PROJECT - 2

CLASSIFICATION USING NEURAL NETWORKS

Mohammed Mahaboob Khan
Department of Computer Science
University at Buffalo
Buffalo, NY 14214
mkhan32@buffalo.edu
PERSON #: 50318613

Abstract

There were three tasks to be performed in this project. The problem given to was a ten-class problem. The classification task was to recognize an image and classify it as one of ten classes by training the classifier using Fashion-MNIST dataset. In the first task I built a Neural Network with one hidden layer from scratch in Python. In the second task I built a multi-layer Neural Network with the open-source neural-network library, Keras. Finally, the third task was to build a Convolutional Neural Network with the open-source neural-network library, Keras.

1 Introduction

The study of Neural Networks is inspired by the observation that biological learning systems are built of very complex webs of interconnected neurons that communicate in order to process information.

Thus, Neural networks are a set of algorithms, modelled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labelling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

Neural networks help us cluster and classify. You can think of them as a clustering and classification layer on top of the data you store and manage. They help to group unlabelled data according to similarities among the example inputs, and they classify data when they have a labelled dataset to train on.

2 Dataset

The dataset in use is the Fashion - MNIST 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. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255.

The training and test data sets have 785 columns. The first column consists of the class labels (see above), and represents the article of clothing. The rest of the columns contain the pixel-values of the associated image.

- **Training data**:

The training data is a set of 60,000 examples from the Fashion – MNIST dataset. Thus, it is a 2D array of the dimensions (60000,785)

- **Testing data**:

The testing data is a set of 10,000 examples from the Fashion – MNIST dataset. Thus, it is a 2D array of the dimensions (10000,785)

3 Pre-processing

Before feeding the data to the model, we separate the label values from the training and testing data. Therefore, after pre-processing we have the following:

- Training data is split into X_train and Y_train, having the dimensions (60000,784) and (60000,1) respectively.
- Testing data is split into X_test and Y_test, having the dimensions (10000,784) and (10000,1) respectively.

