Title: Bank Note Authentication - https://github.com/daveylu/10315-project

Group Members: Alina Chen, Dave Lu, Jin Rong Song

Introduction:

Around the world, government banks use banknotes– a promissory note/bill that one party can use to pay another– as a standard form of currency (Kagan). However, many counterfeit banknotes are produced without legal sanction, which can harm the lives of individual citizens and the economy overall. In our project, we aim to produce a classifier that can distinguish between real and counterfeit bank notes as accurately as possible. Specifically, we are going to use two different approaches (Random Forest and Neural Nets) and compare which one results in a more exact classifier. We will use the following data set of genuine and forged banknote specimens from the UCI ML repository: UCI Machine Learning Repository: banknote authentication Data Set (Dua). In this set of 1372 data instances, the Wavelet Transform tool was used to extract features from 400x400 pixel images. Our inputs are four continuous attributes of a banknote specimen extracted using the Wavelet transforms: the variance, the skewness, the curtosis, and the entropy of the transformed image. Our output is an integer (0 or 1) classifying whether the specimen is counterfeit or not. In this experiment, we seek to answer the following question: How does the performance accuracy of the Random Forest Algorithm compare to that of an optimized Neural Network in terms of binary classification on this banknote authentication data set?

Methods/Models:

In our project, we used two different methods – neural networks and random forest classifiers – and compared which one had a better performance at classifying the bank notes into genuine versus counterfeit. On each of the models, we focused on accuracy/cross-validation entropy loss for our performance measure. In particular, we also plan on comparing false positives and false negatives metrics.

For the neural network, similarly to what we've learned from class, we used a sigmoid activation function at the end to classify the bank note and evaluated the loss using a binary cross entropy objective function. Overall, we found what architecture would produce the neural network with the best performance; we determined our final neural network to have four hidden layers, using ReLU, Sigmoid, leaky ReLU, and then Sigmoid activation functions, respectively. The initial layer took an input of 4 dimensions (for each of the four features of our data instances) and expanded it to 8. The following layers expanded them to a dimension of size 12, and the final layer outputted a one-dimensional binary output for our final answer.

```
class NeuralNet(nn.Module):

    def __init__(self):
        super(NeuralNet, self).__init__()

        self.hidden_layer_1 = nn.Linear(in_features = input_dim, out_features = input_dim * 2)
        self.relu = nn.ReLU()
        self.hidden_layer_2 = nn.Linear(in_features = input_dim * 2, out_features = input_dim * 3)
        self.sigmoid = nn.Sigmoid()
        self.hidden_layer_3 = nn.Linear(in_features = input_dim * 3, out_features = input_dim * 3)
        self.leaky_relu = nn.LeakyReLU()
        self.hidden_layer_4 = nn.Linear(in_features = input_dim * 3, out_features = 1)
        self.final_activation = nn.Sigmoid()
```

For the random forest classifier, we determined to use a depth limit of 4 (due to data only having four features) on a base collection of 100 decision trees. Having the appropriate

amount of decision trees ensures that the computational toll of training the model is not too great, and restricting the depth limit prevents overfitting the training data.

```
N = X_test.shape[0]

if(train_models == True):
    clf = RandomForestClassifier()              # initialize random forest classifier object
    clf.fit(X_train, y_train)                   # fit the classifier to our training dataset

    if(save_models == True):
        joblib.dump(clf, "random_forest.joblib") # saves the random forest classifier to a file

else:
    clf = joblib.load("random_forest.joblib") # reads the random forest classifier from a file

y_pred_proba = clf.predict_proba(X_test)    # predict the probabilities of the samples in the test dataset being real or counterfeit
y_pred = clf.predict(X_test)                # predict the labels of the samples: real or counterfeit
loss = log_loss(y_test, y_pred_proba)       # compute cross entropy loss using our predicted probabilities and true labels.
accuracy = np.sum(y_test == y_pred) / N     # compute accuracy of our labeling
```

Results:

After training the random forest model on 80% of the data and testing on 20%, our final performance had the following validation results and confusion matrix:

