Bank Note Authentication Report

GitHub link: Bank Note Authentication

Problem Statement:

The banknote dataset involves predicting whether a given banknote is authentic given a number of measures taken from a photograph.

Data Set Information:

Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images.

The dataset contains 1,372 rows with 5 numeric variables. It is a classification problem with two classes (binary classification).

Attribute Information:

1. variance of Wavelet Transformed image (continuous)
2. skewness of Wavelet Transformed image (continuous)
3. curtosis of Wavelet Transformed image (continuous)
4. entropy of image (continuous)
5. class (integer)

Neural Network:

Given that the dataset is small, a small batch size is probably a good idea, e.g. 16 or 32 rows. Using the Adam version of stochastic gradient descent is a good idea when getting started as it will automatically adapt the learning rate and works well on most datasets.

We define a minimal MLP model here. In this case, we will use one hidden layer with 10 nodes and one output layer (chosen arbitrarily). We will use the

ReLU activation function in the hidden layer and the "*he_normal*" weight initialization.

The output of the model is a sigmoid activation for binary classification and we will minimize binary cross-entropy loss.

The model summary is shown below. This includes the number of parameters(weights and biases) and the hyperparameters(number of hidden layers, number of neurons in each layers etc).

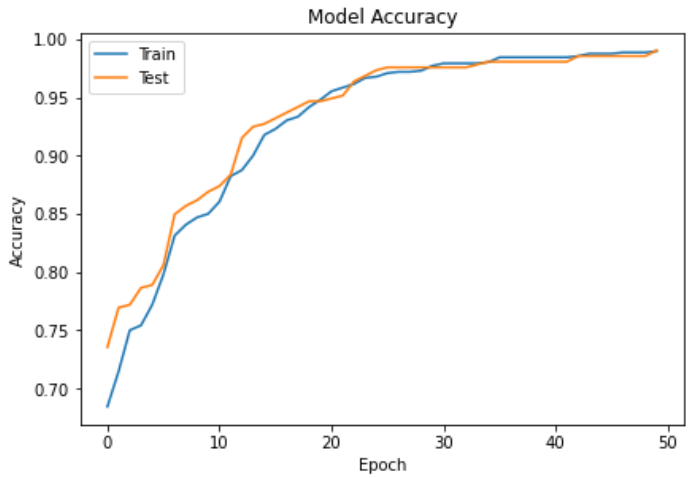
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 10)                50
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 11
=================================================================
Total params: 61
Trainable params: 61
Non-trainable params: 0
```

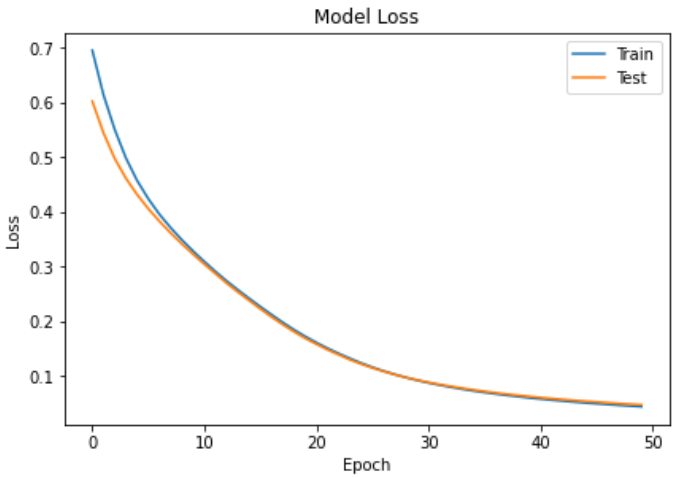
Now we will fit the model for 50 training epochs with a batch size of 32 because it is a small dataset.

At the end of training, we evaluate the model's performance on the test dataset and get the accuracy of the model as well as the F1 score.

We achieve an accuracy of 99% and F1 score of 99%.

Finally, we plot learning curves of the cross-entropy loss on the train and test sets during training.





We can see that the model appears to converge well and does not show any signs of overfitting or underfitting.

The weights and biases of the layers are as follows:

```
First layer weights:
 [[-1.3706262   1.4476063   0.25271323 -1.1387359  -0.7349581   1.5193254
  -1.126378   -1.9485296   1.3475441  -0.06308684]
 [-0.5722223  -0.65950054  0.6096384  -1.048388   -0.73161393  1.1307217
  -1.4979749  -1.1856549  -0.52967536  1.8283366 ]
 [-1.3921794   0.52446866  0.44071794 -1.8586632  -0.80698323  1.2622175
  -0.36691308  0.64747185 -0.26380765  1.2084091 ]
 [-0.88719165 -0.87527806  0.27265665  0.56715673  0.01852613  0.1783846
  -0.4744514  -0.02313008 -1.1457193  -0.67862797]]
First layer biases:
 [ 0.04697642  0.65690607 -0.3040594   0.36812392  0.24776587  0.89643127
  0.3471015  -0.07937246 -0.29635066  0.39331102]
--------------------------------------------------------------------------------
Second layer weights:
 [[ 0.13348322]
 [-1.0306451 ]
 [ 0.03711682]
 [ 0.9791785 ]
 [ 0.82784194]
 [-1.3336751 ]
 [ 0.6688549 ]
 [ 0.429432  ]
 [ 0.29093134]
 [-0.44755286]]
Second layer biases:
 [-0.44964656]
```

Our model has achieved a great accuracy of 99% percent. This might suggest that the prediction problem is easy and/or that neural networks are a good fit for the problem.