4 Architecture

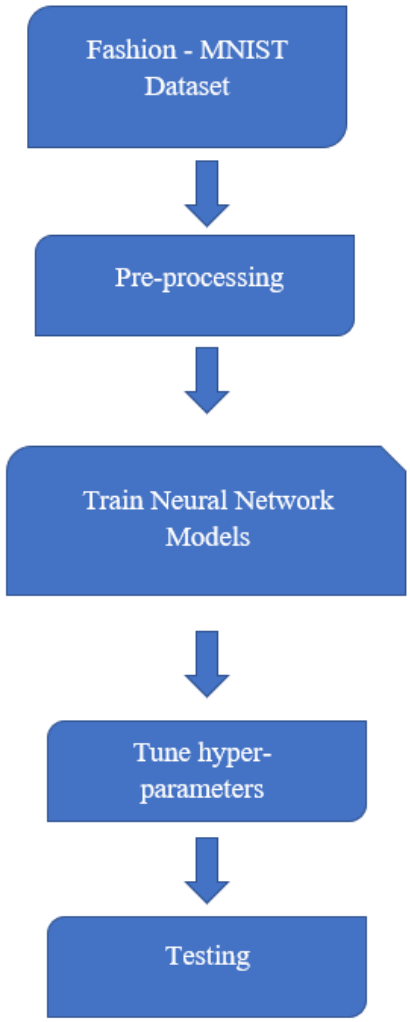


Figure 1 – Architecture

- **Neural Network:**

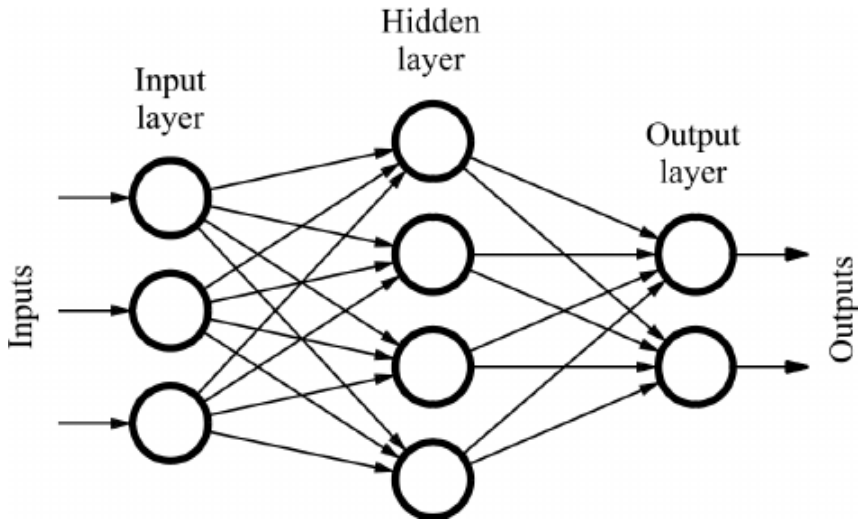


Figure 2 – Neural Network

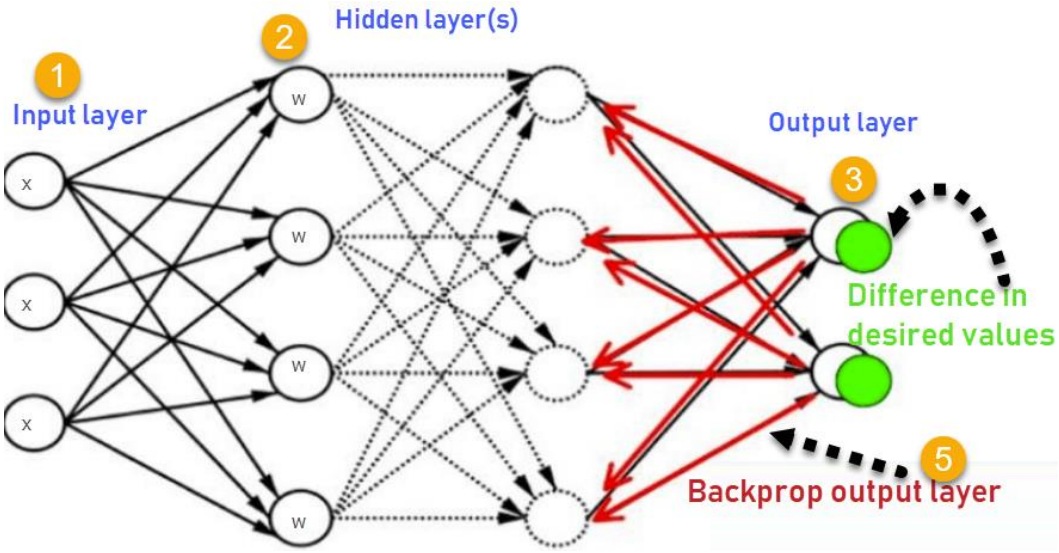


Figure 3 – Feed Forward and Backpropagation

5 Results

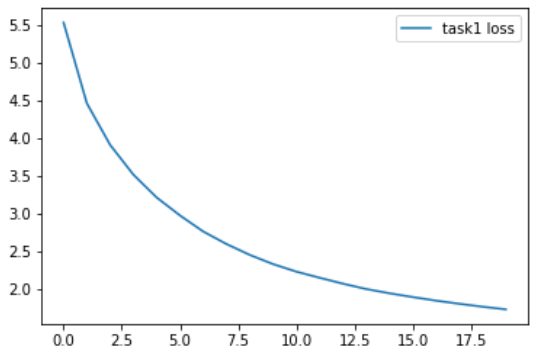
5.1 Task1

I tune my hyper-parameters and choose them to be,
Learning rate = 0.75
Epoch = 2000
Number of hidden nodes = 50

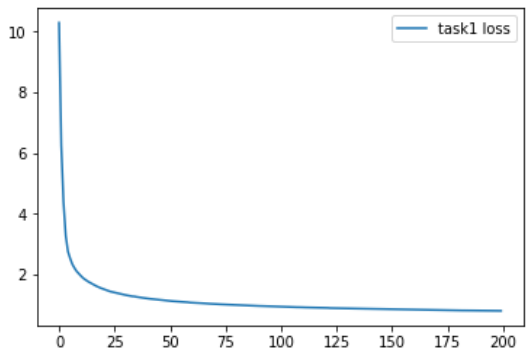
- **Choosing Epoch:**

Epoch is the hyper-parameter that defines the number times that the learning algorithm will work through the entire training dataset.

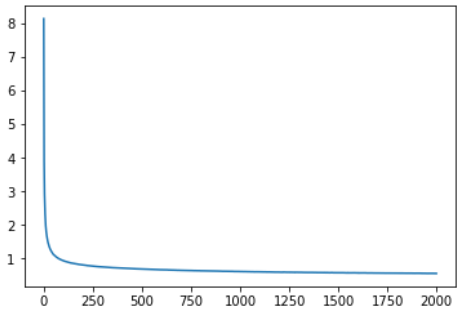
The screenshots below show different epoch settings.



