RNA LAB

Jaime Ferrando Huertas

January 2021

Contents

1 Introduction **1**

2 MNIST **1**

3 CIFAR **4**

4 Conclusion **6**

1 Introduction

This report contains RNA lab assignment. The assignment requires us to solve two problems, achieving 0.8% classification error on the MNIST dataset and 5% on the CIFAR dataset. The report contains an overview of the methods we used to achieve this along with others that did not work as expected, we provide a comparison of those methods and an explanation of their results. All code can be found along with the deliverable of this report and results can be reproduced running the necessary scripts.

2 MNIST

MNIST is an image dataset of handwritten digits, has a training set of 60.000 images, and a test set of 10.000 examples. It contains 10 classes (0..9) and each image has a resolution of 28*28.

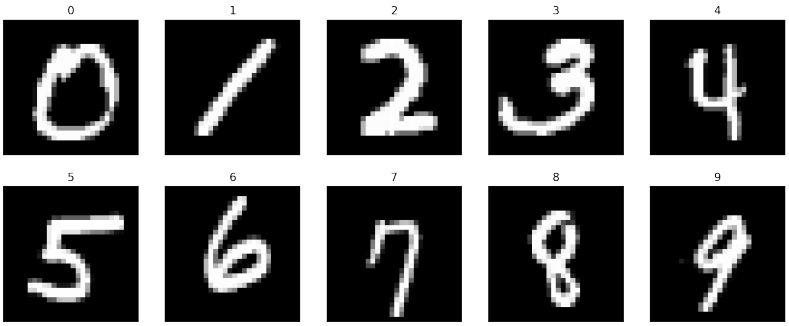

Figure 1: Sample of MNIST dataset

For this assignment we were asked to achieve an error rate lower than 0.8% without using convolutional networks, we did this with a neuronal network consisting of one noisy block (Figure 2) and one output layer with 10 neurons to represent the probability of each class. A noisy block consists of 1 linear layer of 4096 neurons followed where we apply batch normalization and random Gaussian noise, we use relu as activation function on the result of this layer.

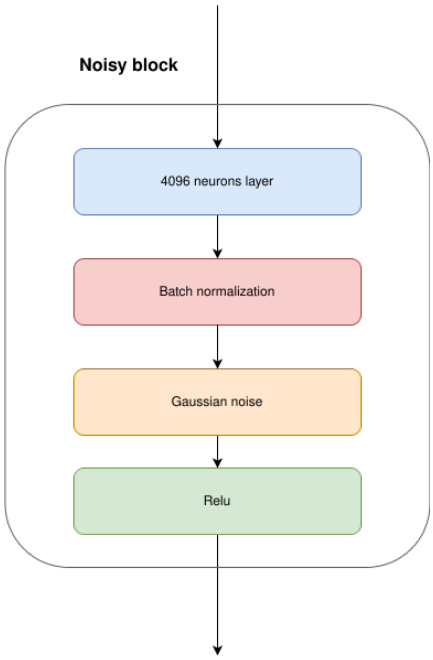

Figure 2: Noisy block structure

For the data pipeline during training we used a technique called data augmentation that consists of applying random small modifications to our images

as we feed them into the model, allowing us to artificially expand the size of our dataset. We applied random rotations to the image with a random affine. For training, we used the SGD optimizer with an initial learning rate of 0.1 which we reduced by a factor of 0.1 on epochs 75, 125, 175.

Achieving these results was not straight forward, we first tried with normal multi layer neuronal networks with a constant learning rate but results would not pass the 2% error rate. With the addition of batch normalization and Gaussian noise in the form of noisy blocks, we saw an increase in performance to 1.4% when using 4 noisy blocks. It was when we tweaked the learning rate to reduce as epochs increased that we achieved less than 0.8% error. We finally tried to reduce the number of noisy blocks to 1 to see if it will still achieve this error rate and the model did at the cost of a larger number of epochs. We choose to deliver this 1 noisy block model as with it we reduce model perplexity and not many reports will include a 1 layer solution to this problem.

