Homework #4

Matthew Favela
Tiffany Le

Introduction

The problem that this data analysis aims to solve is predicting which number an image is of. The possible digits in the picture are 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9. Each image, made of 784 pixels, can only contain one digit. To predict the number, we treat each pixel in the image as a feature. This data analysis will be completed through Logistic Regression and Deep Neural Networks, and a better performance model will be utilized to finalize this analysis.

Through the success of this model, we can further the complexity and reach of feature recognition in images. Additionally, for future images, we are able to predict the digit an image is of.

Methods

The data analysis of predicting the number an image is of will be completed by two different predicting algorithms: “Logistic Regression” and “Deep Neural Networks.”

Data Preparation

Before fitting the data to the predictive models, the training data, from the loaded dataset, needs to be reshaped into a 2D array, where each row represents a flattened image and the columns represent the features (pixels), instead of the image matrix that the data came from. Following this, data needs to be rescaled within the range [0, 1], instead of the original range of [0, 255]. To train the models, we need to “One-Hot Encode” the labels (digits 0 to 9) using LabelBinarizer.

Logistic Regression

To develop our initial model, we begin with the initialization of a blank model. Following this, we construct an empty pipeline and integrate our model into it. Subsequently, we train the model, execute predictions, and evaluate the likelihood that a number falls into a specific category. Finally, we evaluate the model's performance by computing its accuracy score and analyzing the results through a confusion matrix.

Deep Neural Network

To construct our DNN model, we set up a sequential model featuring an initial input dimension of 784. This is followed by five progressively denser hidden layers: 128, 64, 32, 16, and 8 nodes, respectively. The model culminates in an output layer with 10 nodes, corresponding to the ten potential numeric digits. Before initiating the training process, we designated "Categorical Crossentropy" as our loss function and chose "Stochastic Gradient Descent" as the optimizer to enhance accuracy. Following this, the model is trained using the training data for 100 epochs, and the validation data (testX and test Ynn) was used to monitor the model's performance. Finally, the trained model's performance is evaluated on the test data.

Results

How do your models perform (discuss accuracy, overfitting for both train/test)? Is one model better than the other? Are you surprised by which one did better?

After conducting the performance analysis on the two models, the models' accuracy scores were relatively similar; we can see the Accuracy Scores of each model below in Figure #1. Additionally, we can visualize each model's predictions through the confusion matrices in Figures #2-5. While the scores were relatively the same, the logistic regression model performed better than the deep neural network model (very slightly) for both the training and testing (validation) sets. Additionally, there doesn't appear to be any form of overfitting for either model due to the insignificant difference between the training and testing accuracy for both models. That being said, due to their complexities, deep neural networks have a higher tendency to overfit the training data. After comparing the results, we are relatively surprised that the performance of both models was really similar to one another. Given the complexity of DNN, we expected that a DNN would perform better than the Logistic Regression model.

	Logistic Regression	**Deep Neural Network**
Training Accuracy	0.9346	0.9325
Testing Accuracy	0.9256	0.9215

Figure #1: This graph visualizes the Accuracy Score for the Logistic Regression and Deep Neural Network's training and testing sets.

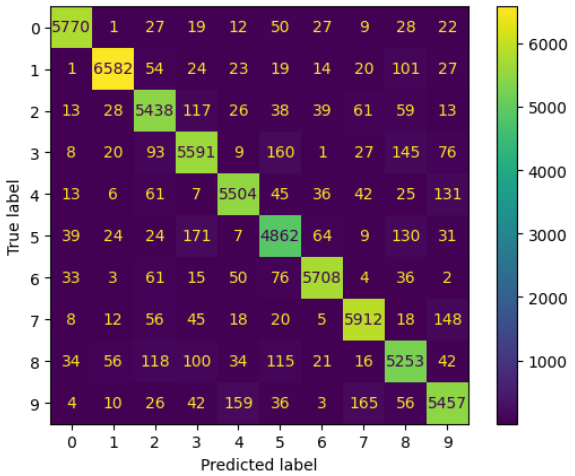

Figure #2: This confusion matrix visualizes the predictions the Logistic Regresion model made on the Testing Data.

