Deep Learning
CS 898BD
Zachary Pearson
D439H842
Assignment 2

Introduction

The 15-scene dataset contains over 4000 images each classified according to the type of location they capture. This data, however, is not uniform in dimension and each classification is not represented equally. As a result, significant preprocessing of the data is required prior to training.

In this report, I discuss the type of pre-processing used, describe the particular architecture, and compare performance between SGD, Adam, and RMSprop optimizers.

Methodology

The data contained uneven classification distribution so the first thing done was to ensure that each class had equal number of instances. The class with the lowest number of examples held 210 instances. As a result, it was decided that each class would be pared down to 200 instances for the training data. This ensured balanced training data. Samples were removed from the bottom of the list, no other considerations were taken when deciding on what was removed. The resulting training data had a total of 3000 samples. To ensure that the training data represented 80% of all data used, 750 samples from the removed samples were taken for test data. The remaining data was not used in this assignment.

Of the data that was used for this assignment, their respective sizes all differed from each other. To ensure a uniform shape for all of the data, 220x220 was chosen as the dimension that all images would be cropped to fit. This size was chosen as it was the minimum size represented in the data.

Due to the relative limited number of samples, the use of image rotation and flipping was included to artificially inflate that variation between training samples of the same class. Rotation was limited to +-15% and horizontal flipping was applied randomly to the input.

Once the data was processed, it was used to train 3 models. Each model possessed the same architecture, but differed in the choice of optimizer used.

Batch size used was 32.

Architecture

The particular architecture used was taken from a kaggle notebook[1] that boasted a 70%+ accuracy on the 15-scene data using a CNN. This architecture can be broken down into 4 parts – 3 convolution sections and the dense classification section. Each convolution section takes on the same form – 2 convolutional layers followed max pool down sampling. Batch-normalization was applied after each convolutional layer. Once down sampling was completed, dropout was employed.

The convolutional sections all contained the same architecture but differed in the depth of the convolutional layers as well as with the drop-out rates. The first convolution section had each convolutional layer with a depth of 32 and a dropout rate of 20%. The second section had a depth of 64 and and a dropout rate of 30%. The final convolutional layer had a depth of 128 and a dropout rate of 40%.

