Layer API - A Simple Neural Network

Group 18: Akhilesh Devrari, Chirag Vashist, Prabhakar Prasad

March 15, 2019

Contents

1 Aim **3**

2 Datasets **3**

3 Code Specifications **3**

4 Dataset-1 : 28*28 MNIST **3**

4.1 Dataset Preparation 3

4.2 Classifier-1 4

4.2.1 Network Architecture 4

4.2.2 Results 4

4.2.3 Inferences 4

4.3 Classifier-2 6

4.3.1 Network Architecture 6

4.3.2 Results 6

4.4 Classifier-3 7

4.4.1 Network Architecture 7

4.4.2 Results 7

5 Dataset-2 : 28*28*3 Coloured Lines on Black Background Dataset **8**

5.1 Dataset Preparation 8

5.2 Classifier-1 8

5.2.1 Network Architecture 8

5.2.2 Results 8

5.2.3 Inferences 10

1 Aim

Given two image data-sets, we were required to use code a Layer API to build a fully-connected neural network, in order to classify the images.

2 Datasets

We are given the following dataset :

1. 28*28 MNIST Dataset.
2. 28*28*3 Coloured Lines on Black Background (CLBB) Dataset.

3 Code Specifications

We were successfully able to code the required API by using Python Library Numpy. No other curated neural networks library like TensorFlow, Keras or PyTorch were used.
We constructed two major classes :

1. Layer Class : This class is used to design individual layers of the neural network. Using this class the user can decide the activation functions and the dropout probability.
2. Model Class : This class is used to tune the entire network as a whole. Using this class, the user can decide the Optimizer, the learning rate, the number of maximum iterations.

The various specifications provided in the Layers API :

1. Activation Functions : Sigmoid, Rectified Linear Unit, tanh, Softmax.
2. Regularization : Inverted Dropout.
3. Optimizer : Gradient Descent, Gradient Descent with Momentum, RMS Prop, Adagrad and Adam.
4. Loss Function : Cross-entropy.

4 Dataset-1 : 28*28 MNIST

4.1 Dataset Preparation

The data-set contained 70000 images of 28*28*1 Black-and-White images. In order to prepare the data, we first flattened each data image to a single vector with pixel values between 0 and 1. This resulted in a 784-dimensional feature representing a single image. Next we partitioned data into training and testing set.

1. Total Data-Samples : 70000
2. Training Samples : 60000
3. Testing Samples : 10000

4.2 Classifier-1

4.2.1 Network Architecture

1. Layer-1 : 784 neurons. ReLU activation
2. Layer-2 : 196 neurons. ReLU activation
3. Layer-3 : 58 neurons. ReLU activation
4. Layer-3 : 10 neurons. Softmax activation

The learning rate was set to 0.001. The desired accuracy was set to 99% and number of maximum iterations to 100.

4.2.2 Results

$$\text{Confusion Matrix} = \begin{bmatrix} 965 & 0 & 1 & 2 & 1 & 4 & 2 & 2 & 1 & 2 \\ 0 & 1127 & 0 & 1 & 0 & 1 & 2 & 2 & 2 & 0 \\ 5 & 4 & 1000 & 5 & 5 & 0 & 4 & 2 & 6 & 1 \\ 0 & 1 & 3 & 975 & 0 & 15 & 0 & 4 & 5 & 7 \\ 0 & 0 & 1 & 0 & 967 & 0 & 5 & 1 & 1 & 7 \\ 3 & 0 & 0 & 3 & 1 & 876 & 2 & 1 & 4 & 2 \\ 4 & 3 & 1 & 0 & 5 & 23 & 920 & 0 & 2 & 0 \\ 0 & 5 & 7 & 4 & 3 & 1 & 0 & 998 & 0 & 10 \\ 4 & 0 & 2 & 6 & 3 & 12 & 3 & 3 & 933 & 8 \\ 2 & 3 & 0 & 4 & 10 & 6 & 0 & 3 & 1 & 980 \end{bmatrix}$$

1. Training Accuracy : 99.05%
2. Testing Accuracy : 97.41%

4.2.3 Inferences

1. It is evident from the Confusion Matrix that a major source of inaccuracy is images of Class-6 being classified into Class-5. This makes sense as both the number have a very similar structure.
2. Using this model, we are able to get a good amount of training accuracy : 99.055%, but the test accuracy is lower, around 97.41%. Thus although this is not a significant issue, but this indicates that the model is starting to over-fit.

