Basic Of Deep Learning - Mid Semester Project Part 2

Stav Elizur Tommy Afek

Submitted as mid-semester project report for Basic Of Deep Learning course, Colman, 2023

1 Introduction

This project is a conclusion of all the skills we have acquired during our classes mainly relating to building an artificial neural network by solving a computing vision multi class classification problem. For this project the problem is a classification of 10 types of clothing. This project is built with Tensorflow Keras platform, which is a neural network python library that is mainly used for building a Neural Network in much easier and simpler way.

1.1 Data

We used Fashion MNIST data set that contains 70,000 gray scale images in 10 categories. 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 784 columns.

1.2 Problem

The problem of our project is a multi class classification of 10 different types of clothing images.

2 Solution

2.1 General approach

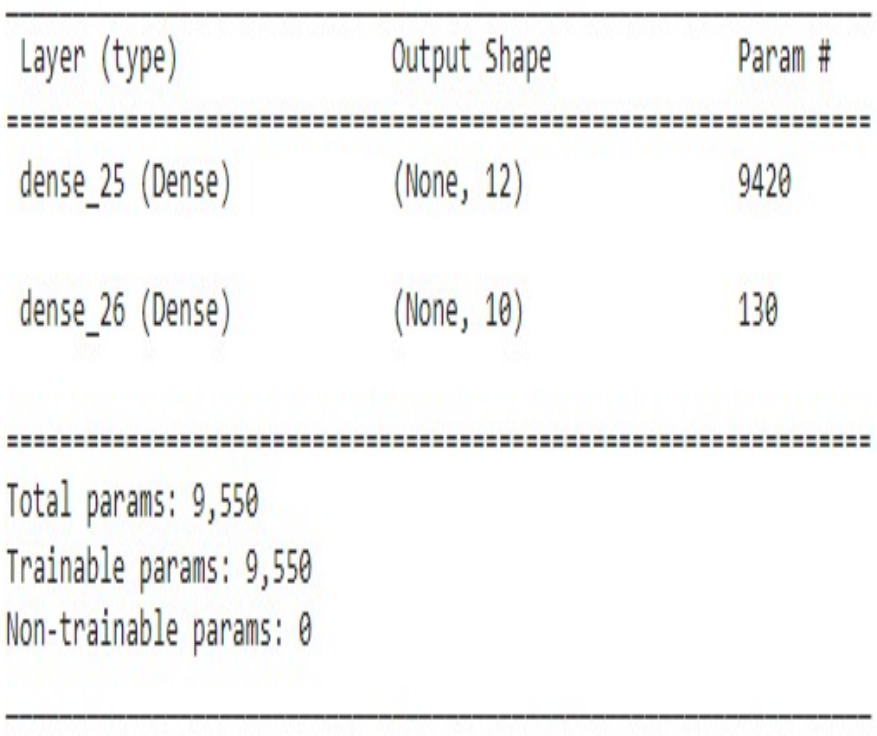
In order to solve our multi class images classification problem we built an artificial neural network. An artificial neural network is a Deep Learning algorithm that can take in an input image as array of pixels value.

