Major Neural Network Optimizers

Convolutional Neural Networks on CIFAR-10 Dataset

Group 2:
Ziqing Lu
Vignesh Murali
Ritvik Reddy

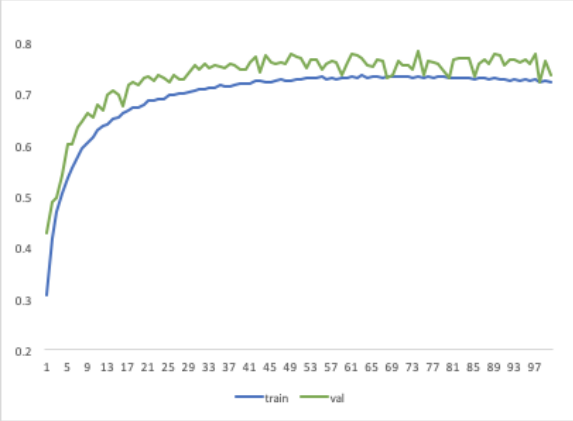
Abstract

In neural network training, a lot of time was wasted on disturbing directions as the error surface in space is usually ill-conditioned. There are many optimizers helps with this situation and speed up the training. In this report, we summarized the concepts and benefits of 6 major optimizers, and compared their learning effects based on an experiment testing different setting of optimizers on a Convolutional Neural Networks (CNN) using CIFAR-10 dataset.

To speed up the training progress, we need to move more quickly in directions with small but consistent gradients, more slowly in directions with big but inconsistent gradients. [1] There are mainly two ideas to achieve this: 1. use the velocity to direct to the right direction 2. use adaptive learning rate.

RMSprop

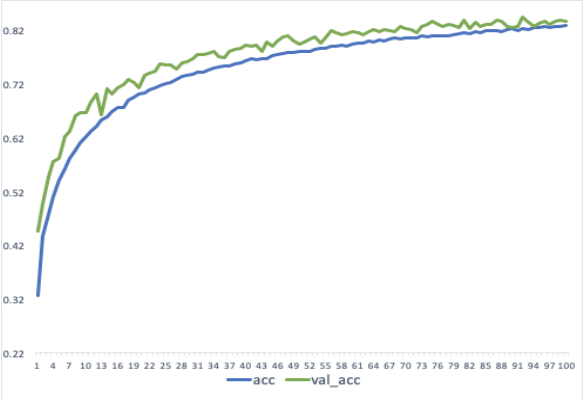
- Keras: keras.optimizers.RMSprop(lr=0.0001, rho=0.9, epsilon=None, decay=0.0)[2]
- Concept: RMSprop divides the learning rate by an exponentially decaying average of squared gradients. This ensure the learning rate for the right direction is way bigger than the learning rate for the direction that causes oscillation. This optimizer is usually a good choice for recurrent neural networks.[3]



[1] Geoffrey Hinton: *Neural Networks for Machine Learning*
[2] *Keras Document*
[3] *Keras Document*

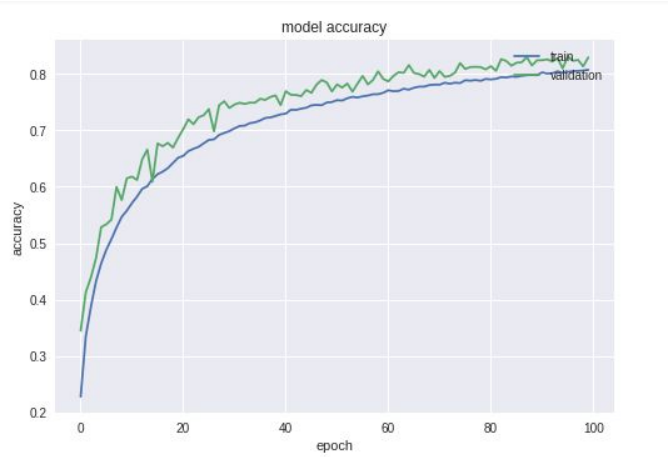
Adam

- Best Model: keras.optimizers.Adam(lr=0.0001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
- Concept: Adam is the abbreviation for Adaptive Moment Estimation, another method that computes adaptive learning rates for each parameter. In addition, Adam also keeps an exponentially decaying average of past gradients, similar to momentum. It makes the algorithm keep going in the previous direction and dampens the oscillations. So Adam is a combination optimizer of Momentum and RMSprop.



