Analysing the performance of neural nets

Joakim Olsen

February 2019

Introduction

In this report, we will look at the performance of different neural nets, to try and understand how different techniques and tricks can improve the performance. We will build two types of neural nets. First we will train a standard neural net to the MNIST-data. This data set consists of 70K digit images, with 28x28 pixels. There are different digits, thus 10 classes, and 60K images will be used to train the neural net, and 10K images will be used to test the neural nets. Our primary goal is to find a neural net that can classify with an error lesser than 0.75%. Secondly, we will train a convolutional neural network to the CIFAR-data. This data set consists of 60K pictures, with 32x32 pixels in RGB. The pictures contains one of 10 different objects/animals, thus 10 classes, and 50K images will be used to train the neural net, and 10K images will be used to test the neural net. Our goal here is to find a convolutional neural net that classifies with an error lesser than 10%.

Standard neural net with MNIST data

Our starting point is the neural net that is first proposed as an example on the github page posted on Poliformat. This is a neural net with two hiddel relu layers, which both have 512 output nodes, and a final softmax layer with 10 nodes, since there are 10 classes. Batch size is 128, it has 25 epochs and learning rate parameter 0.1. From here we apply one by one techniques and tricks that we have discussed in class, to see how they influence the performance. Table 1 contains a summary of the results, while the changes done for each run is explained point-wise below.

1. Standard nerual network proposed in the first example on github.
2. Increasing output nodes to 1024 in both hidden layers.
3. Adding batch norm to the relu activation functions.
4. Adding gaussian noise regularizer with parameter 0.3 and input noise.
5. Adding dropout with parameter 0.1.
6. Setting dropout parameter to 0.02.
7. Increasing to 75 epochs.
8. Adding learning rate scheduler with learning rate 0.1 before epoch 25, 0.01 before epoch 50 and 0.001 in the end.
9. Adding a hidden relu layer.
10. Adding data augmentation using known transformations and parameters 0.1.
11. Reducing batch size to 100.
12. Reducing GN parameter to 0.1.

Run	Error (%)
1	1.95
2	1.78
3	1.67
4	1.41
5	1.54
6	1.43
7	1.17
8	1.08
9	1.04
10	0.79
11	0.74
12	0.64

Table 1: Summary of performance history for the normal neural network and MNIST data.

Conclusion

We see from Table 1 that the improvement happens little by little, and there are no method that improves the model a lot, but many small tricks which improve the model a little. Put together however, the model can improve a lot. Increasing output nodes in the hidden layer has a good impact, showing that increasing the complexity of the model is good. Batch norm and gaussian noise regularizer, thus reducing over-fitting, improves the model a lot. Increasing epochs, and adding learning rate scheduler also improves the model a lot, showing that more time and adjusted step lengths can improve the model. Finally adding data augmentation works very well, improving the model substantially. After these steps, the goal of 0.75 % is achieved by adjusting some parameters. This shows that the choice finding the best parameters is also a part of the process.

Convolutional neural net with CIFAR data

Our starting point here as well is the first example code proposed at the github page. This is a convolutional neural network where one layer consists of applying a number of 3x3 2D filters, then a relu function and finally adding a max-pooling of size 2x2. The network has five of these layers with respectively 32, 64, 128, 256 and 512 filters, one standard relu layer with 512 output nodes, and a final softmax layer with 10 output nodes, as there are ten classes. It has batch norm, gaussian noise regulation with parameter 0.3, batch size 100, 75 epochs and learning rate parameter 0.1. We then follow the same procedure as with MNIST, adding techniques and tricks one by one to see how they influence the performance of the neural net. Table 2 contains a summary of the results, while the changes done for each run is explained point-wise below.

1. This is the convolutional network proposed in the example on github.
2. Adding learning rate scheduler with learning rate 0.1 before epoch 25, 0.01 before epoch 50 and 0.001 in the end.
3. Setting GN parameter to 0.1
4. Setting GN parameter to 0.03
5. Setting GN parameter back to 0.3 and adding data augmentation using known transformations and parameters 0.2.
6. Setting data augmentation parameters to 0.1.
7. Setting data augmentation parameters to 0.08.
8. Setting data augmentation parameters to 0.12.
9. Setting data augmentation parameters back to 0.1, and adding feature normalization to data augmentation.
10. Adding rotation range with parameter 20 to the data augmentation.
11. Adding zoom range to data augmentation with parameters 1.0 and 1.2.
12. Adding another 3x3 convolution to the standard layer block.
13. Increasing number of epochs to 150.
14. Removing rotation and zoom range since this made the performance worse.

Run	Error (%)
1	22.79
2	17.82
3	18.58
4	18.64
5	15.55
6	14.28
7	14.56
8	14.97
9	13.25
10	14.97
11	14.47
12	11.16
13	9.56
14	9.00

Table 2: Summary of performance history for convolutional network and CIFAR data.

Conclusion

Here as well, we can improve the performance of the neural net a lot, not by a singe method, but little by little, with many methods and tricks. We first see again the importance of a learning rate scheduler. Data augmentation is the next trick that improves the model a lot. Adding another convolution to each layer block, thus increasing flexibility, also improves the model a lot. Finally, the number of epochs is increased, which also improves the model, again showing the effect of giving the training process more time. A lot of parameter adjusting has been done, in addition to adding extra features to the data augmentation, without improving the model. It is however possible to see that the choice of parameters have an impact, and is therefore an important part of the process.