**NEURAL VISION: UNRAVELING HANDWRITTEN DIGITS**

COMP/EECE 7745 MACHINE LEARNING

Project Report

Pranathi Sundari (U00869903)

# Introduction

Question 1: Computational Graph

Σ is the sigmoid function

=1/1+e-x

From the given information,

W1 = -1.5, x1=0.7, x2=0.2 and w2=0.35

Y = Tanh {(x1w1 +max (x2, w2))}

Y=sigmoid (0.7x -1.5) + Max (0.34,-0.35)

=sigmoid (-1.05 + 0.34)

Y= sigmoid (0.71) = 0.607

Y =0.607 = 0.61

The graph for sigmoid function is S – shaped

Sig (t) = 1/1+e-t = et/1+et

=1/2(1 + tanh (t/2))

-1

-2

-3

-4

1

2

3

4

**0**

# Methodology

A methodology refers to a method or system followed when doing something, (Goundar, 2012). In designing the deep neural model, Keras methodology was used.

Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation.  Keras allows you to switch between different back ends. The frameworks supported by Keras are: [Tensorflow](https://www.simplilearn.com/tutorials/deep-learning-tutorial/tensorflow), Theano, PlaidML, MXNet and CNTK (Microsoft Cognitive Toolkit).

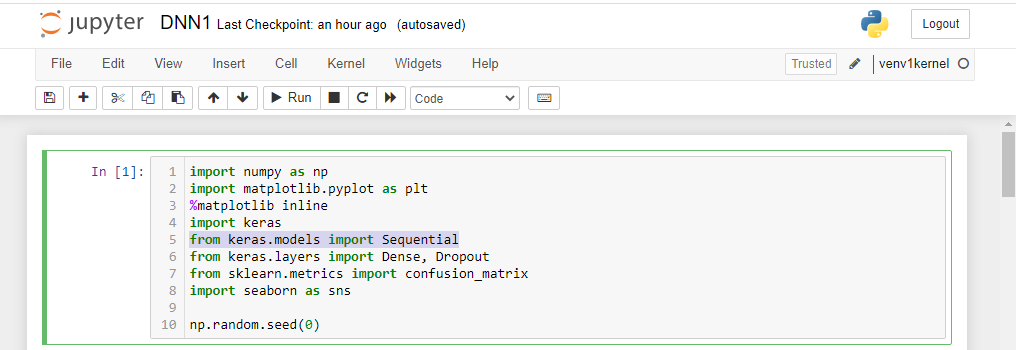
Among the frameworks of Keras methodology we employed the use of TensorFlow then constructed the network. TensorFlow is a Python-friendly open source library for numerical computation that makes machine learning and developing neural networks faster and easier, (Yegulalp, 2022).

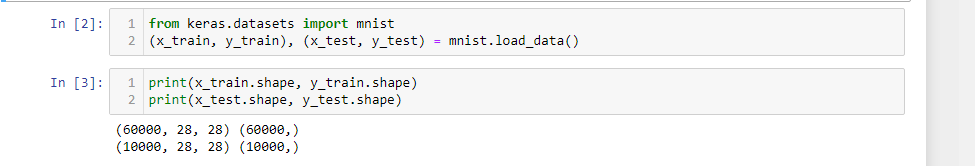
Our model builds the model layer by layer using the sequential API in Keras. The created model relies on an instance of the sequential () class. The first model has an input variable, a hidden layer consisting of two neurons, and a binary output. For the second model, an additional layer was created and added. The model contains the following details: the hierarchy and hierarchy order of the model, the output form of each hierarchy (number of elements in each output data dimension), the number of parameters (weights) per count, and the total number of parameters in the model.

