CSE 574 - Introduction to Machine Learning Project 2 Report

Ajith Kumar Natarajan
UB Person number: 50318505
Fall 2019
`ajithkum@buffalo.edu`

Abstract

In this project I have implemented three different types of neural networks to study and observe the properties. The dataset used throughout the project is the *Fashion-MNIST* images. The three different types of implementations provide two different types of network, they are: Neural network with 1 hidden layer and Convolutional Neural Network (CNN) with 2 convolutional layers.

It is observed that the accuracy percentage increases with the introduction of convolutional layers. The results obtained can be summarised as follows:

- Plain neural network with one hidden layer, comprising of 128 nodes along with gradient descent optimization gives an accuracy output of about 52.39% (without using libraries& regularization) and 85% (with Keras) with training over 37 epochs
- An implementation consisting of 2 convolutional neural network layers gives an accuracy of 92% after training over 37 epochs

1 Introduction

Image classification is one of the widely used applications of modern machine learning and deep learning techniques. It has wide rage of applications varying from robotics and autonomous cars to support devices for visually challenged people.

In this work, a simple yet powerful classification method has been deployed using neural network and CNN. No regularization technique has been used. Cross entropy was used to evaluate the loss.

2 Dataset

The Fashion-MNIST is a dataset of Zalando's article images, consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Hence the number of features available to train the model is 784. This feature set consists of single pixel-value associated with each fashion apparel, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255. The classes are:

- T-shirt/top
- Trouser
- Pullover

- Dress
- Coat
- Sandal
- Shirt
- Sneaker
- Bag
- Ankle Boot

3 Pre-processing

There were two major pre-processing steps required before using the data to build the model. The **one-hot encoding** method is applied to the label dataset to convert from categorical nature to a a vector of length 10 for each sample in which the element corresponding to the label is set to 1 and all others are set to 0.

The second preprocessing that was required is the normalization of data. This is a common step in almost all the machine learning model building exercise. In order to avoid the effect of one particular column, having high values from inhibiting the other features and to prevent the issue of exploding gradients, all the columns have been normalised so that the value of the columns are in the range 0-1 consistently.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

4 Architecture

Since the input from the dataset consists of 784 features or attributes, the number of input nodes in the case was 784 as well. The weights were initialized randomly from a Gaussian distribution with mean 0 and variance 1. Shown below are the various architecture diagrams used.

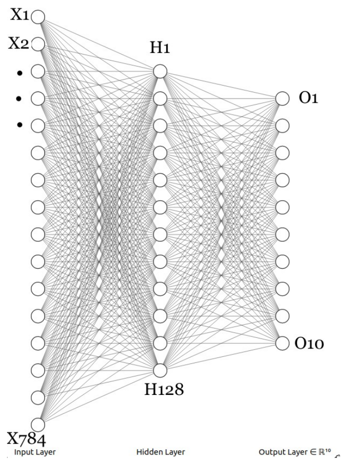


Figure 1: Architecture diagram for task 1

Task 1 and task 2 consisted of 1 hidden layer of nodes with the number of nodes in the hidden layer being 128. The input and output layers obviously had 784 and 10 nodes respectively.