We also tried to solve the assignment with convolutional networks to be able to compare our implementation, this was an easy task and was completed with the first model we implemented with just a 4 layer neural network (2 convolutions + 2 linear layers with dropout) structure. Here is a table with a summary of our experiments that reached the requires 0.8% error rate and graphs of our training/testing loss accuracy of the 1 noisy block model

Experiment	Min. error	Epochs required
4 Noisy blocks	0.76%	38
1 Noisy blocks	0.78%	144
Convolutional NN	0.77%	14

Table 1: MNIST successful experiments

Now you can find the exact loss and accuracy during training of our 1 noisy block model.

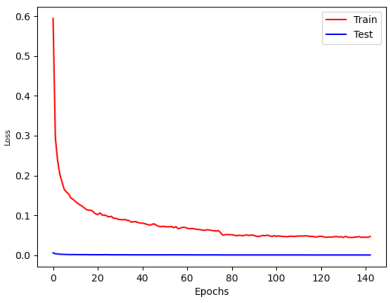

Figure 3: Loss

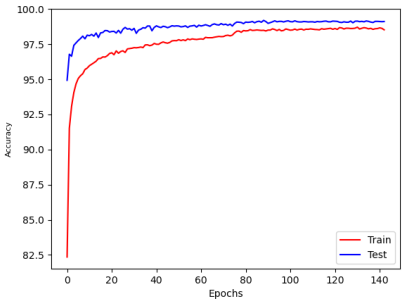

Figure 4: Accuracy

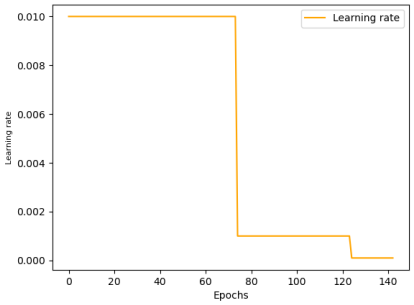

Figure 5: Loss

3 CIFAR

CIFAR (CIFAR-10) is an image dataset of 10 different classes, has a training set of 50.000 images, and a test set of 10.000 examples. It contains 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck) and each image has a resolution of 32*32.

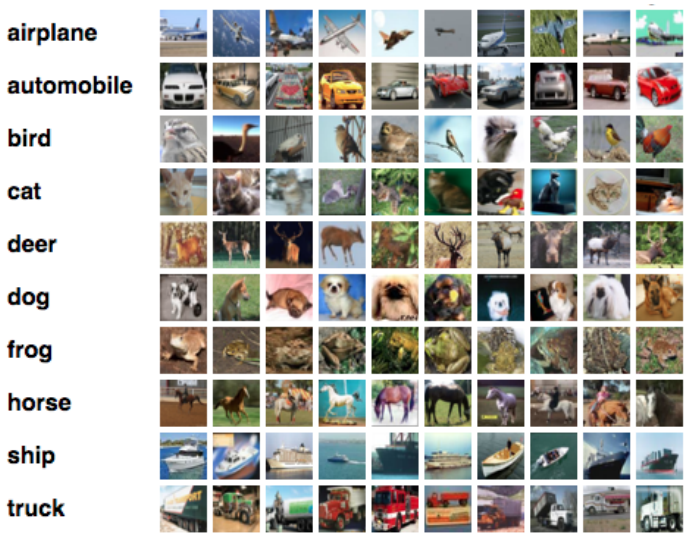

Figure 6: Sample of CIFAR dataset

With CIFAR we needed to achieve less than a 5% error rate with any neuronal network architecture. We did this by implementing a resnet18 equivalent architecture.

For the data pipeline during training, we used data augmentation again. We applied random crops followed by random horizontal.

To solve this problem we first tried to implement a similar structure to the noisy block but changing the linear layer with a convolutional and adding a max-pooling after the relu activation, stacking 5 of these blocks got us to 8% error rate. We tried a larger number of blocks but it did not report any improvements as we were probably facing the vanishing gradient problem due to our large network. To solve this we implemented the resnet18[1] (Figure 5) architecture described in lectures and achieved 7% error rate, this was using the same optimizer and learning rate scheduler as we did in MNIST. Since we were really close to the 5% we decided to try to optimize our learning rate scheduler a bit more as it worked for us in the MNIST assignment. Changing the epoch number where our learning rate will decrease to 100, 175, 250 gave us the 5% error rate we were looking for, we think this is due to both the data and the network being far more complex than our previous work at MNIST, requiring more time to train.

