Averaging Weights Leads to Wider Optima and Better Generalization

Indian Institute of Information Technology Vadodara

Course Coordinator

Dr. Jignesh Bhatt

Teaching Assistant

Ms. Swati Rai

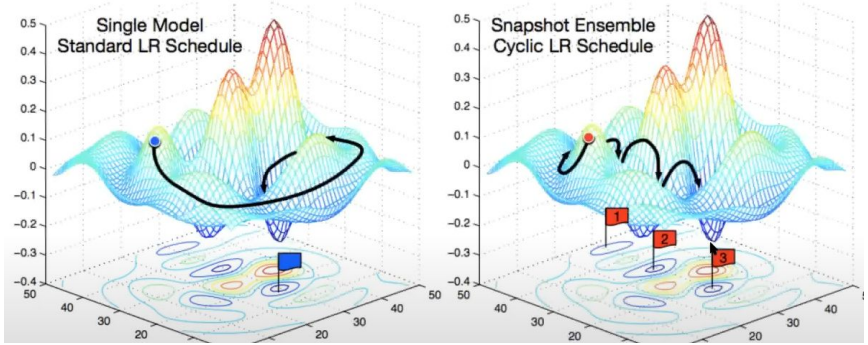
Team Members:

Aman Kumar Raj (201951019) **Abhiyank Raj Tiwari (201951011)** **Sanskar Srivastava (201951136)**

Paper implemented: https://arxiv.org/abs/1803.05407

Overview

- Deep neural networks are typically trained by optimizing a loss function with an SGD variant, in conjunction with a decaying learning rate, until convergence.
- We show 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.
- We also show that this Stochastic Weight Averaging (SWA) procedure finds much flatter solutions than SGD.

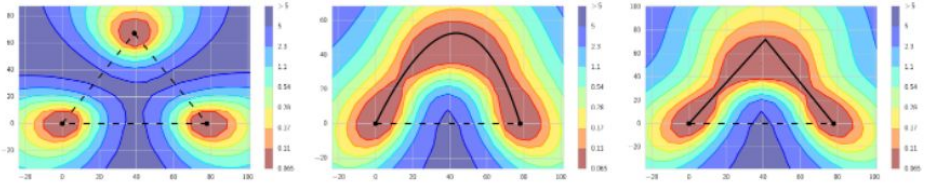


Introduction

- With a better understanding of the loss surfaces for multilayer networks, we can accelerate the convergence, stability, and accuracy of training procedures in deep learning.
- We show that SGD with cyclical and constant learning rates traverses regions of weight space corresponding to high-performing networks.
- We find that while these models are moving around this optimal set they never reach its central points. We show that we can move into this more desirable space of points by averaging the weights proposed over SGD iterations.

Continued…

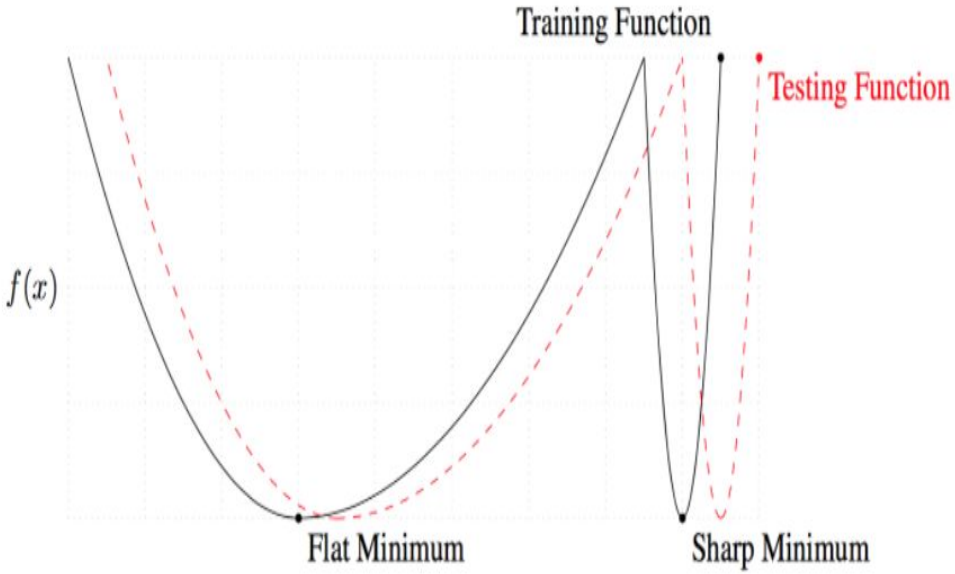
- We demonstrate that SWA leads to solutions that are wider than the optima found by SGD.
- SWA achieves notable improvement for training a broad range of architectures over several consequential benchmarks.
- SWA is extremely easy to implement and has virtually no computational overhead compared to the conventional training schemes.



LEFT: Traditional intuition is that good local minima are separated by regions of high loss. This is true if we travel along the lines connecting local minima. MIDDLE and RIGHT: However, there exist paths between local minima, such that loss stays low on these paths. FGE takes snapshots along these paths and creates an ensemble out of the snapshots. Source.