# Model descriptions

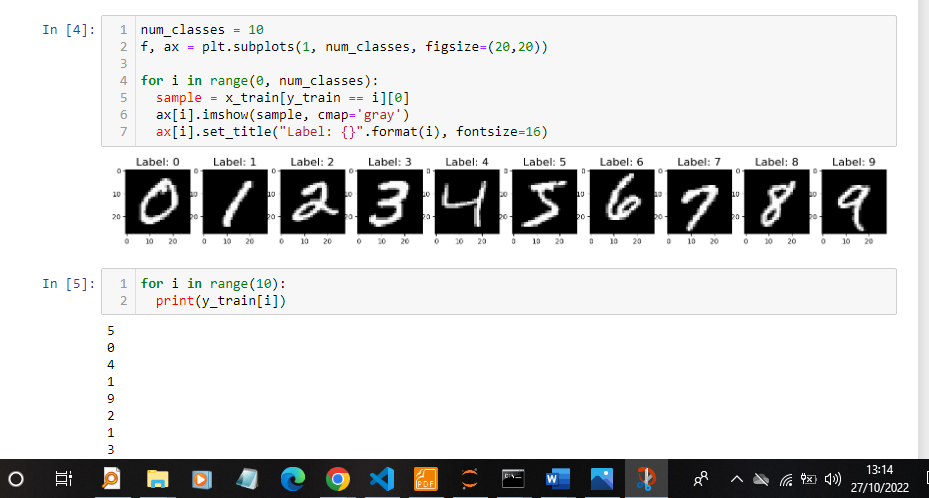
We started by importing all the required modules. Then processed matrix values using NumPy, then displayed images with Matplotlib, and finally, created a neural network model with Keras.

The dataset can then be loaded using the following code. Once the following code is run, the four variables X train, y train, X test and y test (where X is the target label and y is the image) exist. These training and test sets have 60,000 and 10,000 photos, respectively, and they are all the same size (28x28 pixels).

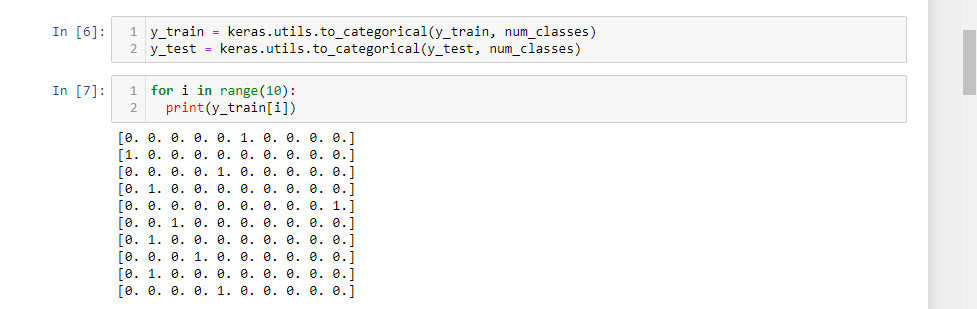


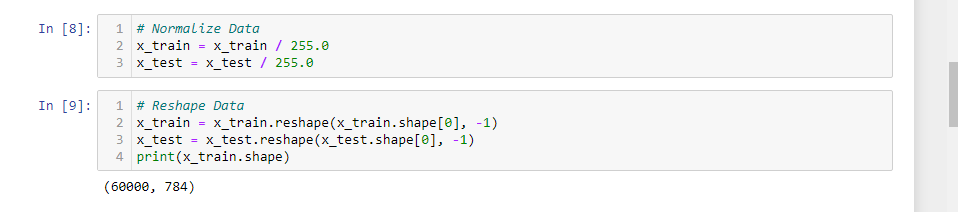


We used following code to display the first 10 images in the dataset.



