**Assignment 3:** MNIST CNN Visualization

**Name: Tyler Conger Email: Tyler-Conger@Tyler-Conger.com**

Statement:

For this assignment's preparation, the author(s) have utilized Chat GPT-3.5, a language model created by OpenAI. Within this assignment, Chat GPT was used for purposes such as brainstorming, asking specific code error questions, and help with proof-reading”

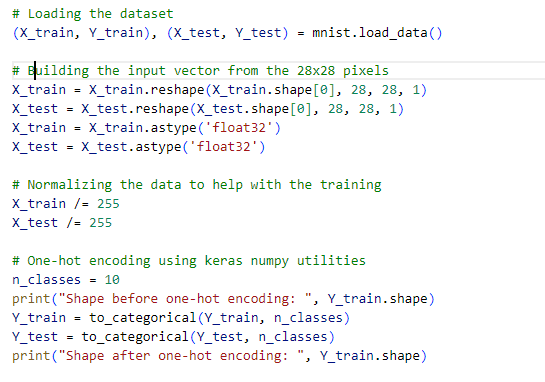
**Objective**: Train a Convolutional Neural Network (CNN) on the MNIST dataset and visualize the learned features at different layers.

**Tasks**:

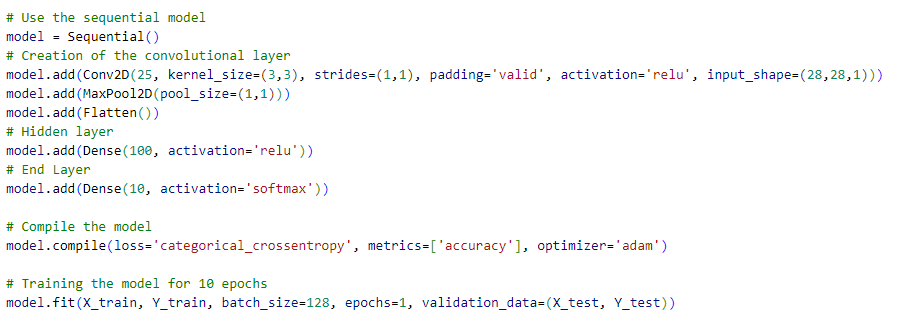
1. **CNN Implementation**:
   * Load the MNIST dataset.
   * Preprocess the dataset: normalize the images, convert labels to one-hot vectors, etc.
   * Design and implement a CNN model using TensorFlow/Keras.
   * Train the model on the training data and validate its performance on the validation data.
   * Report the test accuracy and loss.

The first step in the implementation is to load the data set this can be done through an import statement such as “from keras.datasets import mnist” then the next line uses this import of mnist data set as such: “(X\_train, Y\_train), (X\_test, Y\_test) = mnist.load\_data()”

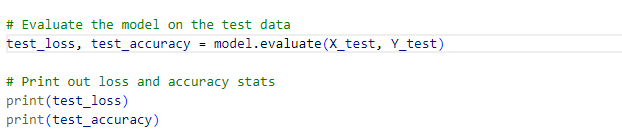
The next step is to pre-process the data the mnist data requires minimal preprocessing but some normalization is required such as rescaling the image to have each pixel on a range between 0 – 1 this is done by dividing each by 255. Finally, we must use one-hot encoding to convert the data to categorial information that is used. All of these steps can be seen in the below figure.

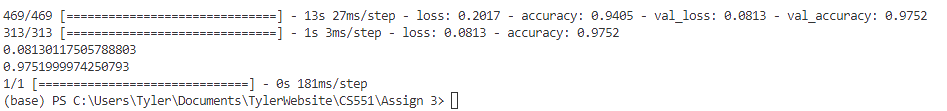


Next is the process of model selection and usage. We use a sequential model with a single convolutional layer. More could be used but I was able to achieve a high accuracy with limited layer usage. The model is compiled and trained with 10 epochs. This can be seen in the image below.



Finally, the accuracy of the model is tested and evaluated with the test data which was chosen at the start of the process when the data was loaded. Then the accuracy and test loss values are reported via print statements as seen in the accomponying image below.