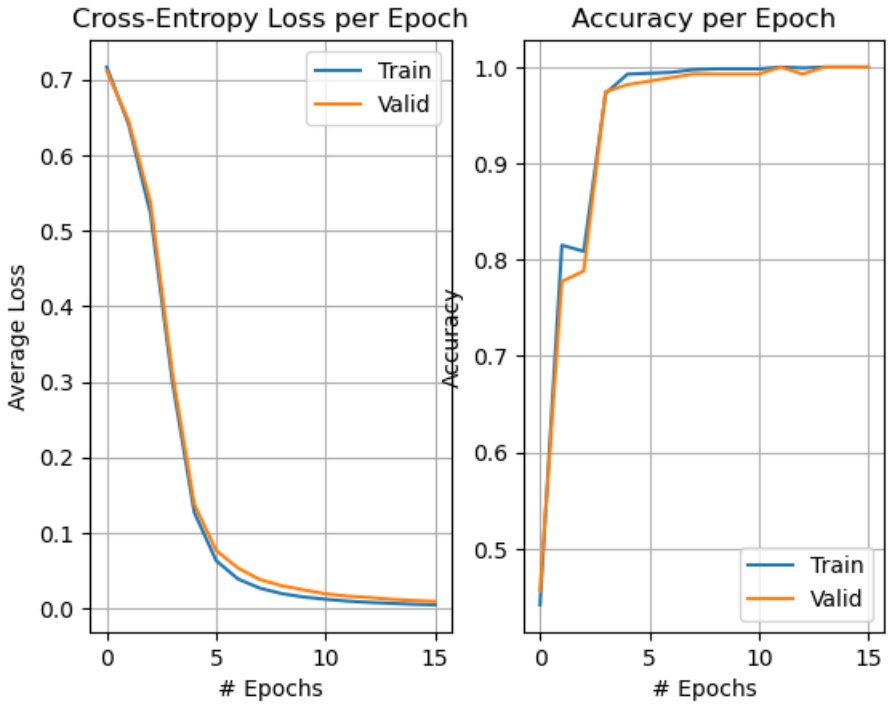
RF Validation Cross Entropy Loss	0. 0369
RF Validation Accuracy	0. 9891

	Authentic	**Counterfeit**
Authentic (Prediction)	144	1
Counterfeit (Prediction)	2	128

Similarly after training our neural net on 80% of the data and testing on 20%, our final performance had the following validation results:

NN Validation Cross Entropy Loss	0. 0094
NN Validation Accuracy	1. 0

Additionally, below are the graphs of the performance accuracy at each epoch for the neural network.



Discussion and Analysis:

Between the two methods we analyzed, our four-hidden-layer neural network not only had a smaller validation cross entropy loss and better accuracy than the random forest classifier, but also had perfect performance in terms of accuracy, obtaining a rate of 100%. This latter point was somewhat surprising since obtaining a perfect accuracy rate does not seem practical, however it may be due to the limited amount of data that we had. A further extensional experiment could be performed to test the validity of our two models if there were more data. Yet, it is clear that in terms of our given data, the neural network is the best model we could have generated. From the confusion matrix, we see that while the Random Forest has an incredibly high accuracy rate, it is still prone to producing false negatives and lowering accuracy. In general, it was fairly expected to see a high accuracy rate on both

models. Specifically, in other studies, random forest tends to outperform most other models such as SVM and logistic regression (Kale), so it is reasonable to also see it with such a high accuracy on the bank note data set.

Works Cited

Dua, Da and Graff, C. “banknote authentication Data Set.” *UCI Center for Machine Learning and Intelligent Systems*, 16 April 2013, archive.ics.uci.edu/ml/datasets/banknote+authentication#. Accessed 15 April 2023.

Kagan, Julia. “What Are Banknotes and How Are They Used?” *Investopedia*, 30 Oct. 2020, www.investopedia.com/terms/b/banknote.asp. Accessed 15 April 2023.

Kale, Hrishikesh. “BankNote Authentication using Machine Learning Algorithms.” *VSH Solutions*, 5 Feb. 2019, www.vshsolutions.com/blogs/banknote-authentication-using-machine-learning-algorithms/. Accessed 27 April 2023.

Virtue, Pat. “Programming 7: Training and Fine-Tuning Image Classifiers with PyTorch.” Introduction to Machine Learning, 1 April 2023, Carnegie Mellon University.