2.2 Design

Our project code divides into 5 parts: Data pre-processing, Building the Base ANN for binary classification, Training the ANN, test our binary classification, write a base model for multi class classification, experiences with Performance evaluation and test our best model. In the Data pre-processing part we learn from the exploration how difficult is the data set, cause we need to be able to give a better approximation to how our ANN architecture is supposed to look like. Then, we load our Data set by using the Keras data sets which return split data set to train data and test data, then we scale the data with sklearn library to adjust our data to be able to use it in our artificial neural network. In the next part, we always load the best model in order to save run time and we skip the ANN building and training process because we get from those processes the same model we loaded. When we build the ANN for binary or multi class classification, We use Keras's tools for the building and training processes, which makes each stage of our building and training processes easier. We initialize our ANN network and use sigmoid activation function which is used in our hidden layer and the purpose of this function to convert and normalize a number to a number between 0 and 1. The final output layer with the Softmax activation function for multi class classification and sigmoid activation function for binary classification for the prediction. Softmax activation function is used to give a probability for each class prediction. Then we compile our ANN with the Adam optimizer, which is the best variation of the Gradient Descent algorithm because it has the smallest slope between the rest of the algorithms so it gives the best results for loss per epochs. We use the loss categorical cross entropy function for multi class classification and binary cross entropy function for binary classification which does severity punishment for our model depending on the distance between the predicted value and the actual value. Also we defined that the accuracy metrics between each epochs will be calculated and saved to make sure our model is not over fitted. Then, we train our ANN with the training set, test our model with the test set and after that we save it. The training process took 40 seconds. In the last part, after our ANN is built, we evaluate our performances using the accuracy metrics our model has to calculate our the validation and test loss and accuracy in each epoch and we create the confusion matrix for our experiments to define which experiment has better results. The technical challenges were understanding how many layers we should use for building the ANN classification which we improve by experimenting values.

2.3 Base Model

Our base model for binary classification and multi class classification is an artificial neuron network that is built from 1 input layer of 784 neurons + 1 constant bias neuron, 1 hidden layer that has 12 neurons and 1 bias neuron, and 1 output layer that has the output neuron. Our base models gets an image of 28x28 being reshaped to a vector of 784 pixels and it predict value of the input.



ANN architecture of the base model

2.4 First Experiment

In our first experiment, we paid more attention to our ANN architecture. Then, in the fully connected layers we had 1 more fully connected layer than our base model and also a different number of neurons per fully connected layer than our base model. The order of the layers is: a fully connected layer with 120 neurons, a fully connected layer with 60 neurons, a fully connected layer with 30 neurons and an output layer with 10 neurons. The number of epochs we used to train our neural network was 10, the each batch size of our data was 32 and Learning Rate of 0.001, same as the base model hyper parameters.

Layer (type)	Output Shape	Param #
dense_27 (Dense)	(None, 12)	9420
dense_28 (Dense)	(None, 120)	1560
dense_29 (Dense)	(None, 60)	7260
dense_30 (Dense)	(None, 30)	1830
dense_31 (Dense)	(None, 10)	310

