100.0 | Train Accuracy: 100.0 Test\_accuracy: 55.5556 |

The table above demonstrated the cross validation score and the accuracy scores for all the datasets with two different algorithms. From the table above, we could find that the cross validation score increased a little with the Adam optimizer, despite a few scores dropped a little. However, for the blood dataset, the score had increased from 79 to 90, while for the contract dataset, the score decreased from 98 to 94. It seems that the Adam optimizer did not affect much with the accuracies performance.

Then when we took a look at the training and test accuracies for both algorithms, we may conclude the fact the change of SGD to Adam did not make any significance difference on the predicted accuracies. One interesting thing is that the SWALP algorithm with Adam increased the test accuracy but decreased the train accuracy, for example, in the contract dataset, the test accuracy has changed from 45 to 48, while the train accuracy decreased from 89 to 79. Overall, we did not find any remarkable changes on the accuracies between these two algorithms, although we did notice some drops for the second stage.

| Dataset Name | Training-Loss | Training-Loss-2 | Validation-Loss | Validation-Loss-2 | SGD-Error | SGD-Error-2 | SWA-Error | SWA-Error-2 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Abalon | 0.6069 | 0.5991 | 0.7872 | 0.8661 | 40.9638 | 39.0280 | 42.1686 | 39.2086 |
| acute inflammation | 0.0045 | 0.0000 | 0.3390 | 0.0634 | 0.0 | 0.0 | 0.0 | 0.0 |
| Acute nephritis | 0.0048 | 0.0000 | 0.1006 | 1.5828 | 0.0 | 0.0 | 0.0 | 0.0 |
| Annealing | 0.0000 | 0.2025 | 19.0896 | 11.7899 | 17.6470 | 20.3389 | 17.6470 | 23.7288 |
| Ar4 | 0.4981 | 0.4605 | 0.5418 | 0.5198 | 0.0 | 14.2857 | 0.0 | 14.2857 |
| Arrhythmia | 0.0166 | 0.6213 | 2.6651 | 21.8564 | 11.1111 | 66.6667 | 11.1111 | 66.6667 |
| Audiology-std | 0.0705 | 0.0000 | 2.3391 | 40.0482 | 33.3333 | 38.4615 | 33.3333 | 38.4615 |
| Iris | 0.0693 | 0.0001 | 0.1940 | 6.1306 | 0.0 | 40.0 | 0.0 | 40.0 |
| Balance-scale | 0.6929 | 0.0002 | 0.1928 | 0.5311 | 8.3333 | 7.3170 | 8.3333 | 7.3170 |
| Bank | 0.0325 | 0.1466 | 0.7501 | 1.0272 | 11.1111 | 13.2890 | 11.1111 | 13.2890 |
| Ozone | 0.0103 | 0.0329 | 0.1212 | 0.1868 | 6.0 | 7.1005 | 6.0 | 7.6923 |
| Blood | 0.4604 | 0.3885 | 0.5875 | 1.3441 | 42.8571 | 40.8163 | 42.8571 | 36.7346 |
| breast\_cancer | 0.0941 | 0.0001 | 1.3192 | 3.8803 | 40 | 36.8421 | 40 | 31.5789 |
| Car | 0.0367 | 0.1168 | 0.1664 | 0.5380 | 5.8823 | 6.6956 | 5.8823 | 7.8260 |
| Cardiotocography-3clases | 0.0379 | 0.2152 | 0.4568 | 1.0152 | 11.9047 | 16.3120 | 11.9047 | 16.3120 |
| Congressional-voting | 0.5491 | 0.5260 | 1.0058 | 7.3040 | 50 | 41.3792 | 50 | 41.3792 |
| Contract | 0.5555 | 0.4008 | 1.4648 | 4.7112 | 58.6206 | 53.0612 | 51.7241 | 57.1528 |
| credit\_approval | 0.0373 | 0.0001 | 1.1418 | 4.3429 | 0.0 | 28.2608 | 0.0 | 28.2608 |
| Ecoli | 0.1363 | 0.0004 | 0.5968 | 3.6293 | 33.3333 | 22.7272 | 33.3333 | 22.7272 |
| Fertility | 0.0121 | 0.0000 | 0.7579 | 27.1134 | 0.0 | 50.0 | 0.0 | 50.0 |

The table above indicated the loss value averaged from 5 runs for the training set and the validation set as well as the error for SGD and SWAs. The SWA error for stage one was measured for SWALP algorithm, and the SWA error for stage two was measured for SWALP with Adam.