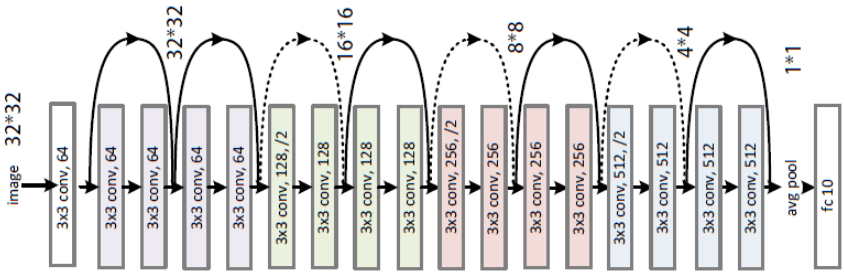

Figure 7: Resnet structure

Afterwards, we also tried OneCycleLR [2] with a maximum learning rate of 0.1 and 200 epochs length and we managed to achieve 5% at a lower epoch number (187). This is a learning rate I have used a lot of times to debug and tweak network architectures as it helps reduce training time while maintaining acceptable performance. Here is a table with a summary of our experiments and graphs of our training/testing learning rate loss accuracy of resnet18 with OneCycleLR.

Experiment	Min. error	Epochs required
4 Noisy blocks - convolutional version	14.11%	300
8 Noisy blocks - convolutional version	15.81%	300
Resnet18 - SGD & Original MultiStep LR	7.20%	300
Resnet18 - SGD & Modified MultiStep LR	4.98%	288
Resnet18 - SGD & OneCycle LR	4.97%	187

Table 2: CIFAR experiments

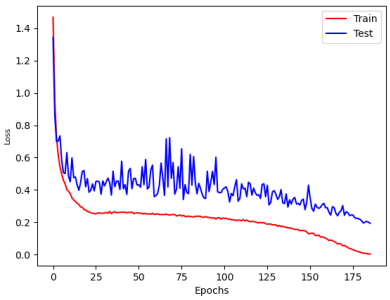

Figure 8: Loss

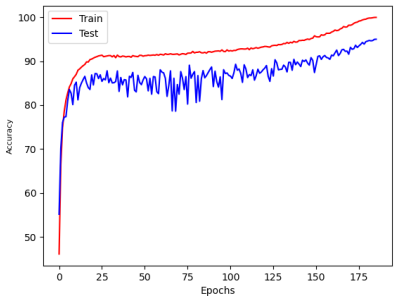

Figure 9: Accuracy

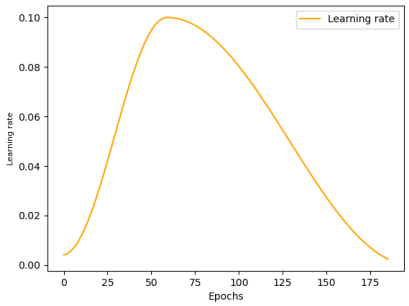

Figure 10: Learning rate

4 Conclusion

We thought the assignment was quite interesting and it was a really good point to constraint the MNIST section to not use convolutional layers, this made us learn more on how to optimize our network and also the first time we used Gaussian noise for training. With CIFAR it was even a harder task to solve as the modified noisy blocks with convolutional layers did not solve the problem, we spent a lot of time trying to optimize that architecture without success. Implementing resnet18 was a learning experience and we managed to solve the gradient vanishing problem that led us to 5% error rate.

We think we also could have reduced our error rate by using optimizers from the Adam family or trying different data augmentation techniques. The 4 noisy block model for MNIST could also converge sooner if we apply a reduction to the learning rate in early epochs as with the current setup it did not even reach the first reduction epoch at 75.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[2] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017.