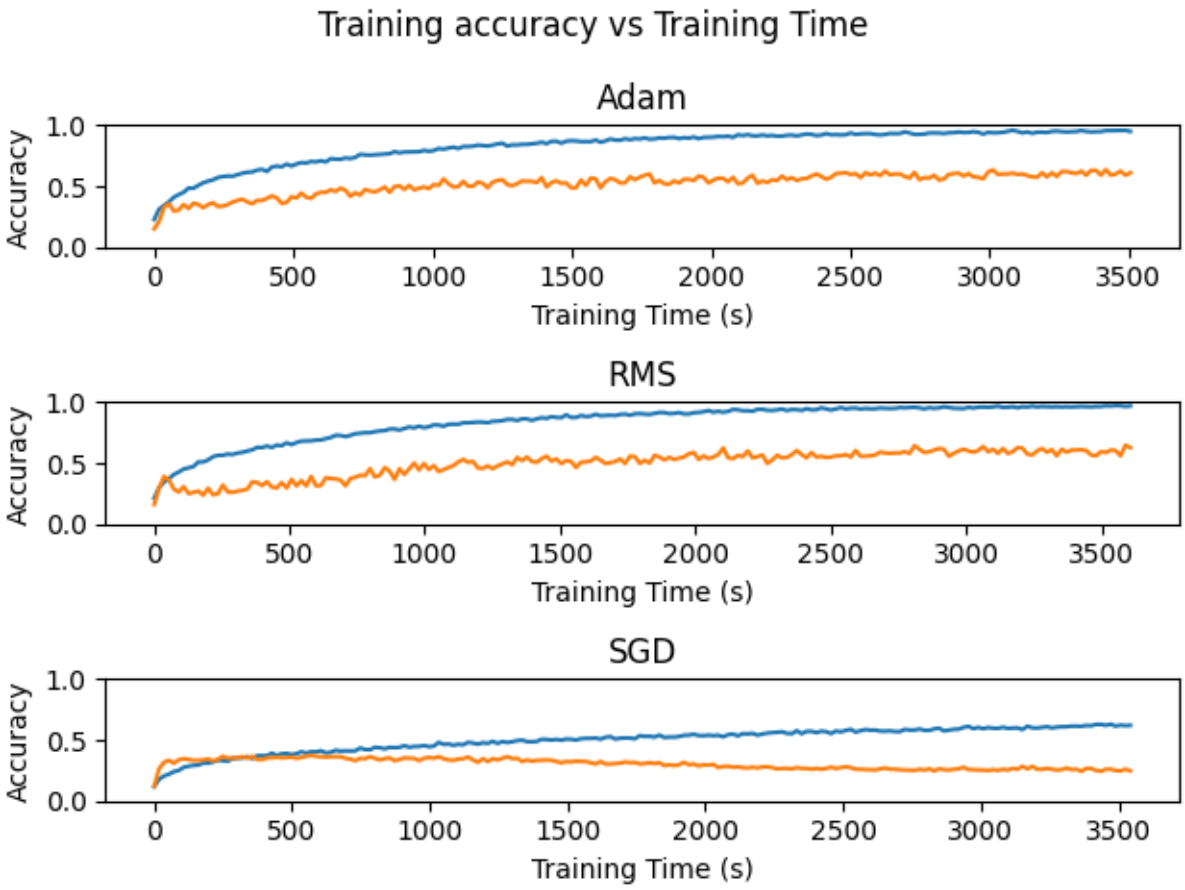
Results

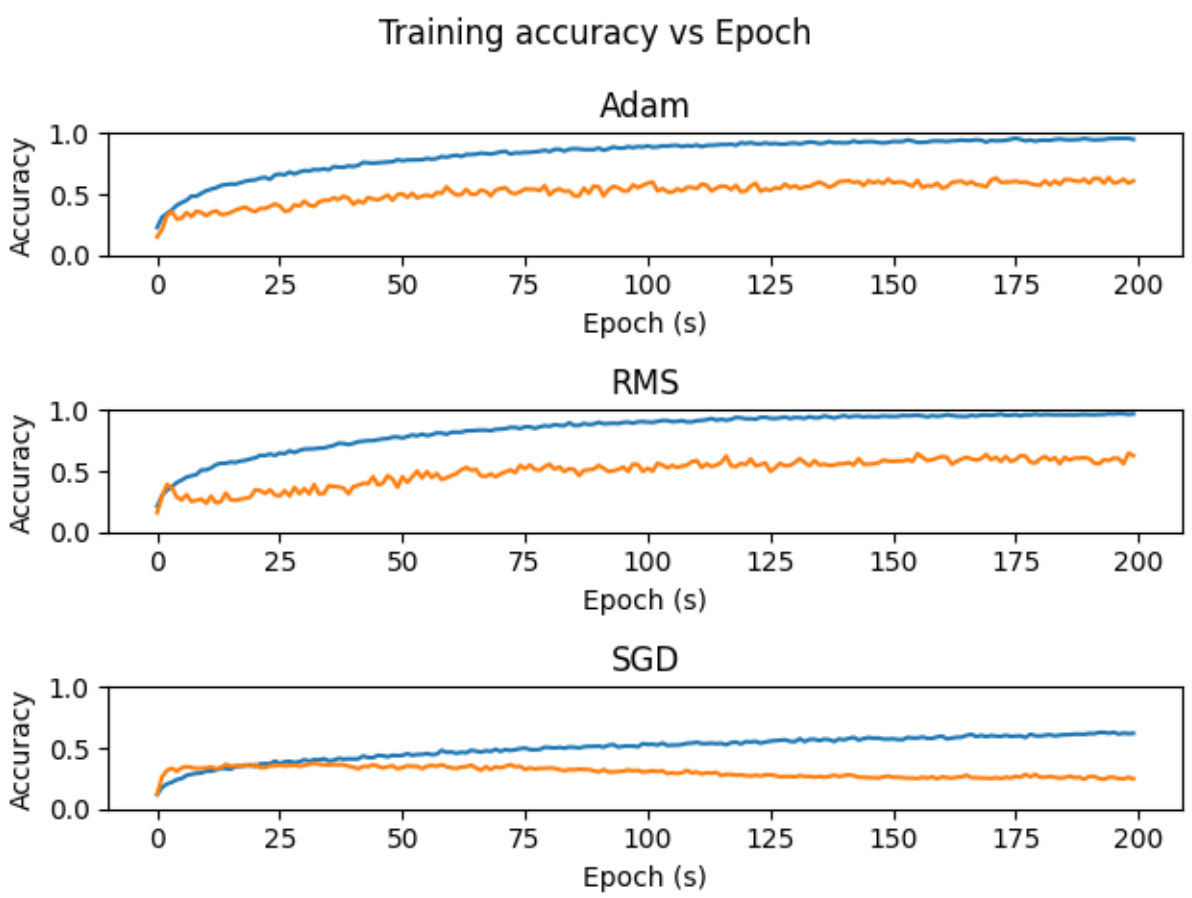
As can be seen from the figures (see figure section below), the average performance for the models using Adam and RMSprop for optimization was roughly 60% accuracy with loss leveling out after several epochs. However, the model utilizing SGD performed particularly poorly with loss increasing with each epoch and a constantly decreasing accuracy.