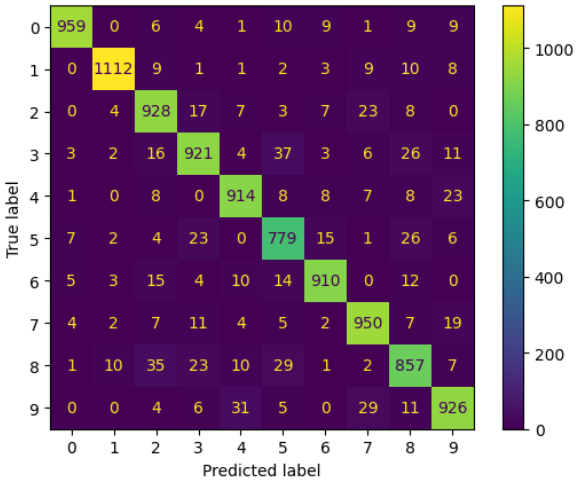

Figure #3: This confusion matrix visualizes predictions that the Logistic Regression made on the Training data.

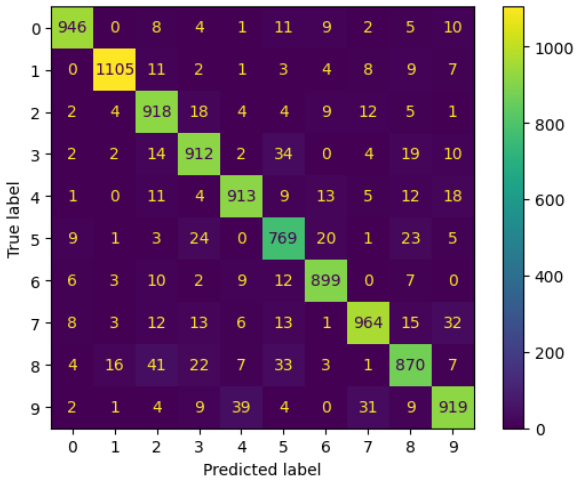

Figure #4: This confusion matrix visualizes the predictions the Deep Neural Network model made on the Testing Data.

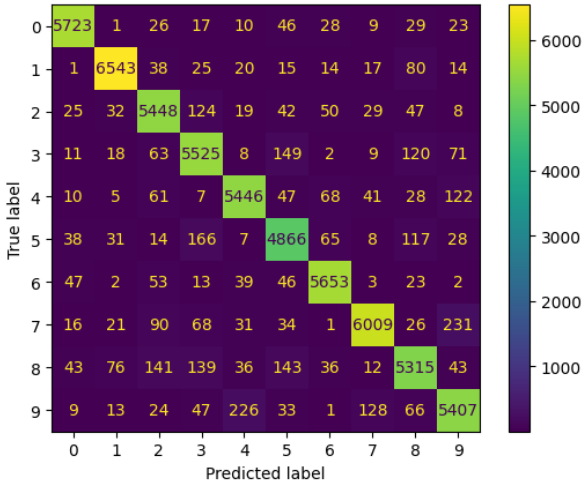

Figure #5: This confusion matrix visualizes predictions that the Deep Neural Network model made on the Training Data.

What happens to the loss and accuracy of your train/test set as you go through all 100 epochs (include a plot of the loss over time for both train and set set, and a plot of the accuracy over time for both train and set set)?

In regards to the loss of the Train and Test (Validation) set as we go through all 100 epochs, we can see a significant decrease in loss at around 8 epochs. The loss change for both sets is visualized in Figure #6. Following this, the training data loss begins to enter the elbow of the plot. For the validation set, the loss begins small ebb up and down around the loss value of 0.28, but after 8 epochs, the loss values begin to change much less compared to the training set.