Epoch = 20



Epoch = 200



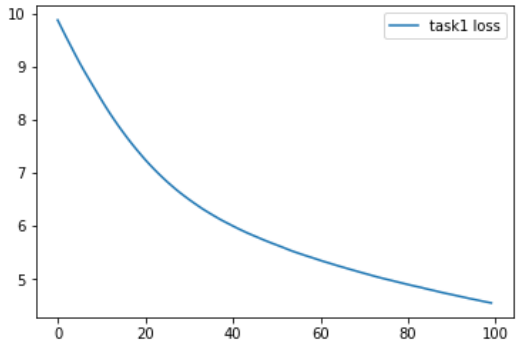
Epoch = 2000

Figure 4 – Choosing Epoch

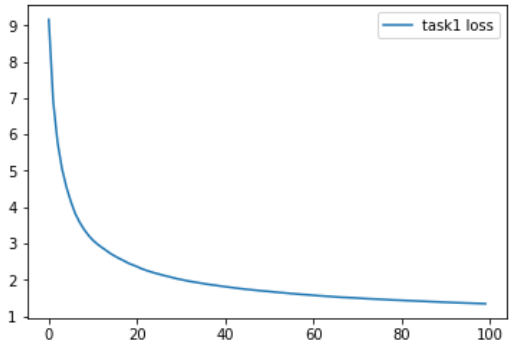
- **Choosing Learning Rate:**

Learning rate is a hyperparameter that controls how much we are adjusting the weights of our network with respect to the loss gradient. The lower the value, the slower we travel along the downward slope. While this might be a good idea in terms of making sure that we do not miss any local minima, it could also mean that we'll be taking a long time to converge.

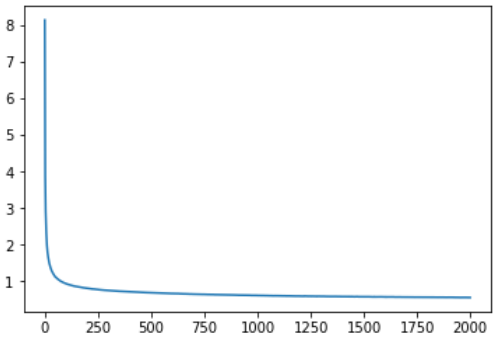
The screenshots below show different learning rate settings