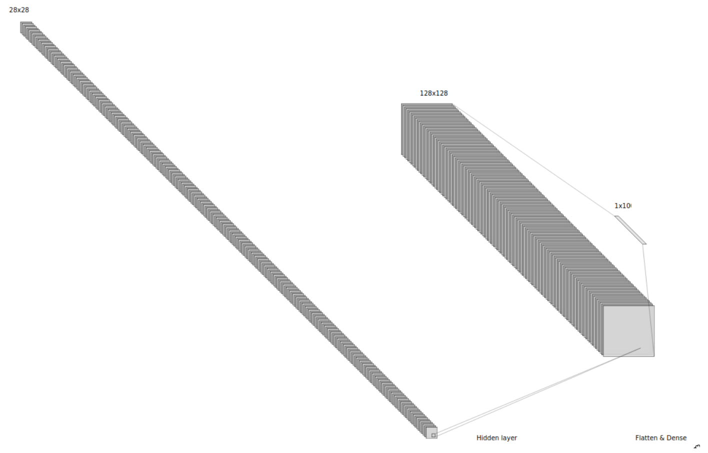


Figure 2: Architecture diagram for task 2

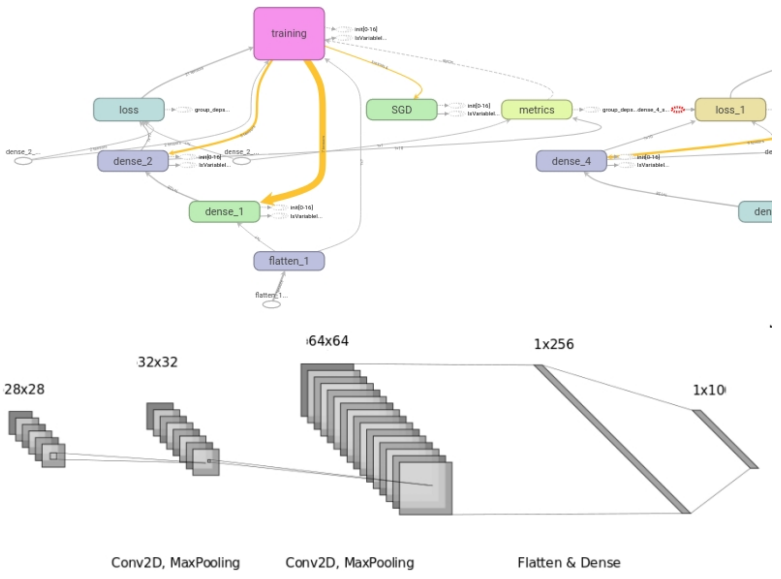


Figure 3: Architecture diagrams for task 3

Task 3, in which a CNN has been implemented consists of 2 CNN layers, having 32 and 64 filters respectively. A 25% dropout was implemented to build a robust model.

5 Results

This section consists of the results obtained by implementing the models described.

Graphs have been plotted to show the variation of loss (or cost) as a function of epochs. This is done to observe the convergence of the loss with increase in epochs.

For **task 1** and **task 2** it is inferred that from the graph that loss decreases with epoch. As the loss decreases, the accuracy increases. It can also be seen that the nature of the curve produced is smooth both while increasing in the case of accuracy and while decreasing in the case of loss. The accuracy obtained was 52.39% (without using libraries& regularization) and 85% (with Keras).

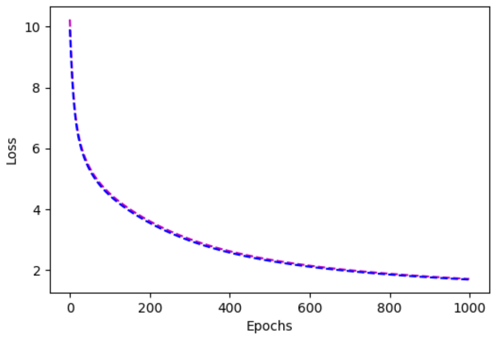


Figure 4: Accuracy vs epoch and loss vs epoch for training and validation data for task 1

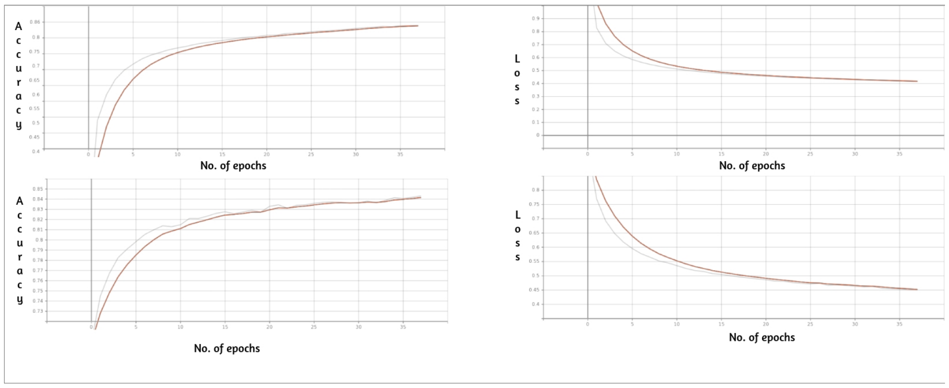


Figure 5: Accuracy vs epoch and loss vs epoch for training and validation data for task 2

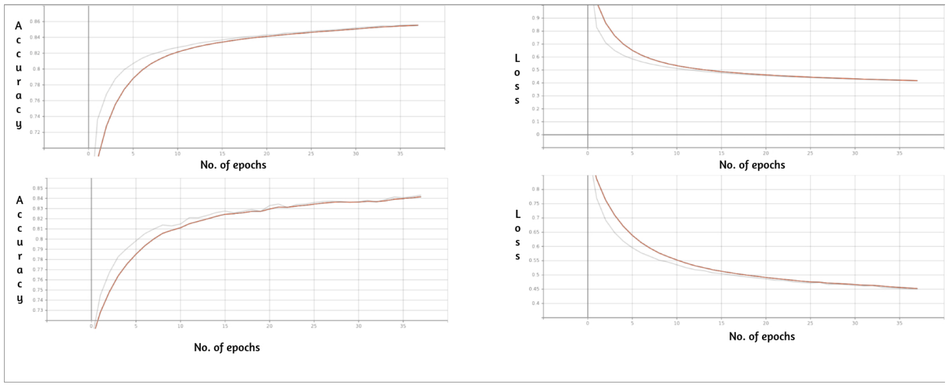
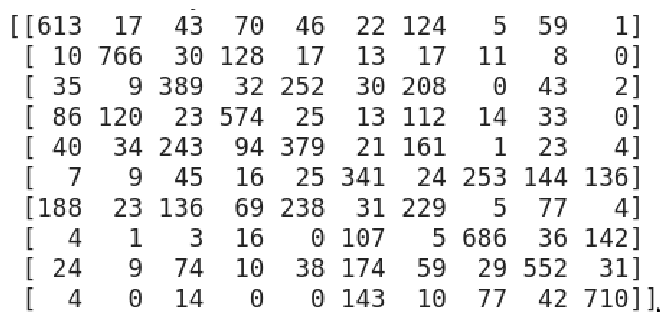


Figure 6: Accuracy vs epoch and loss vs epoch for training and validation data for task 3

While the accuracy with neural network came to about 85.6%, with the implementation of CNN the accuracy increased to 92%.