Total params: 20,380
Trainable params: 20,380
Non-trainable params: 0

Architecture of the first experiment model

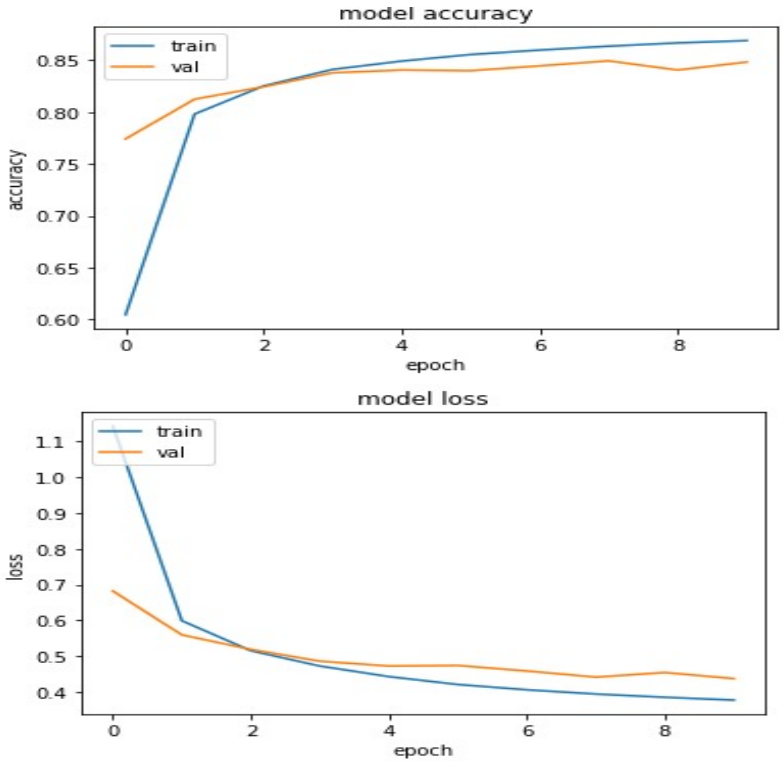


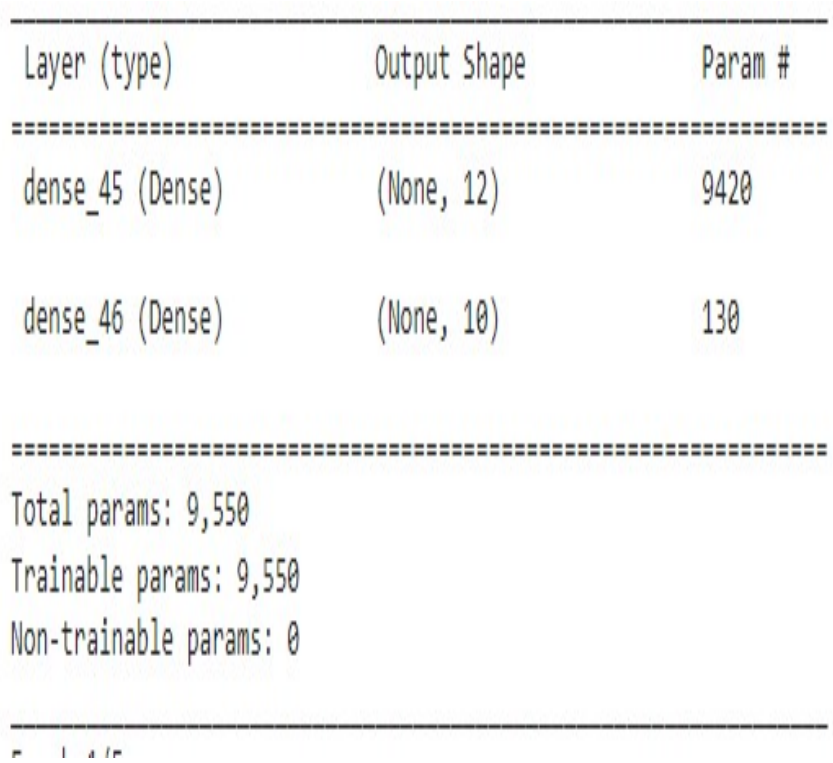
Image of the first experiment loss and accuracy per epoch

```
[[817   0  14  69   2   1  87   0  10   0]
 [  0 954   9  31   3   0   1   0   2   0]
 [ 17   3 800  12  70   0  94   0   4   0]
 [ 31  10   8 914  18   1  11   2   5   0]
 [  1   0 162  63 688   1  79   0   6   0]
 [  0   1   0   0   0 921   0  52   6  20]
 [166   0 123  65  65   0 568   0  13   0]
 [  0   0   0   0   0  22   0 946   0  32]
 [  5   1  12  12   7  19  14   4 925   1]
 [  1   0   0   1   0  39   1  40   0 918]]
              precision    recall  f1-score   support

 T-shirt/top       0.79      0.82      0.80      1000
     Trouser       0.98      0.95      0.97      1000
    Pullover       0.71      0.80      0.75      1000
       Dress       0.78      0.91      0.84      1000
        Coat       0.81      0.69      0.74      1000
      Sandal       0.92      0.92      0.92      1000
       Shirt       0.66      0.57      0.61      1000
     Sneaker       0.91      0.95      0.93      1000
         Bag       0.95      0.93      0.94      1000
  Ankle boot       0.95      0.92      0.93      1000

    accuracy                           0.85     10000
   macro avg       0.85      0.85      0.84     10000
weighted avg       0.85      0.85      0.84     10000

Accuracy: 85%
```

Image of the first experiment confusion matrix

2.5 Second Experiment

In our second experiment, we focused more in our hyper-parameters and used different numbers than the base model. We had the exact same architecture as the base model ANN, using different hyper-parameters which are the batch size of 40, learning rate of 0.001 and 5 number of Epochs.



