Abhishek Bhagwat, NUID 00144721

CIFAR-10 Assignment Report on Multilayered Perceptron

Abstract: The purpose of this report is to build an MLP to classify the CIFAR-10 dataset. The dataset contains 60,000 images that are classified into 10 categories. The categories include airplanes, birds, cats etc. The photos are in red, blue and green components. They measure 32 by 32-pixel squares. For classifying these images, we build an MLP using Keras. We will be modifying the parameters of our MLP to obtain the best model and get maximum accuracy. The parameters we will be modifying are number of epochs, number of layers, activation functions, learning rate, number of neurons and batch size. We will build different models by varying the parameters and observe the changes that they induce in our accuracy and loss measure. Based on these observations we will select the best MLP for our classification problem. I have made a total of 9 models out of that I will summarize the best 8 models in the report.

Observations

Model 1

Architecture: The model consists of 3 dense layers.

Layers

1. Dense1 – 1024 neurons
2. Dense2 – 512 neurons
3. Dense3-512 neurons
4. Dense4(output)-10 neurons

Activation functions: Relu for dense layers. Softmax for output layer.

Dropout Rate: 0.2 for all layers

Epochs: 50

Batch: 32

Learning Rate: 0.01

Accuracy: Testing 55% and Training 61

Loss Rate: Testing 1.25 and Training 1.06

Observation: Since we Trained will less number of epochs our accuracy is not very good. We can improve by increasing the number of epochs and by adding more layers into our network.

Model 2

Model 2 has all the parameters same as the first model. We only increase the epochs to 100.

Architecture: The model consists of 3 dense layers.

Layers

1. Dense1 – 1024 neurons
2. Dense2 – 512 neurons
3. Dense3-512 neurons
4. Dense4(output)-10 neurons

Activation functions: Relu for dense layers. Softmax for output layer.

Dropout Rate: 0.2 for all layers

Epochs: 100

Batch: 32

Learning Rate: 0.01

Accuracy: Testing 56% and Training 75%

Loss Rate: Testing 1.29 and Training 1.03

Observation: The new parameter did not improve our training accuracy or loss. The model performed well for training but not for testing indicating that our model is overfitting.

Model 3

We add more layers and also increase the epochs. The network will have dense layers.

Architecture: The model consists of 3 dense layers.

Layers

1. Dense1 – 1024 neurons
2. Dense2 – 1024 neurons
3. Dense3-512 neurons
4. Dense4 -512 neurons
5. Dense5-512 neurons
6. Dense6 -512 neurons
7. Dense7(output) – 10 neurons

Activation functions: Relu for dense layers. Softmax for output layer.

Dropout Rate: 0.2 for all layers

Epochs: 150

Batch: 32

Learning Rate: 0.01

Accuracy: Testing 57% and Training 83%

Loss Rate: Testing 0.57 and Training 0.47

Observation: The new parameter did not improve our training accuracy or loss. The model performed well for training but not for testing indicating that our model is overfitting.

Model 4

We use the same architecture as the previous network. Adding more layers is not improving our network anymore. So, we keep the number or layers and neurons constant and we change the activation function to Leakyrelu.

Architecture: The model consists of 6 dense layers.

Layers

1. Dense1 – 1024 neurons
2. Dense2 – 1024 neurons
3. Dense3-512 neurons
4. Dense4 -512 neurons
5. Dense5-512 neurons
6. Dense6 -512 neurons
7. Dense7(output) – 10 neurons

Activation functions: LeakyRelu for dense layers. Softmax for output layer.

Dropout Rate: 0.2 for all layers

Epochs: 150

Batch: 32

Learning Rate: 0.01

Accuracy: Testing 58% and Training 80%

Loss Rate: Testing 1.04 and Training 0.5

Observation: Changing the activation function brought us the best accuracy for the test data. However, we can see vast difference in the testing and training accuracy. Our model is overfitting.

Model 5

We use the same architecture as the previous network. To handle overfitting, we increase the dropout to 0.50 for all the layers except the 6 th dense layer. For 6^{th} layer we do not keep any dropout. We also reduce the epochs since the graphs for the previous model shows that after 70 epochs there is no real improvement in the accuracy and loss for the testing data.