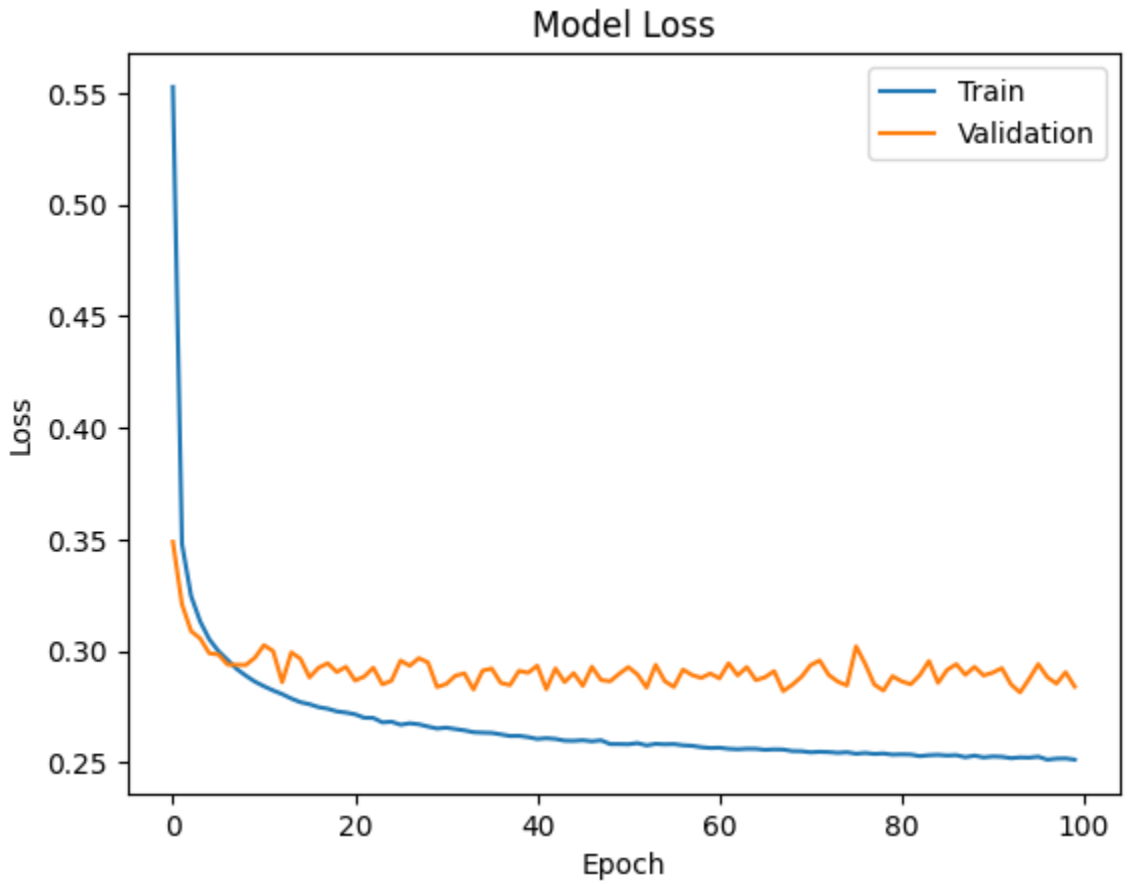

Figure #6: This plot is of the loss over time for both the training and testing (validation) set as it goes through all 100 epochs for the Deep Neural Network model.

Regarding the accuracy of the Training and Testing set as we go through all 100 epochs, there is a similar pattern with the ebb for the validation set. The accuracy change for both sets is visualized in Figure #7. For the training set, there is a consistent (natural ln) shaped increase in

the model's accuracy as the number of epochs increased. Following the elbow at around 10 epochs, the training set's accuracy continues to increase but at a smaller rate.