Layer (type)	Output Shape	Param #
dense_45 (Dense)	(None, 12)	9420
dense_46 (Dense)	(None, 10)	130

Total params: 9,550
Trainable params: 9,550
Non-trainable params: 0

Architecture of the second experiment model

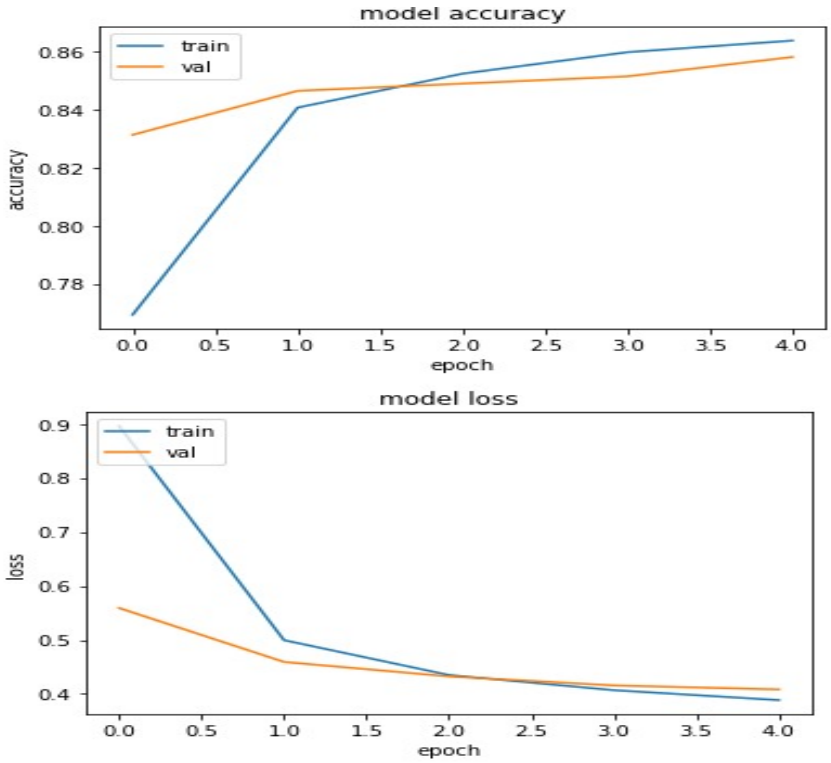
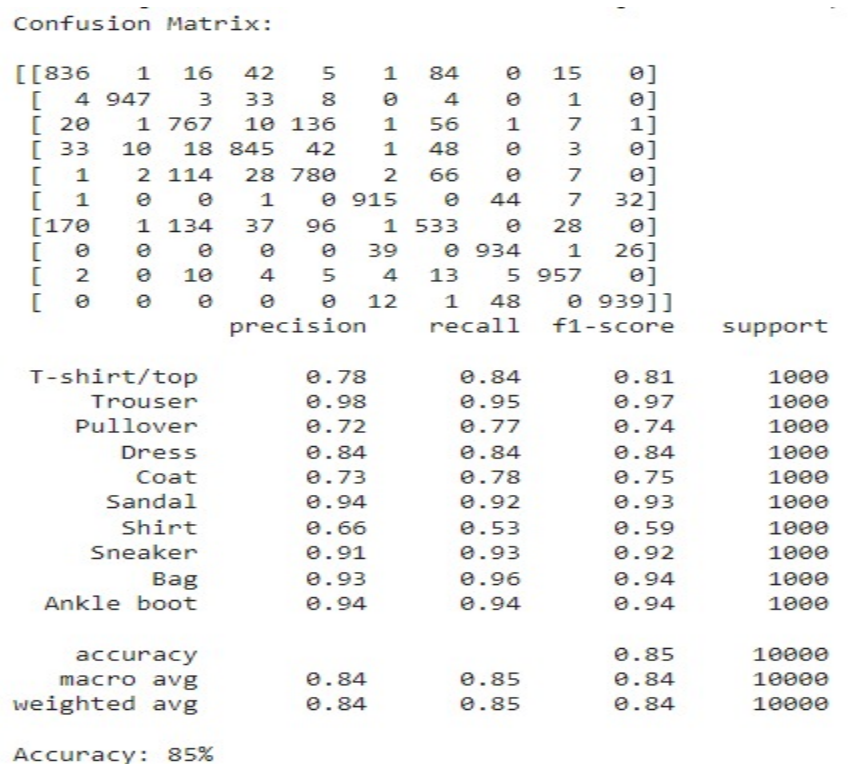