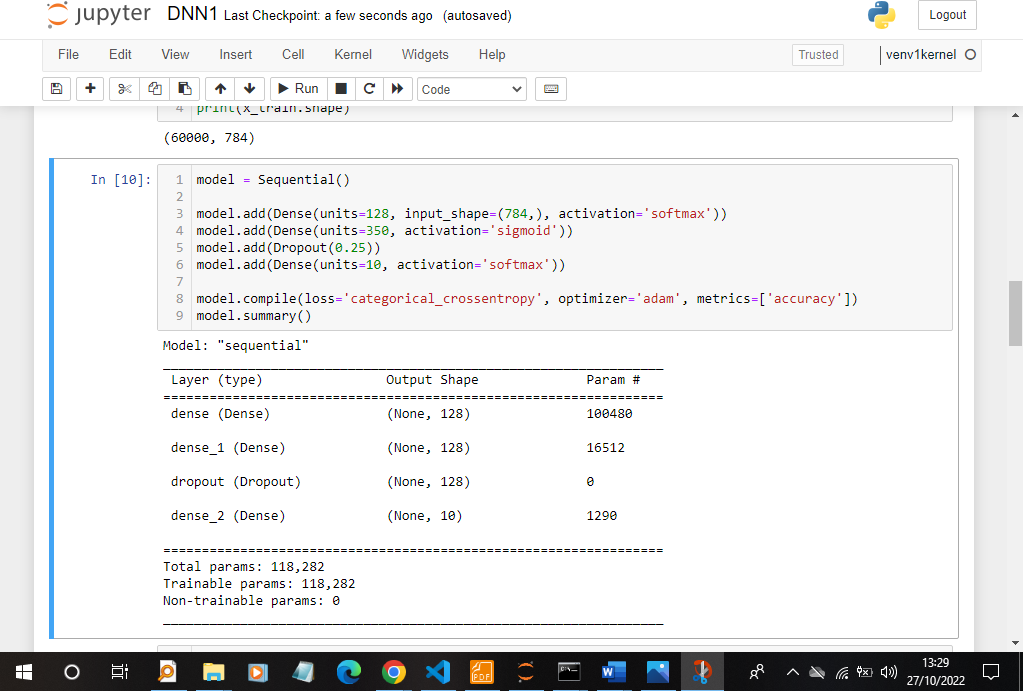
Using the categorical() method in Keras packages we combined all the labels into one presentation. We can observe that the result is an array with all its values set to zero exception of some index values. One-hot encoding is the term used to describe this representation.  We can use the following code to one-hot encode all of the target labels (both y train and y test):





Initializing the sequential model is the first step. In this case, we start the neural network model with a plane, because a 28 x 28 pixel image (two-dimensional) must be transformed into 784 values ​​(one-dimensional). Then we connect these 784 data using 350 neurons with sigmoid activation function. The last component is the second dense layer, which serves as our seed layer and uses a softmax activation function. Since we have 10 different classes in our classification problem, we need to use 10 neurons in the last layer.

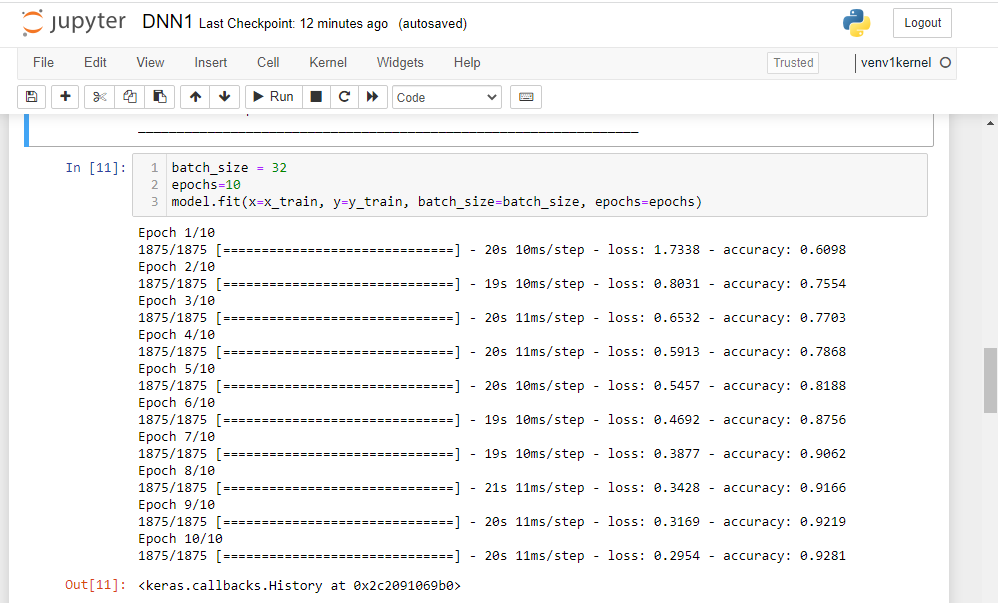
To see the details of our architecture, we can also use summary(). We use categorical cross-entropy as a parameter of the loss function. The next step is to use the Adam optimizer, which is not only the best overall, but also our recommendation. Finally, accuracy is the metric we send to the metric parameter to score the classifier.