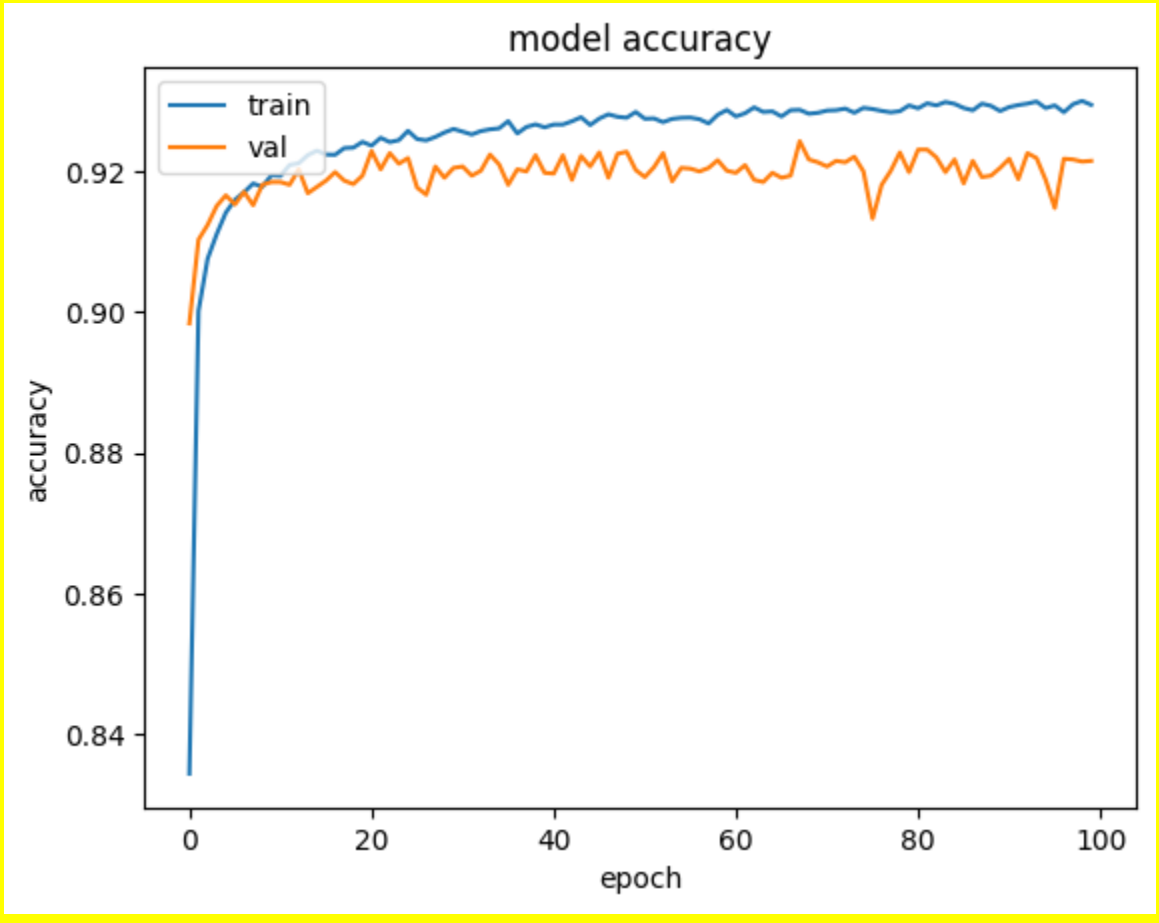

Figure #7: This plot is of the accuracy over time for both the training and testing (validation) set as it goes through all 100 epochs for the Deep Neural Network model.

What is the job of a loss function in general? What specific **loss function does your model use? Why is this loss function appropriate for predicting digits (0-9)**

A loss function's role is crucial in training a machine learning model. It quantifies how well the model performed by comparing its predictions with target values. Training a model aims to reduce or minimize this loss, which can lead to a more accurate model. The specific loss function we decided to use is "Categorical Crossentropy" which is more suited for classification tasks such as "digit recognition" where the goal is to classify instances into classes or categories. For this prediction, we are giving a range and requirement for the digits in the image to be. There is only one digit; the digits are whole numbers, ranging from 0 to 9 (inclusive). That being said, these requirements structure this prediction to be a classification problem, not a regression problem.

What is an activation function? What activation function did we use in your Neural Network? Why is this activation function appropriate for predicting digits (0-9)

An activation function is a type of math equation that will transform the data in a node. In our code, we used a softmax activation function to convert the classification output to a probability instead of a hard classification (similar to the difference between KMeans and GMM classification). This is appropriate for multi-classification situations like this with digits between 0 and 9 because we will get back the probabilities of the number being each of the digits. Then, we can pick the digit with the highest probability and classify it as that.

The first argument in any `Dense()` layer is the number of nodes in that layer. How many nodes does your last Neural Network layer have? Why did we choose that number?
The last layer in our NN has 10 nodes representing the probability of a number belonging to a category. Furthermore, there are 10 digits we are trying to classify (0-9), so we want to have 10 different outputs, each with a probability of that number belonging to each category.

If you had more complicated images, like images of pets which you're classifying as "Corgi" or "Not Corgi", do you think a logistic regression would do well? Why or Why not.

Given the model's simplicity, the logistic regression model wouldn't perform as well on "Corgi" classification vs the "number" classification. The model is not complex enough to capture the features and relationships of what makes something a Corgi or not.

What's the benefit of using Deep Neural Networks compared to simpler models like Logistic Regression, Decision Trees...etc?

A benefit of using a DNN over other models is that the DNNs are much more customizable and better for capturing complex relationships in high-dimensional data. It gives us the power to easily transform the data using activation functions and to change the shape of the data with hidden layers.

Discussion/Reflection

A few sentences about what you learned from performing these analyses, and at least one suggestion for what you'd add or do differently if you were to perform this analysis again in the future.

After performing these analyses, we learned about the importance of model selection based on the data and problem that we work with. One of the biggest things that we learned is the importance of the Bias-Variance Tradeoff that occurs when creating the Logistic Regression model (much less complex) and the Deep Neural Network model (more complex) and comparing the results of both. A higher complex model does not always equate to a higher performance, rather that type of model can overfit the data depending on the complexity of the task and data. One of the things that we would like to add if this analysis was completed again is adding in Gradient Boosting and/or Random Forest Classification to the model selection pool to see how the other models compare to the Logistic Regression and Deep Neural Network models.