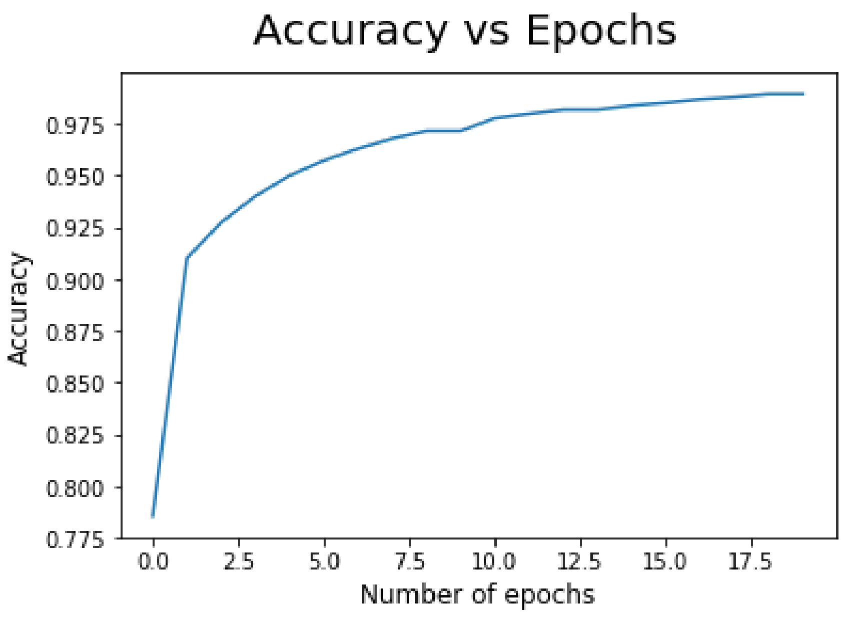

Figure 1: MNIST Data-Set : Accuracy vs Epochs

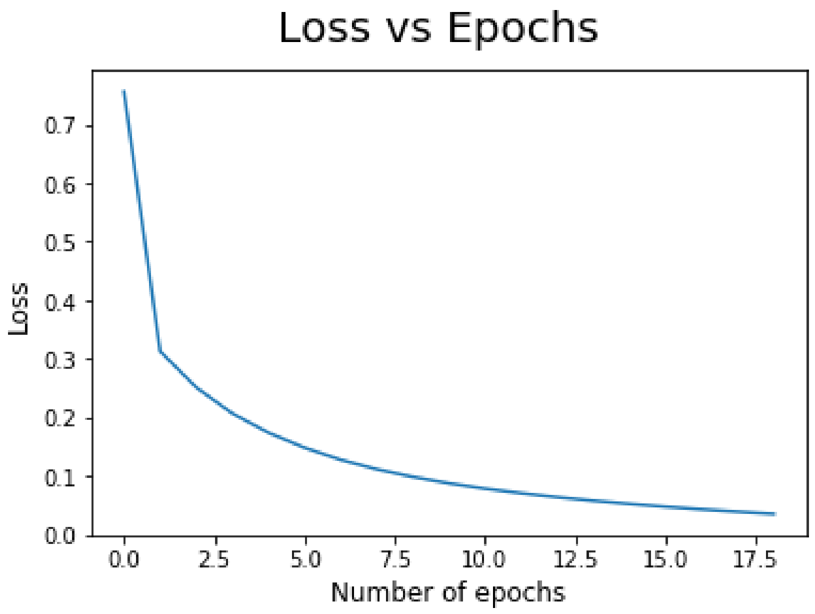

Figure 2: MNIST Data-Set : Loss vs Epochs

4.3 Classifier-2

In order to tackle the overfitting in the previous model, we introduced drop-out regularization in hopes of getting better performance.

4.3.1 Network Architecture

1. Layer-1 : 784 neurons. ReLU activation. Keep Probability : 0.75
2. Layer-2 : 196 neurons. ReLU activation. Keep Probability : 0.9
3. Layer-3 : 58 neurons. ReLU activation. Keep Probability : 1.0
4. Layer-3 : 10 neurons. Softmax activation

The learning rate was set to 0.001. The desired accuracy was set to 99% and number of maximum iterations to 100.

4.3.2 Results

1. Training Accuracy : 93.18%
2. Testing Accuracy : 93.4%

As we can, by using drop-out regularization, we are able to bring down the difference between training and testing accuracy. The model was trained for a handful number of epochs, but I am sure that better accuracy could have been obtained by using more number of epochs.