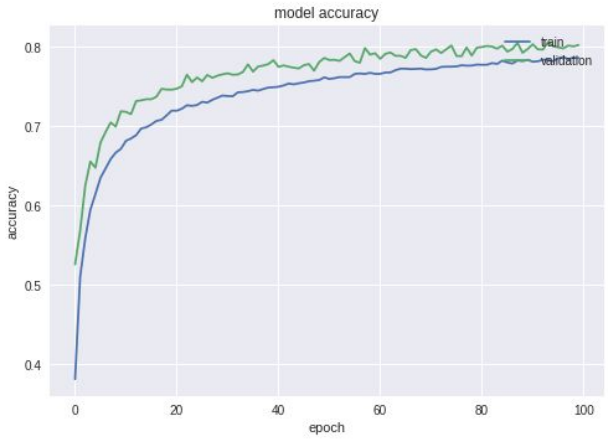
SGD (Stochastic Gradient Descent)

- Best Model: sgd =Keras.optimizers.SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)
- Concept: SGD, or stochastic gradient descent, is the "classical" optimization algorithm. In SGD we compute the gradient of the network loss function with respect to each individual weight in the network. Each forward pass through the network results in a certain parameterized loss function, and we use each of the gradients we've created for each of the weights, multiplied by a certain learning rate, to move our weights in whatever direction its gradient is pointing.



Adagrad

- Best Model: adagrad = keras.optimizers.Adagrad(lr=0.01, epsilon=None, decay=0.0)
- Adagrad is a more advanced machine learning technique (relative to SGD) which performs gradient descent with a variable learning rate. Node weights which have historically had large gradients are given large gradients, whilst node weights that have historically had small gradients are given small gradients. Thus Adagrad is effectively SGD with a per-node learning rate scheduler built into the algorithm.

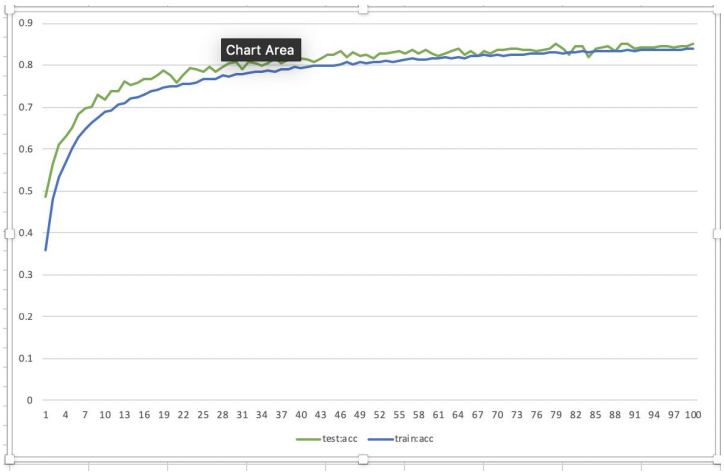


Adadelta

- keras.optimizers.Adadelta(lr=0.01, decay=1e-6)
  Adadelta optimizer. Adadelta is a more robust extension of Adagrad that adapts learning rates based on a moving window of gradient updates, instead of accumulating all past gradients. This way, Adadelta continues learning even when many updates have been done. Compared to Adagrad, in the original version of Adadelta, you don't have to set an initial learning rate. In this version, the initial learning rate and decay factor can be set, as in most other Keras optimizers.

Adamax

- keras.optimizers.adamax(lr=0.001, decay=1e-6)
- Adagrad is an optimizer with parameter-specific learning rates, which are adapted relative to how frequently a parameter gets updated during training. The more updates a parameter receives, the smaller the updates.