Learning rate = 0.01



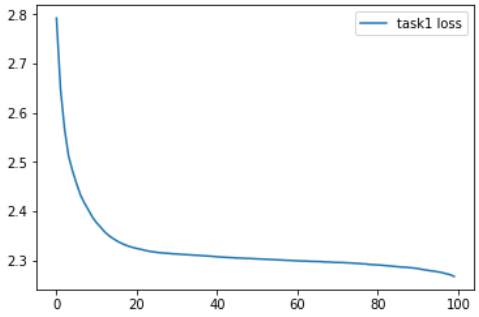
Learning rate = 0.3



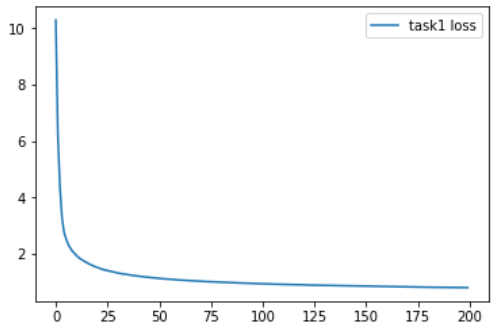
Learning rate = 0.75

Figure 5 – Choosing Learning Rate

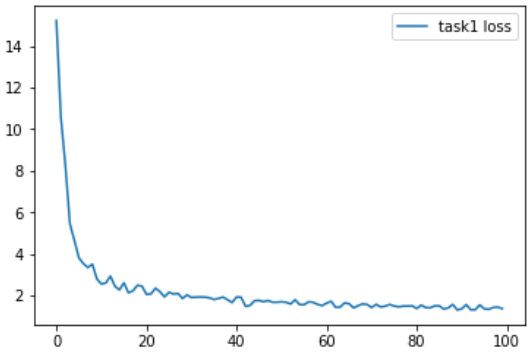
- **Hidden Nodes:**



Hidden Nodes = 3



Hidden Nodes = 50



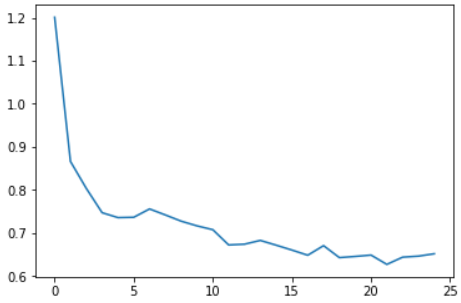
Hidden Nodes = 180

Figure 6 – Hidden Nodes

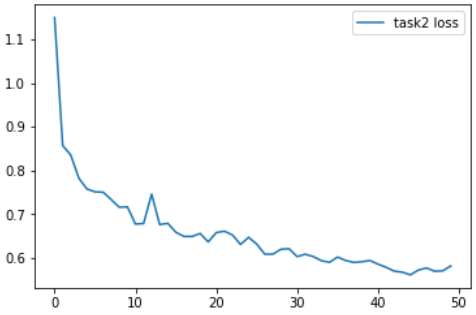
5.2 Task2

I tune my hyper-parameters and choose them to be,
Epoch = 50
Number of hidden layers = 3
I would like to mention that I have used the default optimizer '**adam**' since I found the model to perform better using this optimizer over the others.

- **Epoch:**

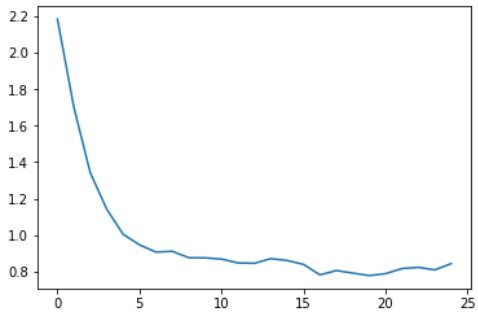


Epoch = 25

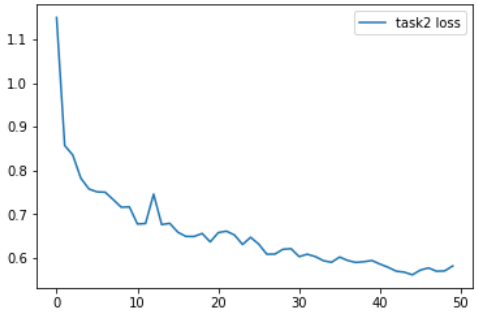


Epoch = 50

- **Number of hidden layers:**



Hidden Layers = 2



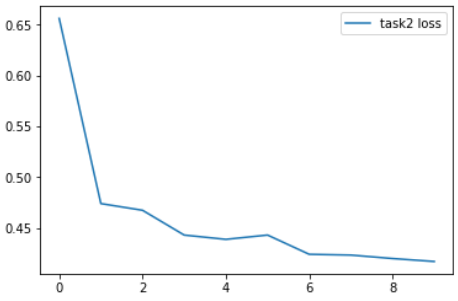
Hidden Layers = 3

Figure 7 – Task 2 hyperparameters tuning

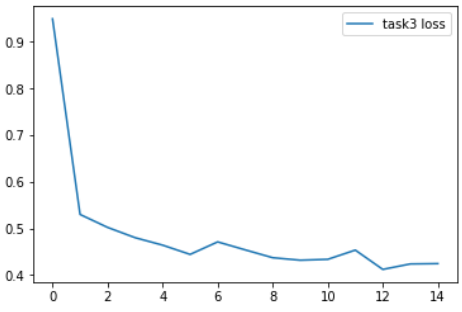
5.3 Task3

I tune my hyper-parameters and choose them to be,
Epoch = 15
Number of hidden layers = 2
Number of filters = 8