Architecture: The model consists of 6 dense layers.

Layers

1. Dense1 – 1024 neurons
2. Dense2 – 1024 neurons
3. Dense3-512 neurons
4. Dense4 -512 neurons
5. Dense5-512 neurons
6. Dense6 -512 neurons
7. Dense7(output) – 10 neurons

Activation functions: LeakyRelu for dense layers. Softmax for output layer.

Dropout Rate: 0.5 for all layers except 6^{th} layer.

Epochs: 70

Batch: 32

Learning Rate: 0.01

Accuracy: Testing 52% and Training 52%

Loss Rate: Testing 1.34 and Training 1.30

Observation: Changing the activation function brought us the best accuracy for the test data. However, we can see vast difference in the testing and training accuracy. Our model is overfitting.

Model 6

We use the same architecture as the previous network. To handle overfitting, we increase the dropout to 0.50 for all the layers except the 6 th dense layer. For 6^{th} layer we do not keep any dropout. We also reduce the epochs since the graphs for the previous model shows that after 70 epochs there is no real improvement in the accuracy and loss for the testing data.

Architecture: The model consists of 6 dense layers.

Layers

1. Dense1 – 1024 neurons
2. Dense2 – 1024 neurons
3. Dense3-512 neurons
4. Dense4 -512 neurons
5. Dense5-512 neurons
6. Dense6 -512 neurons
7. Dense7(output) – 10 neurons

Activation functions: LeakyRelu for dense layers. Softmax for output layer.

Dropout Rate: 0.5 for all layers except 6^{th} layer.

Epochs: 70

Batch: 32

Learning Rate: 0.01

Accuracy: Testing 52% and Training 52%

Loss Rate: Testing 1.34 and Training 1.30

Observation: Increasing the drop-out rate did seem to reduce overfitting to a considerable extent. So far this has been our best network.

Model 7

We use the same parameters and architecture as the previous model. We try by reducing the learning rate 0.001

Architecture: The model consists of 6 dense layers.

Layers

1. Dense1 – 1024 neurons
2. Dense2 – 1024 neurons
3. Dense3-512 neurons

4. Dense4 -512 neurons
5. Dense5-512 neurons
6. Dense6 -512 neurons
7. Dense7(output) – 10 neurons

Activation functions: LeakyRelu for dense layers. Softmax for output layer.

Dropout Rate: 0.5 for all layers except 6th layer.

Epochs: 70

Batch: 32

Learning Rate: 0.001

Accuracy: Testing 52% and Training 53%

Loss Rate: Testing 1.35 and Training 1.30

Observation: Changing the learning rate did contribute much to the improvement of our network.

Model 8

We try by changing the activation function

Architecture: The model consists of 6 dense layers.

Layers

1. Dense1 – 1024 neurons
2. Dense2 – 1024 neurons
3. Dense3-512 neurons
4. Dense4 -512 neurons
5. Dense5-512 neurons
6. Dense6 -512 neurons
7. Dense7(output) – 10 neurons

Activation functions: tanh for dense layers. Softmax for output layer.

Dropout Rate: 0.5 for all layers except 6th layer.

Epochs: 70

Batch: 32

Learning Rate: 0.001

Accuracy: Testing 45% and Training 43%

Loss Rate: Testing 1.55 and Training 1.58

Observation: The accuracy score has gone down and the loss rate has increased compared to our previous network.

Conclusion: Model 6 has given the best performance. Increasing the drop out rate according to me was the main factor that improved this model over the other models. The architecture consisting of 6 dense layers was the optimum after trying different combinations of layer numbers and neurons. The activation function leaky relu seemed to have performed better than relu. It had epochs of 70 and the batch size of 32. The model performs well in terms that it gives the same accuracy and loss for both training and testing data. According to me the architecture and the number of epochs are optimal. To further improve the performance, I believe that we can tweak the dropout rates for individual layers and also try with different learning rates.