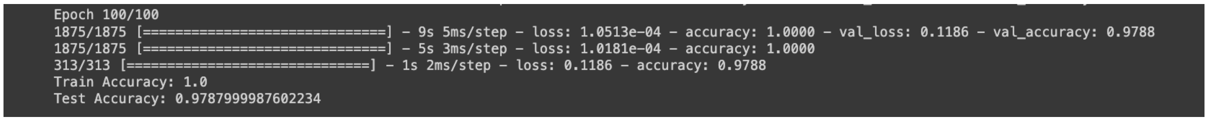
Homework 4

Introduction

In this project we are trying to get a model that can accurately identify the number depicted in a small grayscale image. Each image is a 28x28 pixel grid, showing a single digit from 0 through 9, making up a total of 784 data points that our models will use to make their predictions. An example of an impact it could have is having a computer automatically input data from a photo of an image of a number into a database rather than a person manually inputting what number it is.

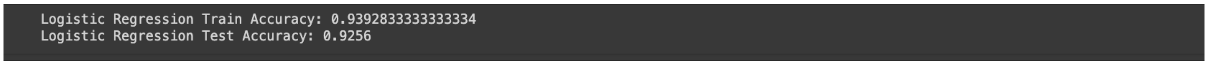
Methods

We used two models: a simple Logistic Regression and a more complex Deep Neural Network. Logistic Regression compares pixel data to recognize digits, while the Deep Neural Network processes data through multiple layers, learning to identify more intricate patterns. For our preprocessing we standardized the pixel values to prepare the images for both models.



```
Epoch 100/100
1875/1875 [==============================] - 9s 5ms/step - loss: 1.0513e-04 - accuracy: 1.0000 - val_loss: 0.1186 - val_accuracy: 0.9788
1875/1875 [==============================] - 5s 3ms/step - loss: 1.0181e-04 - accuracy: 1.0000
313/313 [==============================] - 1s 2ms/step - loss: 0.1186 - accuracy: 0.9788
Train Accuracy: 1.0
Test Accuracy: 0.9787999987602234
```

Figure 1: Deep Neural Network Model Accuracy



```
Logistic Regression Train Accuracy: 0.9392833333333334
Logistic Regression Test Accuracy: 0.9256
```

Figure 2: Logistic Regression Model Accuracy

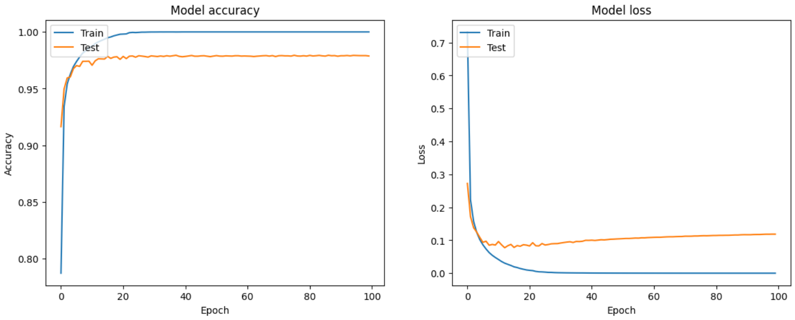


Figure 3: Model Accuracy Plot

Results

The Logistic Regression was about 93% accurate on training data and 92.56% on new data. The Deep Neural Network scored 100% on training data but 97.88% on new data, which could mean it might have memorized the training data too well which could have affected its ability to handle new images and overfit. You can see this overfitting on our plots. In terms of computing time performance, the logistic regression was a very fast process but the deep neural network model took a very long time due to its complexity.

1. 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?

The Deep Neural Network model had a training accuracy of 100% and a test accuracy of about 97.88%. The perfect training accuracy suggests that the model has learned the training data very well, but it's possible it could be overfitting. The test accuracy being slightly lower is a sign it could be slightly overfitting. The logistic regression model showed a training accuracy of about 93.93% and a test accuracy of about 92.56%. In this model it's also very accurate, and also could have a slight overfitting. It's less overfitted compared to the difference between train and test in the other model, with a smaller gap between training and test accuracy. However the overall accuracy is lower than DNNs accuracy which makes sense because DNN is a much more complex model. The fact that DNN is so much more complex and took way more time to train, it's not surprising it had a higher accuracy but the fact that it was 100% for training was a little surprising since this means its perfect training.

2. 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)?

The accuracy plot shows that the DNN model quickly reached high accuracy on the training set and kept it throughout the epochs. The test accuracy also increased quickly but plateaued. This shows the model wasn't improving much on the test set after a certain point. The loss plot shows that the training loss dropped to near zero, while the test loss decreased and then flattened out. This could mean the model could be overfitting because it's not generalizing improvements beyond the training data.

3. 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)

The loss function in a neural network is difference between the predicted values and actual values. The loss function used is categorical_crossentropy, it works for a multiclass classification problem like the MNIST dataset since we're predicting one of the classes from 0 to 9. It measures the performance of the model whose output is a probability value between 0 and 1.

4. 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)

It introduces non-linearity into the model, which allows it to do more complex tasks. The neural network uses the relu activation function for intermediate layers since it allows the model to learn faster and perform better. The softmax activation function is used to convert logits to probabilities that are 1, which is needed for a multiclass classification.

5. 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 of the neural network has 10 nodes for the 10 possible classes for the digits 0-9. This number is chosen because the network needs to output a probability distribution over all possible classes.

6. 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.

Logistic regression would probably not be that accurate with more complex images like for seeing if a dog is a corgi or not. Because of its simplicity, it cant take into account all the complex features and relationships that distinguish if a pet is a corgi. There could be a cat that looks kinda like a corgi and might falsely say it's a corgi because it's not complex enough to have all these features and relationships. A Deep Neural Network would be better since it has hierarchical feature learning and it's a more complex model.

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

Deep Neural Networks are beneficial over simpler models because they can model complex and non-linear relationships in the data. This makes them better for recognizing images, speech, and text, where the features interact in complex ways that simpler models like logistic regression cant accurately predict. The con is they require more data and computational resources and are prone to overfitting.

Discussion/Reflection

This project showed us the pros and cons of using a DNN model because although it can be very accurate, it's so complex that it was prone to overfitting and that is something to take in mind. The time it took for the DNN model also showed us that this model is very computationally expensive so in situations where that's a factor the slight performance boost might not be worth it. For the future we should do some research about the methods we could implement in our code that prevent DNN from overfitting.