Optimization for Machine Learning Project: Combining Optimization Methods

Cyrille Kone, Jiaan Zhu, Michael Chan
EPFL

***Abstract*—The choice of optimizer is tricky for deep learning problems. Combing optimizers may improve the performance of training. In this project, we use two methods to combine SGD and ADAM, and compare their performance with single SGD or ADAM. The results show that mixed optimizer take advantage of both SGD and ADAM, leading to better performance.**

I. Introduction

Optimizers are algorithms or methods to update parameters to reduce the loss, they play important roles in the deep learning. In most papers, only one optimizer is used, which is proved to be better than other optimizers for the particular problem they deal with in their papers. The aim of this project is to investigate the effect of combining optimizers in deep learning. We try two methods to mix optimizers: (1)ADAM on hidden layers and SGD on output layers; (2)combine the SGD and ADAM in the parameter updating step, which named MAS(Mixing ADAM and SGD). In this report, related work is shown Section II and details of our methods are described in Section III and IV. Then Section V concludes this report.

II. Related work

Some researchers already used mixed optimizers in their papers. Landro et.al. develop a new optimizer named MAS(mixing ADAM and SGD) that linearly integrate the SGD and ADAM[1]. This new optimizer shows better performance than the single ADAM or SGD optimizers. Keskar & Socher propose SWATS(Switches from Adam to SGD) in which the switch will be triggered in a certain condition[2]. Their results show the generalization gap between SGD and ADAM is closed in this way.

III. Methods

A. Adam on hidden layers and SGD on output layers

The idea behind this method is that as we go from input to output layers, the parameters have an increasing impact on the final classification. Since Adam converges faster and SGD is known to generalize better, our aim is to help the network generalize better with SGD while still converging faster with Adam in the first and hidden layers.

B. Mixing Adam and SGD variants

The idea of mixing optimizers' updating steps by doing a linear combination on the gradients was shown to work in [1]. And the idea in this section is to reassess and try to build on the concept for variants of the updates. Note that the variants are for now empirically tested.

1) Standard Linear Combination: As shown in [1] the intuition behind summing (scaled) a gradient update of Adam and SGD is to have a smaller update step when the two gradients are far apart while we would have a strong update when the two gradients are close.

2) Direction Correction: The idea is to update weights of the layer using Adam when both gradients have a negative scalar product and update with SGD when the scalar product is positive.

3) Vector Combination: Vector combination would imply combining part of the Adam gradient with SGD gradient.

4) Adam Normalization: Since the update step is computed as well in Adam optimization, we use this computed update step to normalize the SGD gradient and use that as a new update step. The intention behind is to have rather fast convergence rate from Adam while keeping the better validation accuracy from SGD

C. Testing Procedure

Our goal is to test different combinations of optimizers on the same neural network. Furthermore, since training a neural network is a stochastic procedure, we run many trials with the same configuration. Our testing procedure includes two different architectures of neural network. The first is a simple dense neural network (fit to MNIST) and the second is a convolutional neural network (fit to CIFAR 10). All the trainings are done using PyTorch.

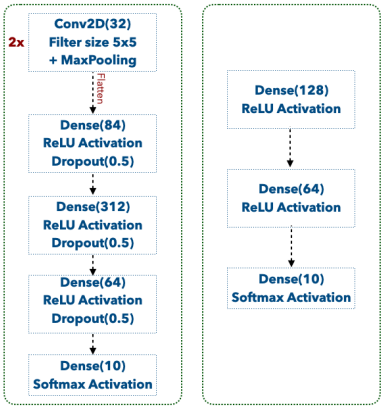


Fig. 1: Architecture of the feed forward neural network (right) where Dense(n) is a perceptron layer with n neurons. Conv2D (left) is the short for 2D convolutionnal layer.

Below is the benchmark procedure:

1) Choose the architecture and the dataset
2) Choose the hyperparameters according to your optimizer
3) Train for 150 epochs with a batch size of 256 for MLP and 25 epochs with a batch size of 20 for CNN
4) For each epoch record some metrics at regular frequency (from 1 to 20 points per epoch according to the computational power available)
5) Repeat the training for 10 trials
6) Report the average metrics and losses over the 10 trials

IV. Testing different combinations of optimizers

A. Adam on hidden layers and SGD on output layers

For this combination, we mix SGD and Adam in the following way:

- We use Adam for the parameters in the hidden layers
- SGD is used for the parameters in the output layer

This optimizer is implemented with the dense neural network. The first layer (see Fig. 1) is used as hidden layer (so trained with Adam) and the last two layers are trained with SGD. The training is done simultaneously. We report some results below.