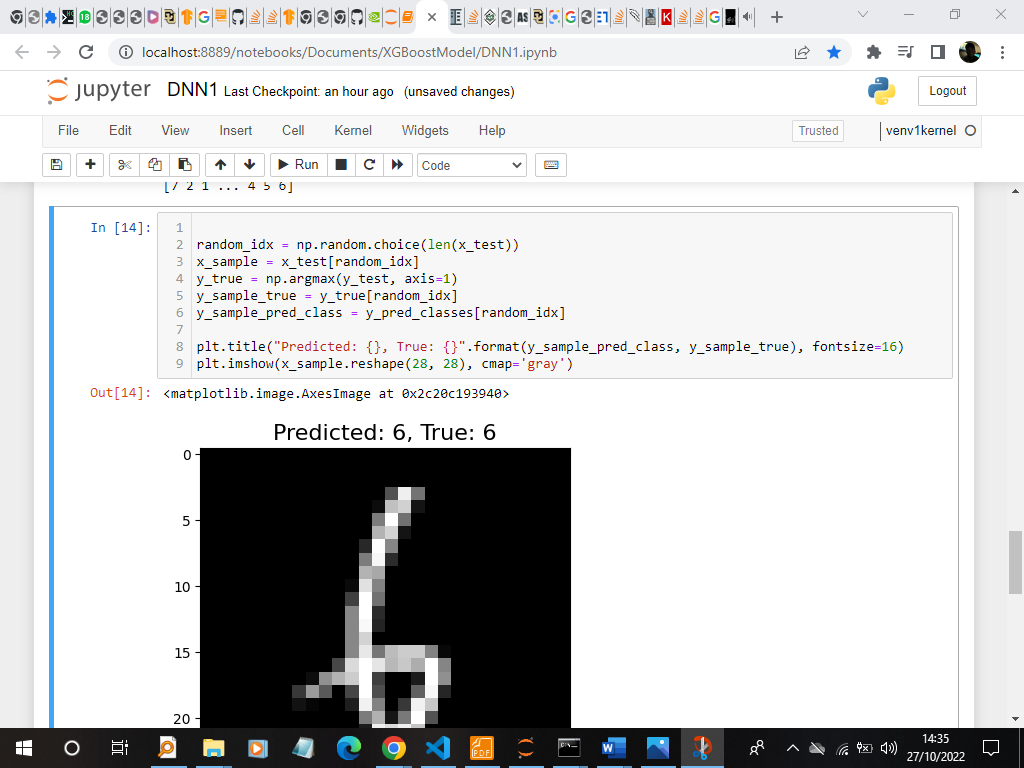
# Experiment and Results

## Training and testing logs

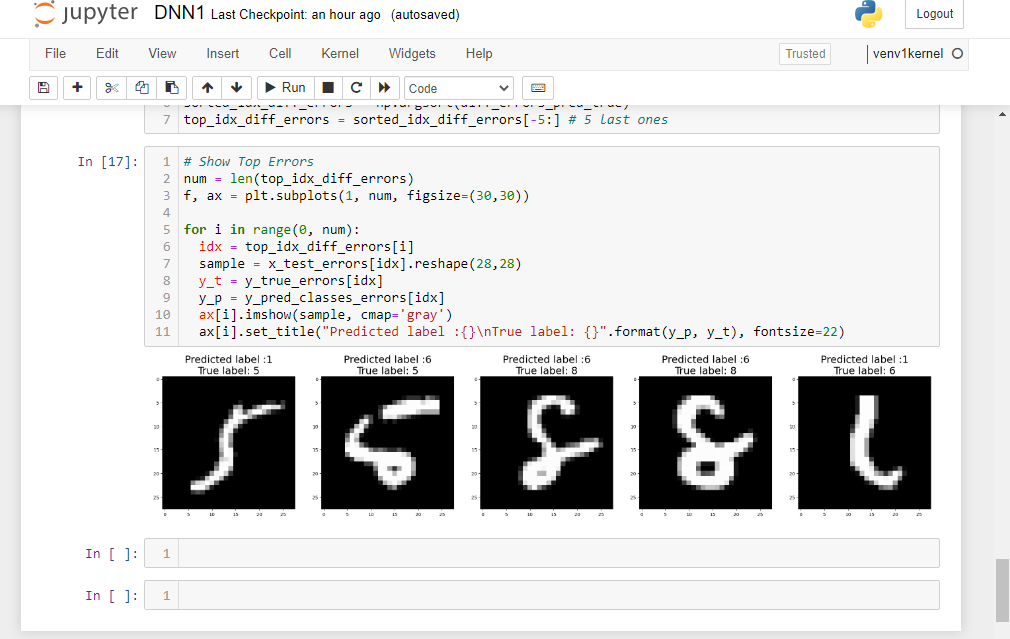
To get the required results, we have to train and test our now created model. Running the model's fit () method is all that's required to train it.



Using the below code we can test image display together with its prediction at the same time. We can see that the image is well classified.



The below screenshot shows the top testing errors. Here the sample is misclassified. For example here, the prediction label indicates 1 instead of 5.



## Discussion and comparison

We observed the two models sorting the Images of handwritten digits into one of 10 categories, each of which represents an integer value from 0 to 9, inclusively. MNIST is a dataset that's "solved" in the sense that it's commonly utilized and well understood. When compared to the XGBoost model and the hold out test dataset, the best-performing models are deep learning neural networks with a classification accuracy of over 99% and an error rate of between 0.4% and 0.2%.

When comparing the model with the configurations 784, 350, 120, 10 to the model with the configurations 784, 500, 250, 100, 10. The later model performed better in terms of predictions but the computational time was low. The computational time for the first model was high with high prediction errors. In terms of performance, the deep neural network model outperformed the XGBoost model.