First of all, we noticed that the most training loss for the second stage decreased at stage 2, but the test loss increased. However, surprisingly we did not find a remarkable difference from SGD to the other models. Because from the chart above, the SGD error and the SWA error stayed the same for most of the datasets. In the meantime, some of the SWA error was less than SGD and some of SWA error was greater than SGD error. Therefore, we could not tell whether the modification made any difference to the two algorithms merely from the chart above.

For the purpose of measure the difference between the two algorithms, we did a Friedman Test.

**Friedman Test**

After we trained the models on two different algorithms, the reported performance of them was shown in the results. However, because of the amount of evaluation metrics, we might not have a good sense of whether their difference are statically significant.

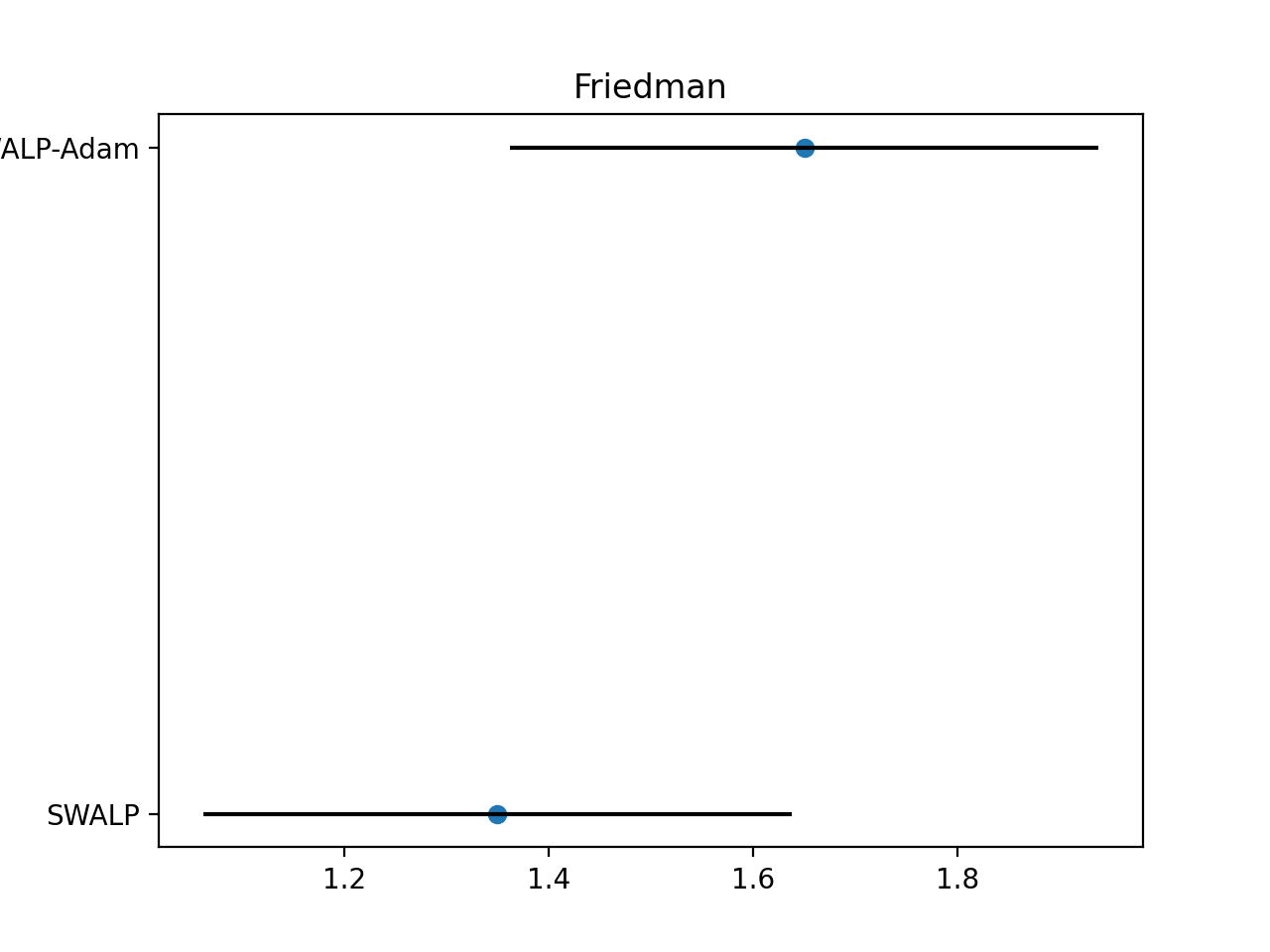
Thus, we use Friedman Test to measure the difference of SWALP and SWALP with Adam.The chosen metrics for this test was “Truth Positive Rate (TPR)”. Because the truth positive rate is the rate of how many datapoint we predicted was the same as the target. It should be a good choice to be metrics of the performance for each algorithm.The Friedman Test function takes in the number of algorithms to be measures, the number of datasets used, and a ranked matrix as parameters. The ranked matrix is to rearrange the scores in the matrix in descending order .