1) Results: We analyse the mixed optimizer for different couples of learning rates. Below (Fig. 2, 3, 4).

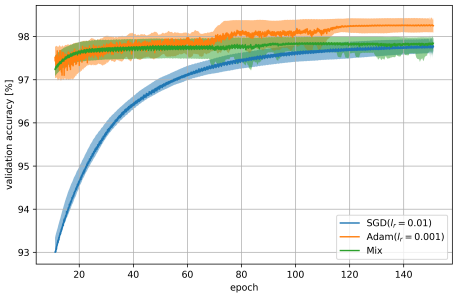


Fig. 2: Evolution of the mean over 10 trials of the validation accuracy for each optimizer. The optimizer designated as mix is obtained by combining SGD and Adam with their respective learning rates as described above. The area of uncertainty is estimated by taking for each trial the maximum and the minimum accuracy achieved.

Above, Adam is used with $l_r = 10^{-3}$ (its common default learning rate) and SGD with $l_r = 10^{-2}$ (also default learning rate) We can see that in this configuration, the mixed optimizer tend to be better than SGD and slightly below Adam. Another advantage of this mixed optimizer is the computation time. We have reported the mean(per trial per epoch) computation time of the three optimizers and we have found respectively (see benchmark procedure section for the setup): $t_{\text{Adam}} = 0.701s$, $t_{\text{SGD}} = 0.506s$ and $t_{\text{Mix}} = 0.541s$. While those values should depend on our hardware (NVIDIA V100- 16GB Kaggle) the relative comparison can be expected to hold device-independent. The mixed optimizer is thus 25-30% faster (in computation time) than Adam and slightly slower than SGD using the PyTorch implementation of both optimizers. To ensure our remarks were not only due to these particular learning rates, we have repeated the same procedure using different combinations of learning rates for Adam and SGD. In particular, we increase the learning rate of Adam. Our goal is to show that using this mixed optimizer, we can still take advantage of SGD to stabilize convergence. We can see in (Fig. 3) that the mixed optimizer is faster than SGD (due to Adam acceleration) but it still preserve its generalization capability.

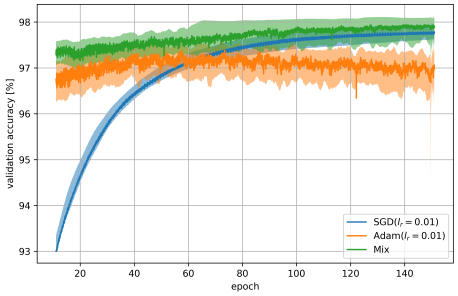
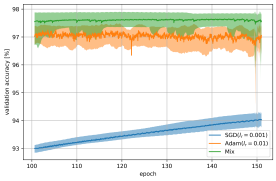
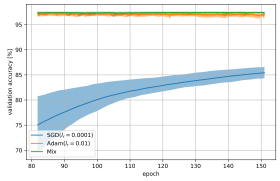


Fig. 3: Evolution of the mean over 10 trials of the validation accuracy for each optimizer. Adam is used with high learning rate(0.01). The figure is foreshortened to plot the curve from validation accuracy higher than 93%

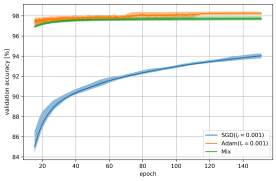
Below we test a couple of learning rates ((Fig. 4). We can still see, that even with very high learning rate for Adam, the mixed optimizer is somehow competitive with lower computation time than Adam.



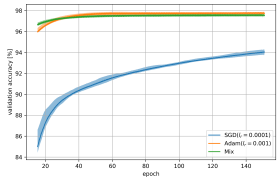
(a) SGD(10^{-3}) ; Adam(10^{-2})



(b) SGD(10^{-4}) ; Adam(10^{-2})



(c) SGD(10^{-3}) ; Adam(10^{-3})



(d) SGD(10^{-4}) ; Adam(10^{-3})

Fig. 4: Evolution of the mean over 10 trials of the validation accuracy for each optimizer. The figure is foreshortened to plot curve from validation accuracy higher than 93%

From those different combinations, we can remark that :

- The area of uncertainty of SGD is thinner and smoother, which means SGD is more robust to initialization

- In terms of convergence speed (measured as the derivative of the validation accuracy), Adam is faster than the mixed optimizer which is faster than SGD
- The mixed optimizer tend to take advantage of both optimizers (at least for our dataset)

However, this way of mixing is lacking some theoretical background (we didn't find it used in the literature) and require to designate some layers has hidden or output layers (recall that what we call output layer is not only the last layer). The choice was easy to make due to our small neural network.