Observations

Optimizer Performance Table

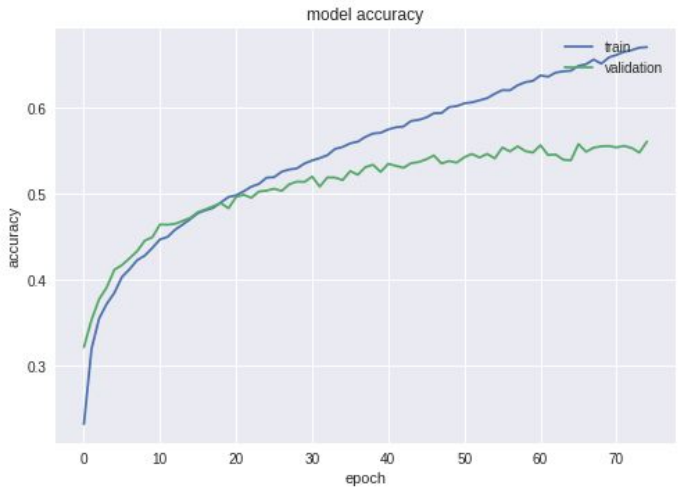
No.	Optimizer	Learning Rate	Test Loss	Test Accuracy	NoteBook
01	RMSprop	0.0001	0.8067	0.7382	Assignment1_Part2_Adam_RMSprop_1
02	Adam	0.0001	0.4856	0.8356	Assignment1_Part2_Adam_RMSprop_1
03	RMSprop	0.001	1.8285	0.3002	Assignment1_Part2_Adam_RMSprop_2
04	Adam	0.001	0.6223	0.7963	Assignment1_Part2_Adam_RMSprop_2
05	RMSprop	0.01	14.506	0.1	Assignment1_Part2_Adam_RMSprop_3
06	Adam	0.01	2.3046	0.1	Assignment1_Part2_Adam_RMSprop_3
07	SGD	0.0001	1.7100	0.3903	Assignment-1-PartA-SGDandAdagrad

08	AdaGrad	0.0001	1.6615	0.4086	Assignment-1-PartA-SGDandAdagrad
09	SGD	0.001	0.9965	0.6468	Assignment-1-PartA-SGDandAdagrad
10	AdaGrad	0.001	1.5146	0.4491	Assignment-1-PartA-SGDandAdagrad
11	SGD	0.01	0.4942	0.8294	Assignme8nt-1-PartA-SGDandAdagrad
12	AdaGrad	0.01	0.5684	0.8027	Assignment-1-PartA-SGDandAdagrad
13	SGD w/ Nesterov Momentum	0.01	0.5539	0.8085	Assignment-1-PartA-SGDandAdagrad
14	Adadelta	0.0001	2.02	0.2816	Ass1_Optimizer1
15	Adadelta	0.001	1.60100	0.4196	Ass1_optimizer1.1
16	Adadelta	0.01	1.0567	0.6263	Ass1_Optimizer1.2
17	Adamax	0.0001	0.533	0.8196	Ass1_Optimizer2
18	Adamax	0.001	0.446	0.851	Ass1_Optimizer2.1
19	Adamax	0.01	2.302	0.10	Ass1_Optimizer2.2

Conclusion

Firstly, the best performers are Adamax, Adam, SGD, AdaGrad. Most of them have applied adaptive learning rate, which will help find the right direction and learn faster.

Base on the experiments, we find that generally learning rate will affect the performance in different way, for example, for RMSprop and Adam, the performance increased with the drop of learning rate, for SGD and AdaGrad, the performance reduced with the drop of learning rate (Possibly due to underfitting). However Adamax works well with both higher & lower learning rates (0.001 and 0.01).



Best MLP Model - Accuracy Chart

Compared with MLP classifier, this Convolutional Neural Network is significantly better in terms with accuracy level. (MLP: ~ 0.5%, CNN: ~ 0.8%) CNN is a more efficient way of classifying images because it is better at processing spatial invariances and natural spatial hierarchies such as pixels,edges and textures.