We took the TPR values from each stage as numpy arrays. The ranked matrix function took the transpose matrix to be the input matrix.

We also computed the CD score from the Nemenyi test for the purpose of plotting the test result.

Then we define our hypothesis for the test that:

H\_0: These algorithms are not statistically different.



Friedman Test

H\_1: These algorithms are statistically different.



Figure 9: <https://wenku.baidu.com/view/98e46485a7c30c22590102020740be1e640ecc2f.html>

Degree of Freedom is set to be 2, and alpha is set to be 0.05, so the critical value should be 5.991 as in the critical value chart.

The output Friedman score of the TPR values is : 1.8791208791208758, which is less than the critical value. Therefore, we have strong evidence to reject the null hypothesis. Then these two algorithms are statistically different.

Below is a plot of the Friedman test. We could notice that nearly half of the lines overlapped but not the whole. This would also be an evidence that there is a difference between these two algorithms.

In order to further test the difference between the algorithms, we need a Post Hoc test.

**Post-Hoc Test**

The post-hoc test is a method to further test the statistical difference between the algorithms. We used the stats package from scipy to compute the f value and the p value., where f -value is to measure the difference between the two TPR lists from two algorithms and p-value is the output significance level. The significance level for this test is set to be 0.05

The hypothesis for this test are:

H\_0: There is no difference between the datasets.

H\_1: There is a difference between the datasets.

The results of the post hoc test is as the following:

f\_value: 0.0035683340235741984

p\_value: 0.9526793249400098

Thus, since p-value is less than the significance level, we do not have enough evidence to reject the null hypothesis.

**Discussion and Conclusion**

In the article, the authors mentioned SWA (Stochastic Weight Averaging) algorithm as a correlated work. We did some experiments of the SWA algorithm for the datasets as well. The results turns out to be similar what the authors suggested in the article that the SWA algorithm and the SGD algorithm act alike during training.

Overall, when we compared the results of SGD error and SWA error for both stages, we noticed that the SWALP and SGD algorithms made some difference at stage two, because most of the data were the same for both at stage one. In addition to that, the SWA error was less than the SGD error at stage two as well. Thus, we could actually see a change with two different methods. Generally speaking, the SWALP with Adam performed better than the SGD models. In terms of the SWALP and the SWALP with Adam, it became a little bit tricky due to the results from the Friedman Test and the Post-Hoc Test.

The Friedman Test shown that the SWALP algorithm and the SWALP with Adam algorithm were statistically different, however, the results of the post-hoc test indicated that we do not have enough evidence to prove the difference. The possible reason behind that might because we did not change much of the SWALP algorithm therefore the improvement of the algorithms was not evident. However, the accuracies improved for some of the datasets for SWALP with Adam and the loss decreased as well. Hence, we might conclude that there was a slight improvement for the SWALP with Adam instead of SGD.

**References**

Kingma Diederik P., Ba Jimmy Lei, “Adam: A Method for Stochastic Optimization”, ICLR, https://arxiv.org/pdf/1412.6980.pdf. 2015.

Yang et.al, “SWALP: Stochastic Weight Averaging in Low Precision Training”, Cornell University, https://arxiv.org/pdf/1904.11943.pdf. 2019.