# Conclusion

Basing on the tests carried out, we can conclude that, deep neural network model performs better than the XGBoost model. This is because its computational time in terms of predicting errors was higher than the XGBoost model.

In terms of accuracy, deep learning is more accurate than XGBoost model. This is because it had a lower error rate compared to XGBoost model.

# References

Goundar, S. ( 2012). Chapter 3 - Research Methodology and Research Method. *Cloud Computing*.

Yegulalp, S. (2022, 6 23). What is TensorFlow? The machine learning library explained. *Software development*.

McDonnell, M. D., Tissera, M. D., Vladusich, T., van Schaik, A., & Tapson, J. (2015). Fast, Simple and Accurate Handwritten Digit Classification by Training Shallow Neural Network Classifiers with the ‘Extreme Learning Machine’ Algorithm. *PLOS ONE*, *10*(8), e0134254. <https://doi.org/10.1371/journal.pone.0134254>

Ciresan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *2012 IEEE Conference on Computer Vision and Pattern Recognition*. <https://doi.org/10.1109/cvpr.2012.6248110>

Lee, H., & Song, J. (2019). Introduction to convolutional neural network using Keras; an understanding from a statistician. *Communications for Statistical Applications and Methods*, *26*(6), 591–610. https://doi.org/10.29220/csam.2019.26.6.591