Stochastic Weight Averaging

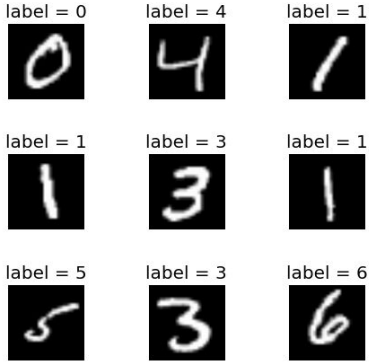
- Stochastic Weight Averaging is an optimization procedure that averages multiple points along the trajectory of SGD, with a cyclical or constant learning rate.
- The local minima at the end of each learning rate cycle tend to accumulate at the border of areas on loss surface where loss value is low . By taking the average of several such points, it is possible to achieve a wide, generalizable solution with even lower loss.



Narrow and wide optima. Flat minimum will produce similar loss during training and testing. Narrow loss, however, will give very different results during training and testing. In other words, wide minimum is more generalizable than narrow. Source.

Dataset:

- MNIST ("Modified National Institute of Standards and Technology") is the default “hello world” dataset of computer vision. Digit Recognizer | Kaggle
- Our goal is to correctly identify digits from a dataset of tens of thousands of handwritten images.
- Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total
- The training data set, (train.csv), has 785 columns. The first column, called "label", is the digit that was drawn by the user.
- The test data set, (test.csv), is the same as the training set, except that it does not contain the "label" column.
- For each of the 28000 images in the test set, output a single line containing the ImageId and the digit you predict.



Model Architecture

We used CNN to make our model. First, we used different Conv2D layers with batch normalization in between them.

We used batch normalization for stabilizing the learning process which dramatically reduced the number of training epochs required to train deep networks.

We used the relu activation function in layers to help the network learn complex patterns in the data.

The output layer of model is a dense layer of 10 neurons(i.e. Number of classes to predict) with softmax activation function to predict probability of each class.

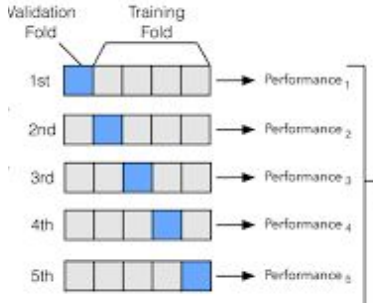
```
model = p1_model2()
model.eval()
```

```
p1_model2(
  (cnn_layers): Sequential(
    (0): Conv2d(1, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(4, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU(inplace=True)
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (linear_layers): Sequential(
    (0): Linear(in_features=196, out_features=10, bias=True)
  )
)
```

Evaluation Metrics and Validation Strategy

- We have used Accuracy as evaluation metric because our classes {0,1,2,3,4,5,6,7,8,9} are balanced in our dataset.

- We have used 5 fold Cross-Validation Strategy to measure performance of our model.
- In 5 Fold Cross-Validation, We split our dataset into 5 folds/parts. Then train our model 5 times. Each time we hold one fold and train our model on rest 4 folds. After training we make predictions for the hold our fold of our training set, This fold has not been seen by our model and thus is a validation data. Each time our validation set changes giving us more robust model.



Aim :-

Our aim is to train a model with using SWA and train another model without using SWA and compare their train score and test score.
First we optimize both models using Optuna.

Best hyperparameters for Model with SWA:

```
{'batch_size': 128, 'epochs': 50, 'learning_rate': 7.992390252758013, 'patience': 5,
'momentum': 0.8380950070538395}
```

Best hyperparameters for Model without SWA:

```
{'batch_size': 512, 'epochs': 50, 'learning_rate': 2.714460153443568, 'patience': 3,
'momentum': 0.34345263068415116}
```

Next we train both models and measure their performance.

Exp1:- (Without SWA)

Train score: 0.99514 Test score: 0.957711

exp7_mnist

Python · kaggle-api-creds, Digit Recognizer

Notebook | Data | Logs | Comments (0) | Settings

Competition Notebook	Run	Public Score	Best Score	
Digit Recognizer	3612.3s - GPU	0.95771	0.95771 V3	Version 3 of 3

Exp2:- (With SWA)

Train score: 0.98803 Test score: 0.97739

exp6_mnist

Python · kaggle-api-creds, Digit Recognizer

Notebook Data Logs Comments (0) Settings

Competition Notebook	Run	Public Score	Best Score	
Digit Recognizer	6232.6s - GPU	0.97739	0.97739 V2	Version 2 of 2

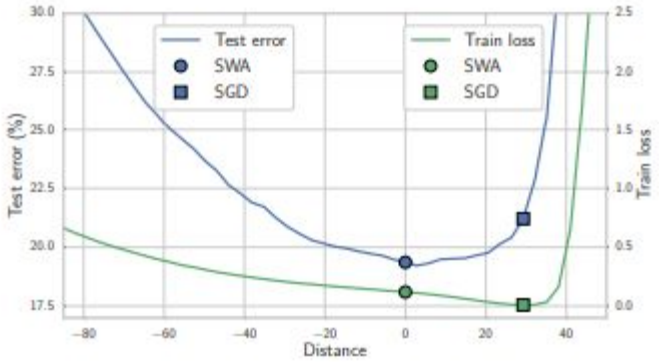
Conclusion:

Model without SWA: Train score: 0.99514 Test score: 0.95771

Model with SWA: Train score: 0.98803 Test score: 0.97739

Model without SWA performed better on Train set but worse on Test set.
Model with SWA performed average on Train set but quite good on Test set.

Hence, SWA is helping generalize our model.



Link of the Inference Kernels:-
https://www.kaggle.com/raj401/exp6-mnist (With SWA)
https://www.kaggle.com/raj401/exp7-mnist (Without SWA)