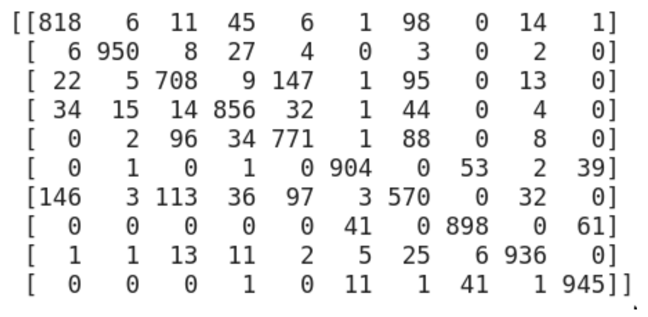
5.1 Confusion matrix

Here are the confusion matrices for all the three types of experiments:



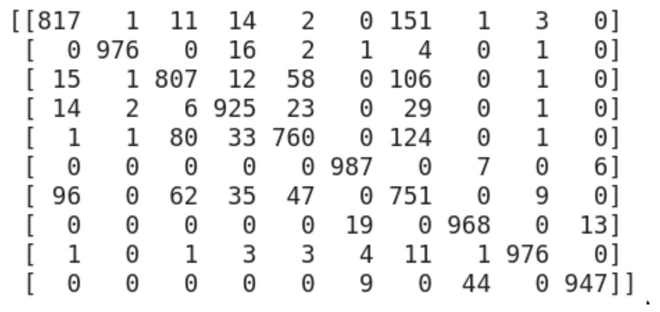
```
[[613  17  43  70  46  22 124   5  59   1]
 [ 10 766  30 128  17  13  17  11   8   0]
 [ 35   9 389  32 252  30 208   0  43   2]
 [ 86 120  23 574  25  13 112  14  33   0]
 [ 40  34 243  94 379  21 161   1  23   4]
 [  7   9  45  16  25 341  24 253 144 136]
 [188  23 136  69 238  31 229   5  77   4]
 [  4   1   3  16   0 107   5 686  36 142]
 [ 24   9  74  10  38 174  59  29 552  31]
 [  4   0  14   0   0 143  10  77  42 710]]
```

Figure 7: Confusion matrix for the implementation of neural network without high-level libraries



```
[[818   6  11  45   6   1  98   0  14   1]
 [  6 950   8  27   4   0   3   0   2   0]
 [ 22   5 708   9 147   1  95   0  13   0]
 [ 34  15  14 856  32   1  44   0   4   0]
 [  0   2  96  34 771   1  88   0   8   0]
 [  0   1   0   1   0 904   0  53   2  39]
 [146   3 113  36  97   3 570   0  32   0]
 [  0   0   0   0   0  41   0 898   0  61]
 [  1   1  13  11   2   5  25   6 936   0]
 [  0   0   0   1   0  11   1  41   1 945]]
```

Figure 8: Confusion matrix for the implementation of neural network with Keras



```
[[817   1  11  14   2   0 151   1   3   0]
 [  0 976   0  16   2   1   4   0   1   0]
 [ 15   1 807  12  58   0 106   0   1   0]
 [ 14   2   6 925  23   0  29   0   1   0]
 [  1   1  80  33 760   0 124   0   1   0]
 [  0   0   0   0   0 987   0   7   0   6]
 [ 96   0  62  35  47   0 751   0   9   0]
 [  0   0   0   0   0  19   0 968   0  13]
 [  1   0   1   3   3   4  11   1 976   0]
 [  0   0   0   0   0   9   0  44   0 947]]
```

Figure 9: Confusion matrix for the implementation of CNN

6 Conclusion

Through this work, it was observed that the CNN is a very powerful image classification method. The mathematical operation of convolution helps in extracting key features that would increase the accuracy of prediction. On the other hand, plain neural network is faster to train and hence it can be used in cases where the computational power available is not good enough for the implementation of CNN. Also multiple comparisons between different types of optimizers was done. It was also observed that implementation of mini-batch and batch normalization fastens the training process and acts as a regularizer to an extent as well.

References

[1] Bishop, Christopher M. Pattern Recognition and Machine Learning. New York :Springer, 2006

[2] Srihari, Sargur N. Lecture Slides for Machine Learning

[3] Stack Overflow

[4] Keras documentation

[5] Tensorflow documentation