- **Epoch:**

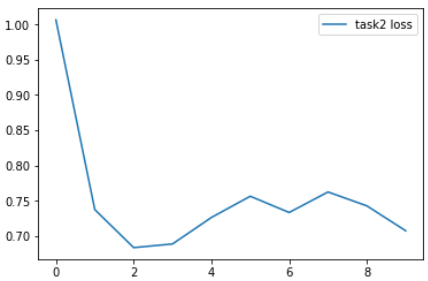


Epoch = 10

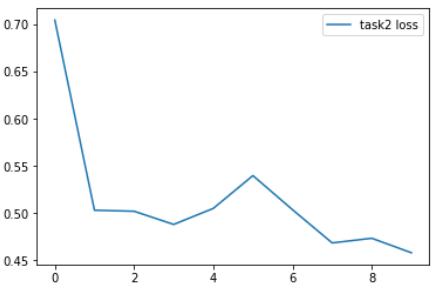


Epoch = 50

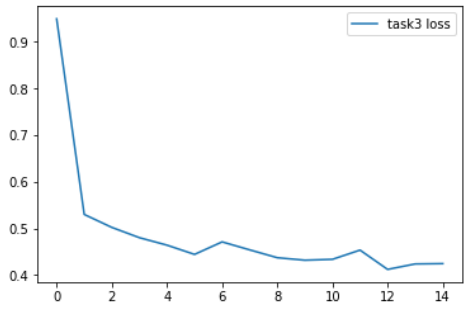
- **Number of filters:**



Filters = 64

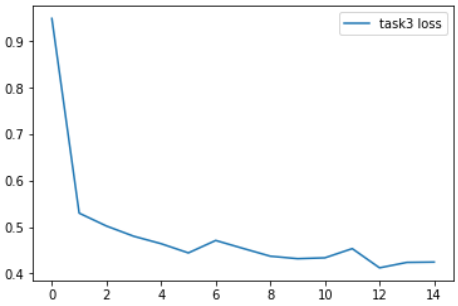


Filters = 15

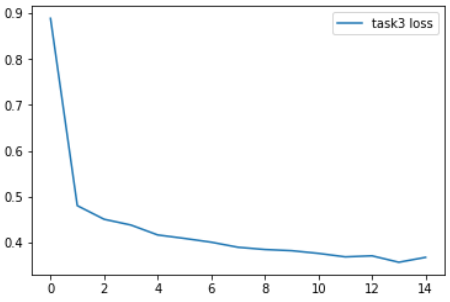


Filters = 8

- **Number of hidden layers:**



Hidden Layers = 1



Hidden Layers = 2

Figure 8 – Task 3 hyperparameters tuning

5.4 Performance

```
Loss:    0.5815476492641646

Accuracy:        77.07000000000001 %

Confusion matrix:
 [[744   9  24  71  18   5 105   2  21   1]
 [  5 923  20  39   6   0   3   1   3   0]
 [ 16   6 616   7 226   1 110   0  18   0]
 [ 51  24  14 790  62   2  46   0   8   3]
 [  1   1 105  33 768   0  85   0   7   0]
 [  2   0   0   2   1 792   2 103  22  76]
 [176   4 136  65 193   5 384   0  37   0]
 [  0   0   0   0   0  57   0 853   7  83]
 [  5   2  25   6   5  14  17   8 912   6]
 [  0   2   0   0   1  23   1  46   2 925]]
```

Figure 9 – Task 1 Results

```
Loss:    0.6062719611167908

Accuracy:        77.36999988555908 %

Confusion matrix:
 [[808   6  45  73   4   0  42   0  22   0]
 [  7 948  12  27   2   1   1   0   2   0]
 [ 11   3 712  12 100   0 152   0  10   0]
 [116  25  20 744  66   3  17   0   8   1]
 [  1   4 193  46 493   1 257   0   5   0]
 [  0   0   0   0   0 894   0  42   8  56]
 [243   2 150  50 114   0 405   0  36   0]
 [  0   0   0   0   0  52   0 919   1  28]
 [  2   2  30  14  16   7  13   3 913   0]
 [  0   0   0   0   0  34   0  64   1 901]]
```

Figure 10 – Task 2 Results

```
Loss:    0.386050368475914

Accuracy:        86.1 %

Confusion matrix:
 [[800   0  16  57   7   4 101   1  14   0]
 [  0 962   2  28   3   0   1   0   4   0]
 [ 18   0 766  12 101   1  95   0   7   0]
 [ 13   7   7 907  31   1  28   0   6   0]
 [  0   2  82  59 755   1  97   0   4   0]
 [  0   0   0   1   0 954   0  30   3  12]
 [122   1 100  46  94   1 611   0  25   0]
 [  0   0   0   0   0  22   0 950   1  27]
 [  1   2   2   5   3   5   8   3 970   1]
 [  0   0   0   0   0  18   0  47   0 935]]
```

Figure 11 – Task 3 Results

6 Conclusion

The end product of this project is a set classification models built using Neural Networks. The first model is a Neural Network with a single hidden layer that has an accuracy of 77.07%. The second model is a multi-layer Neural Network with 3 hidden layers that has an accuracy of 77.36%. Note that, although the accuracy of the 2 models is the same, the first model needed to be trained over 2000 epochs to achieve this. On the other hand, the second model achieved the said accuracy in just 25 epochs. Finally, the third model is a Convolutional Neural Network with 2 hidden layers that has an accuracy of 86.10%.

This project has helped me apply my existing knowledge of classification to practical use and gain deeper insights about classification problems. It has bolstered my understanding of Neural Networks. I hope to further tune the hyperparameters and test my models on other datasets in the future.

Acknowledgement

I am extremely grateful to Professor Sargur Srihari for explaining all the necessary concepts of Classification and Neural Networks required to complete this project. I would also like to thank Mihir Hemant Chauhan, the senior Teaching Assistant for clarifying my doubts and making concepts clear in his recitation.