Image of the second experiment loss and accuracy per epoch



```
Confusion Matrix:

[[836   1  16  42   5   1  84   0  15   0]
 [  4 947   3  33   8   0   4   0   1   0]
 [ 20   1 767  10 136   1  56   1   7   1]
 [ 33  10  18 845  42   1  48   0   3   0]
 [  1   2 114  28 780   2  66   0   7   0]
 [  1   0   0   1   0 915   0  44   7  32]
 [170   1 134  37  96   1 533   0  28   0]
 [  0   0   0   0   0  39   0 934   1  26]
 [  2   0  10   4   5   4  13   5 957   0]
 [  0   0   0   0   0  12   1  48   0 939]]
              precision    recall  f1-score   support

 T-shirt/top       0.78      0.84      0.81      1000
     Trouser       0.98      0.95      0.97      1000
    Pullover       0.72      0.77      0.74      1000
       Dress       0.84      0.84      0.84      1000
        Coat       0.73      0.78      0.75      1000
      Sandal       0.94      0.92      0.93      1000
       Shirt       0.66      0.53      0.59      1000
     Sneaker       0.91      0.93      0.92      1000
         Bag       0.93      0.96      0.94      1000
  Ankle boot       0.94      0.94      0.94      1000

    accuracy                           0.85     10000
   macro avg       0.84      0.85      0.84     10000
weighted avg       0.84      0.85      0.84     10000

Accuracy: 85%
```

Image of the second experiment confusion matrix

2.6 Best model Results and Metrics

After we ran the 2 experiments, we created a combination of them with our base model. We saw that our base model architecture, which had 1 hidden layer with 12 neurons + 1 bias, had 84 percent accuracy. we inferred that the numbers of layers should be bigger. Because of that, we knew that we had to focus more in our architecture and modify them. Then, we ran an experiment with input layer of 784 neurons + bias, a hidden layer of 128 + 1 bias neurons, a hidden layer of 64 + bias neurons,a hidden layer of 64 + 1 bias neurons, a hidden layer of 32 + 1 bias neurons, a hidden layer with 16 + 1 bias neurons and an output layer of 10 neurons as the number of our multi classes. The number of epochs was 15, learning rate was 0.001, and 40 batches. The model had 87 percent accuracy, which was the best had created.