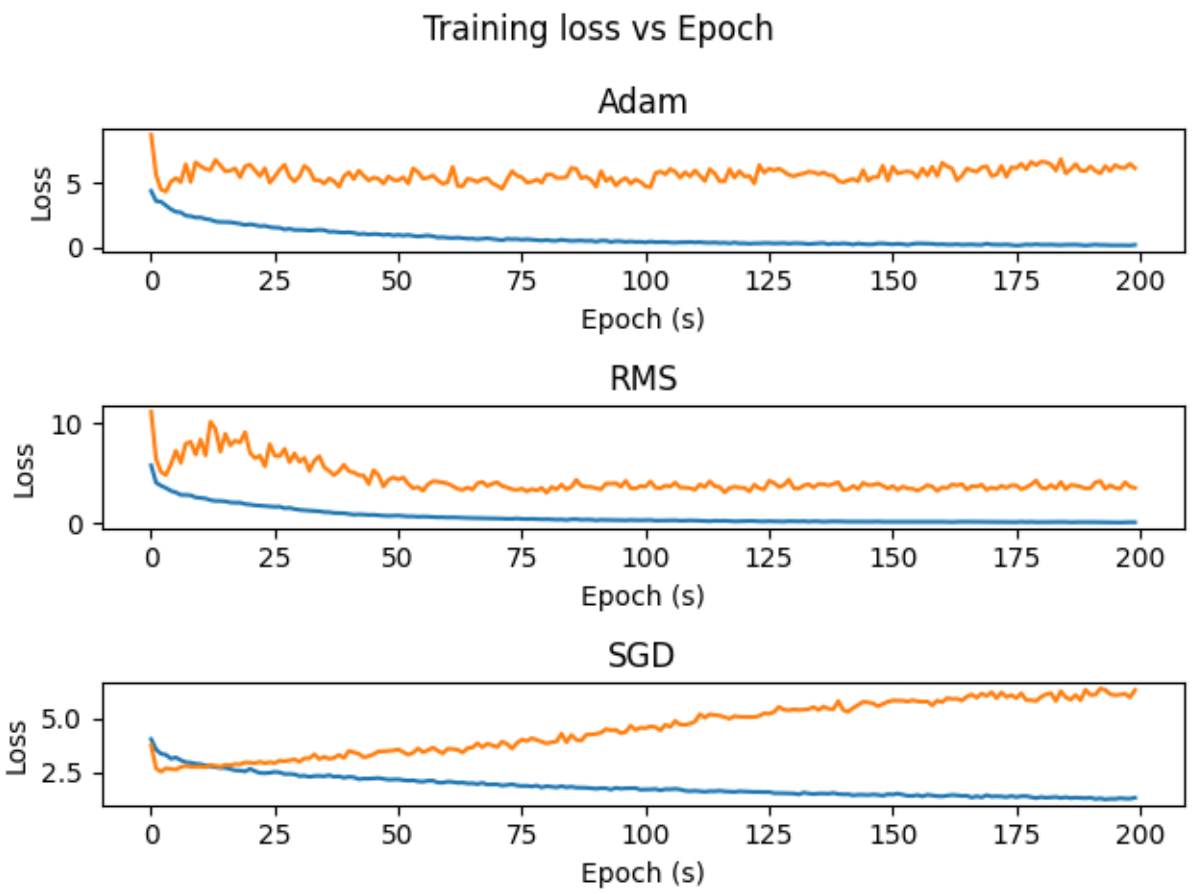
Conclusion

Though it is clear that in this particular instance Stochastic Gradient Descent did not perform well compared to Adam or Root Mean Squared Propagation, it is possible that a change in architecture or initial learning rate and decay could be made to improve performance

Figures


Training accuracy vs Training Time
Adam
Accuracy
1.0
0.5
0.0
0
500
1000
1500
2000
2500
3000
3500
Training Time (s)
RMS
Accuracy
1.0
0.5
0.0
0
500
1000
1500
2000
2500
3000
3500
Training Time (s)
SGD
Accuracy
1.0
0.5
0.0
0
500
1000
1500
2000
2500
3000
3500
Training Time (s)


Training accuracy vs Epoch
Adam
Accuracy
1.0
0.5
0.0
0
25
50
75
100
125
150
175
200
Epoch (s)
RMS
Accuracy
1.0
0.5
0.0
0
25
50
75
100
125
150
175
200
Epoch (s)
SGD
Accuracy
1.0
0.5
0.0
0
25
50
75
100
125
150
175
200
Epoch (s)



References

1) https://www.kaggle.com/code/minbavel/scene-15-cnn