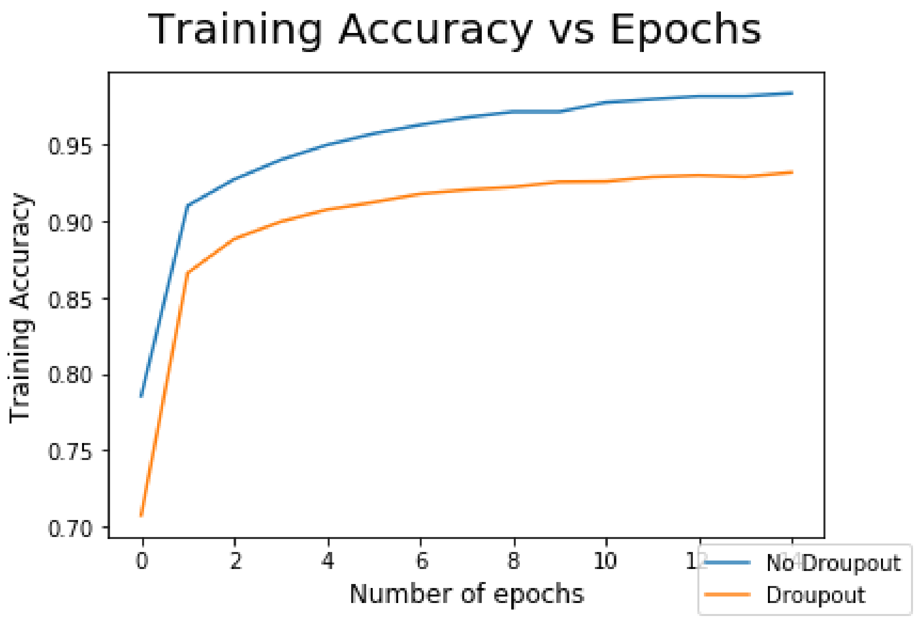

Figure 3: MNIST Data-Set Accuracy : Dropout vs No-dropout

4.4 Classifier-3

In order to explore the convergence rate of various optimizer, we trained a model on RMS Prop optimizer.

4.4.1 Network Architecture

1. Layer-1 : 784 neurons. ReLU activation.
2. Layer-2 : 196 neurons. ReLU activation.
3. Layer-3 : 58 neurons. ReLU activation.
4. Layer-3 : 10 neurons. Softmax activation

The learning rate was set to 0.001. The desired accuracy was set to 99% and number of maximum iterations to 100. However, rather than using gradient descent optimizer, we used a RMS Prop as an optimizer.

4.4.2 Results

1. Training Accuracy : 98.58%
2. Testing Accuracy : 97.14%

It is very obvious from the graph that using RMS Prop optimizer, we are able to converge in lesser number of epochs compared to using Gradient Descent. However, not all optimizers are always very good. Some optimizers like Adam took a lot of time for one epoch. This is due to the various computations happening during the back-propogation step.

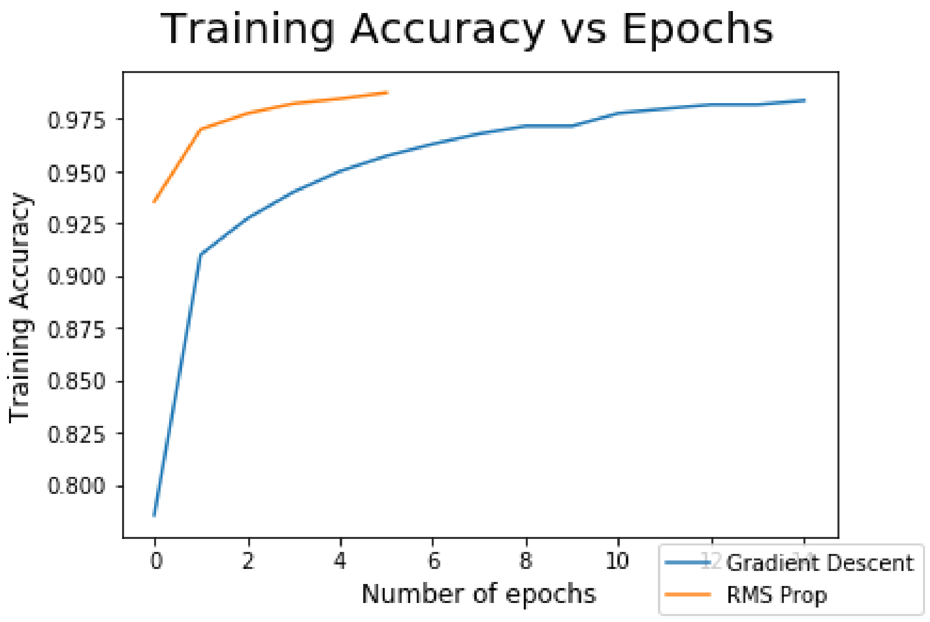

Figure 4: MNIST Data-Set Accuracy : Gradient Descent vs RMS Prop

5 Dataset-2 : 28*28*3 Coloured Lines on Black Background Dataset

5.1 Dataset Preparation

The data-set contained 70000 images of 28*28*3 Black-and-White images. In order to prepare the data, we first flattened each data image to a single vector with pixel values between 0 and 1. This resulted in a 2352-dimensional feature representing a single image. Next we partitioned data into training and testing set.