Layer (type)	Output Shape	Param #
dense_97 (Dense)	(None, 128)	100480
dense_98 (Dense)	(None, 64)	8256
dense_99 (Dense)	(None, 32)	2080
dense_100 (Dense)	(None, 16)	528
dense_101 (Dense)	(None, 10)	170

Total params: 111,514
Trainable params: 111,514
Non-trainable params: 0

Architecture of the best model

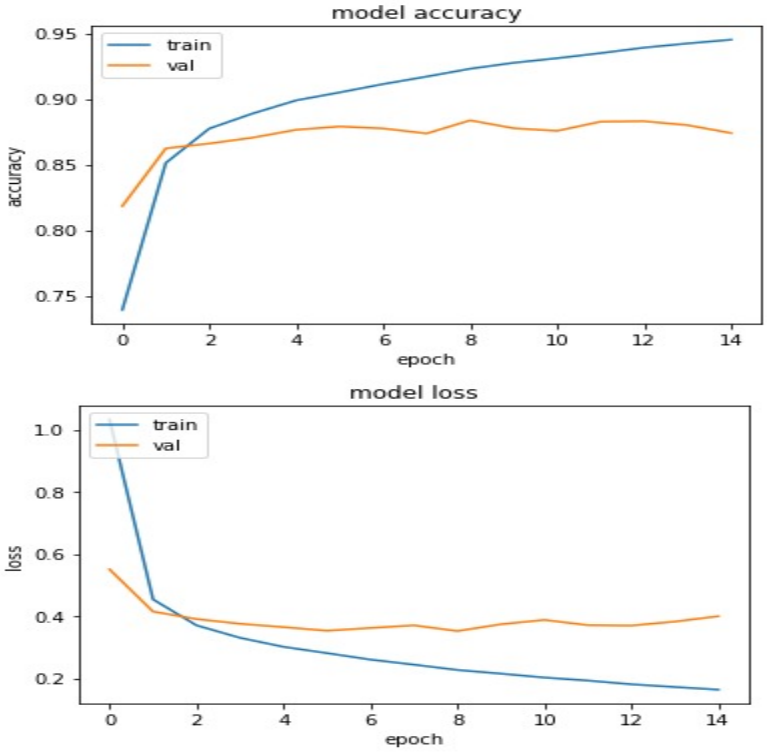


Image of the best model loss and accuracy per epoch

```
Confusion Matrix:

[[847   3  18  17   3   0 104   0   8   0]
 [  7 960   3  22   5   0   3   0   0   0]
 [ 16   0 835   5  66   2  76   0   0   0]
 [ 41   6   9 858  41   1  38   0   5   1]
 [  0   1 135  17 791   1  53   0   2   0]
 [  0   0   0   0   0 950   1  31   1  17]
 [142   3  99  18  84   0 645   0   9   0]
 [  0   0   0   0   0  16   0 973   0  11]
 [  6   1   8   3   4   5  19   4 950   0]
 [  1   0   0   0   0  25   1  56   1 916]]
              precision    recall  f1-score   support

 T-shirt/top       0.80      0.85      0.82      1000
     Trouser       0.99      0.96      0.97      1000
    Pullover       0.75      0.83      0.79      1000
       Dress       0.91      0.86      0.88      1000
        Coat       0.80      0.79      0.79      1000
      Sandal       0.95      0.95      0.95      1000
       Shirt       0.69      0.65      0.66      1000
     Sneaker       0.91      0.97      0.94      1000
         Bag       0.97      0.95      0.96      1000
  Ankle boot       0.97      0.92      0.94      1000

    accuracy                           0.87     10000
   macro avg       0.87      0.87      0.87     10000
weighted avg       0.87      0.87      0.87     10000

Accuracy: 87%
```

Image of the best model confusion matrix

3 Discussion

In summary, after we ran the experiments we understood from the first experiment which was more focused in the ANN architecture itself. Also we learned that having a small change in the number of epochs or the number of hidden neurons can make a great impact on the model itself.. After we ran the second experiment which was more focused on the hyper parameters, we learned that having more epochs should cause over fitting, and the learning rate is dependable on the epochs because when we have a small learning rate, getting to the minimal loss will take a lot of epochs and we needed to find the balance between them. We also learned that the batch size controls the accuracy of the estimate of the error gradient when training neural networks.

4 Code

Colab project