B. Mixing Adam And SGD Variants

As mentioned in previously, we reproduce the MAS versus standard Adam or SGD on CIFAR10 and as well test different variants of the MAS method and compare them once again over CIFAR10 using a same CNN architecture.

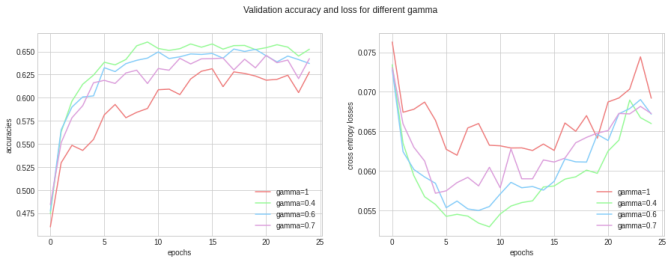


Fig. 5: Gamma parameter choice for combination. hyperparameters $\eta = 10^{-3}$ — $\gamma = 0.4, 0.6, 0.7, 1$

In order to apply a MAS, we require to find a outer-scalar Figure 5 with which we multiply the combined gradient. In this case we decided to go with $\gamma = 0.6$.

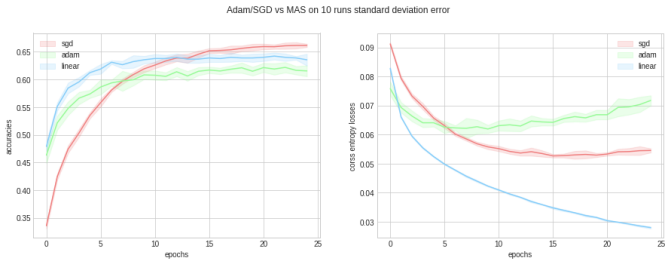


Fig. 6: Evolution of mean over 10 runs of validation accuracy and loss for each optimizer. standard comparison $\eta = 10^{-3}$ — $\gamma = 0.6$

We do find the same qualitative result as in [1]. Indeed in the validation accuracy we obtain a convergence rate similar to Adam while performing better in a fashion close to SGD.

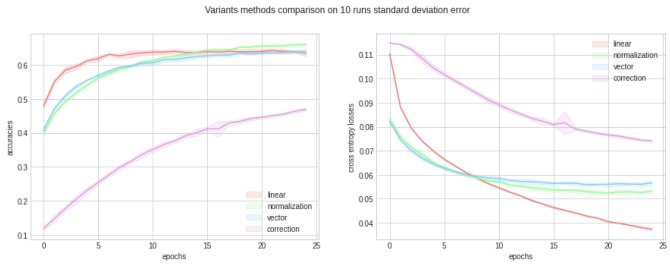


Fig. 7: Evolution of mean over 10 runs of validation accuracy and loss for each optimizer, variants comparison $\eta = 10^{-3}$ — $\gamma = 0.6$

Multiple points are to note: The way in which we create the variants are such that the fast convergence of Adam is kept while wanting to perform with higher validation accuracy. A candidate for such a task would be Normalization, which would mean to scale SGD gradient at each layer with the corresponding norm of Adam gradient. Correction is an attempt to keep a SGD gradient only if the inner product of Adam's and SGD's gradient is positive and if not replace it by Adam gradient and unfortunately fails to improves on validation accuracy. In the same way the vector composition (half Adam/SGD) does not seem to improve validation accuracy substantially, but has a similar convergence speed as Normalization method.

V. Discussions

In this project, multiple methods were tested to mix optimizers but in general the optimizers involved were only SGD and ADAM. In first tests, similarly to what was done in the Section IV, we tried including Frank-Wolfe in alternating layers' weight update and gradient combinations as well but unfortunately results were not conclusive (i.e low accuracy and increasing loss along epochs). As a result, we proved that the two kinds of mixed optimizers in our report take advantages of both SGD and ADAM and shows better performance than the single SGD and ADAM, which shows the power of mixed optimizers. Also the normalization version of the second mixed optimizer we developed showed better performance than the original one they proposed in the paper.

References

[1] N. Landro, I. Gallo, and R. L. Grassa, "Mixing adam and sgd: a combined optimization method," 2020.

[2] N. S. Keskar and R. Socher, "Improving generalization performance by switching from adam to sgd," *arXiv preprint arXiv:1712.07628*, 2017.