1. Total Data-Samples : 96000
2. Training Samples : 72000
3. Testing Samples : 24000

5.2 Classifier-1

5.2.1 Network Architecture

1. Layer-1 : 2352 neurons. ReLU activation
2. Layer-2 : 196 neurons. ReLU activation
3. Layer-3 : 96 neurons. Softmax activation

The learning rate was set to 0.001. The desired accuracy was set to 99% and number of maximum iterations to 100.

5.2.2 Results

Since the data-set has 96 classes, it is impossible to show the F-Measure and the Confusion Matrix in this report. Rather, we can show other data about the model :

1. Training Accuracy : 99.02%
2. Testing Accuracy : 98.9%

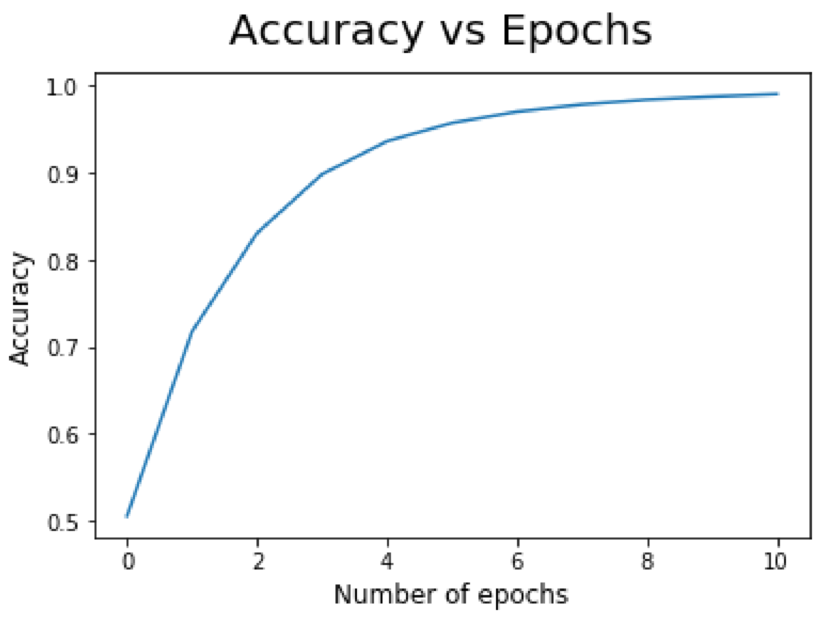

Figure 5: CLBB Data-Set : Accuracy vs Epochs

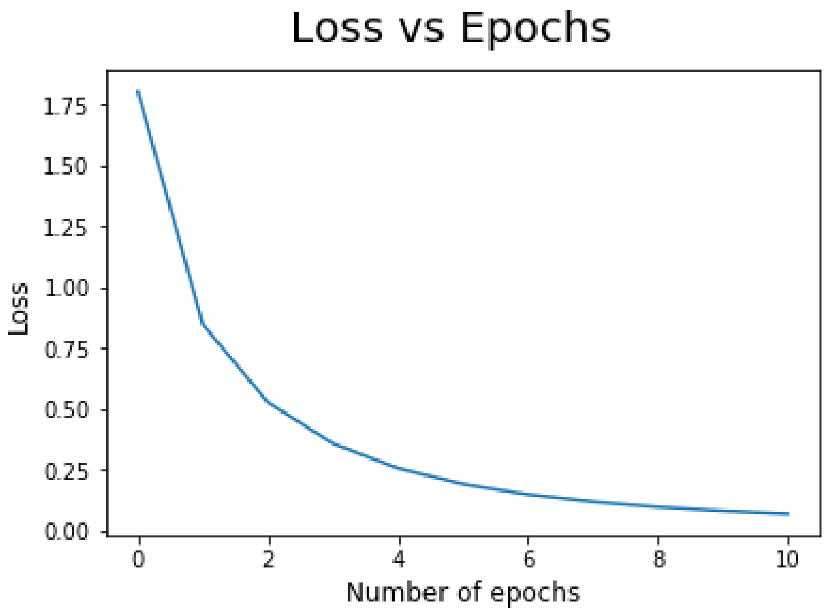

Figure 6: CLBB Data-Set : Loss vs Epochs

5.2.3 Inferences

Even for a very shallow neural network, we get an excellently trained model that gives sublime training and test accuracy. Through this model, we have seen the immense potential of Neural Networks.

References

[1] $https : //towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f$