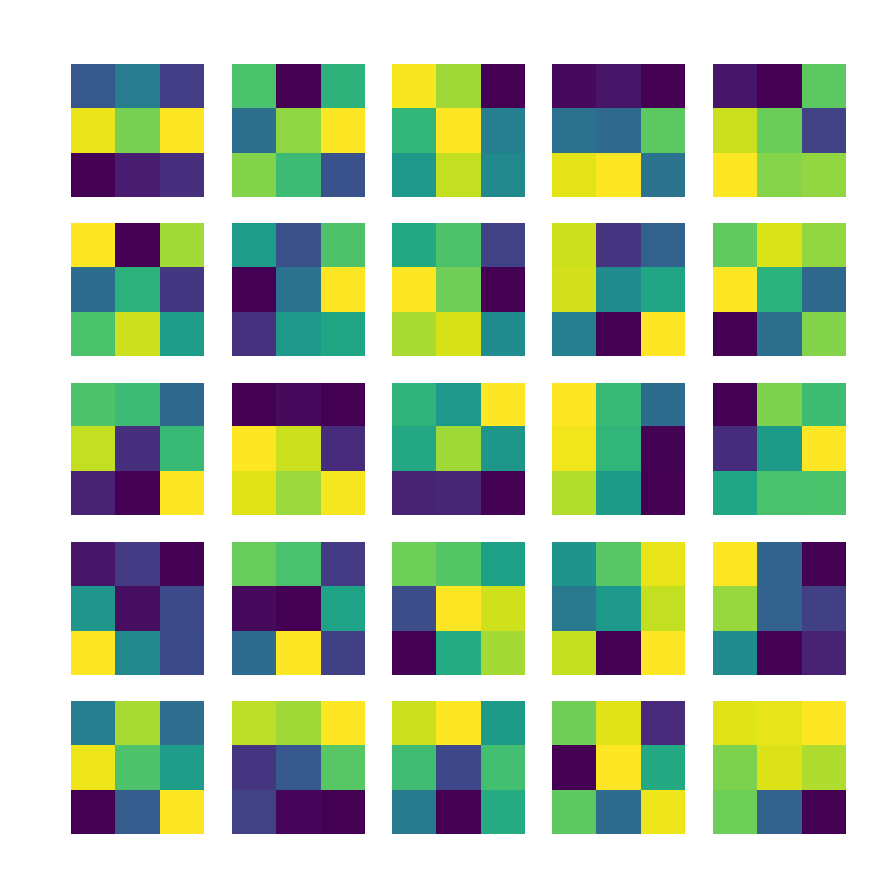




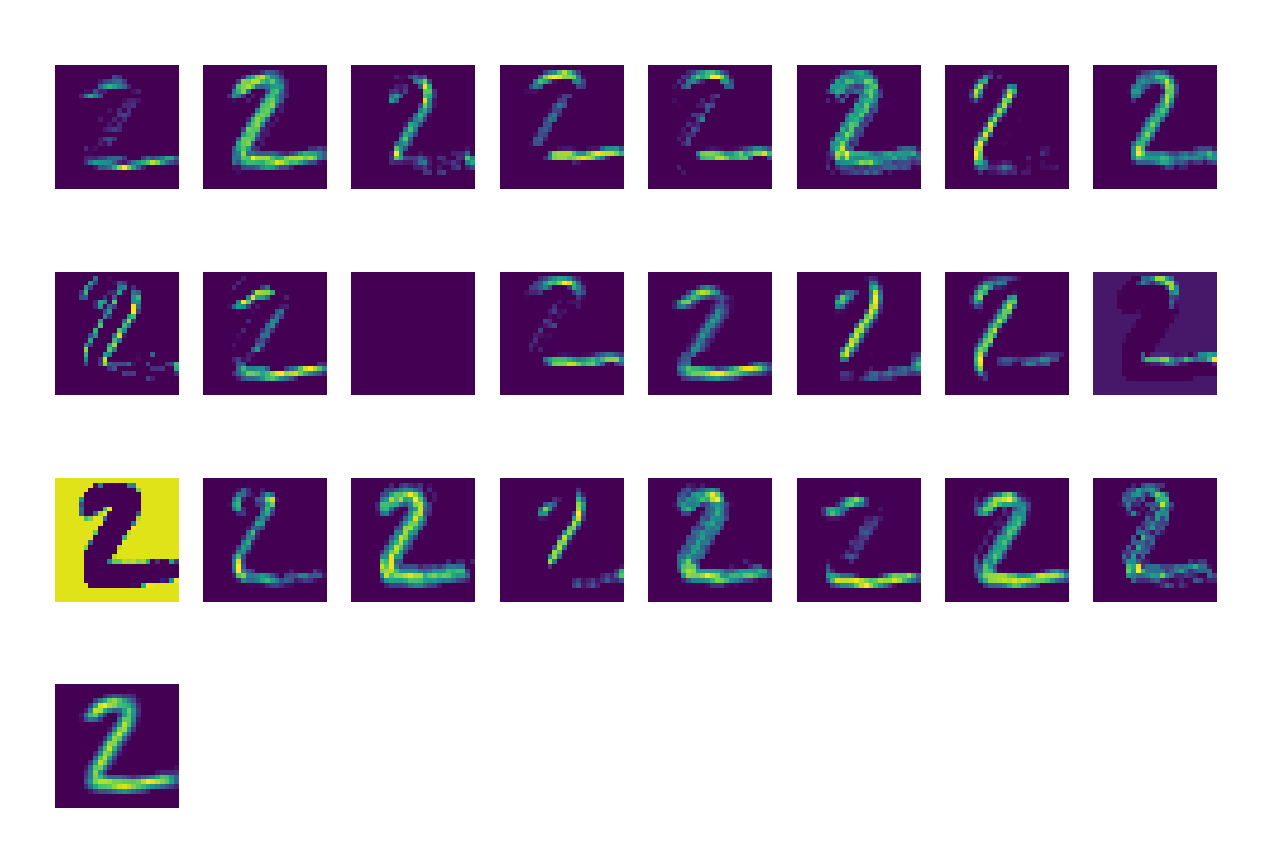
As can be seen the accuracy is around .97 which is pretty good considering we are representing a high degree of accuracy on the MNIST data set, and properly identifying a high percentage of the numbers.

1. **Visualization**:
   * Visualize the filters/kernels from the initial convolutional layers. What kind of features (e.g., edges, textures) can you infer from these visualizations?
   * Use techniques like feature map visualization to see the activations produced by different convolutional and pooling layers for a given input image.
   * Advanced (optional): Implement DeepDream or style transfer for MNIST, or visualize higher layer activations using dimensionality reduction techniques.

The initial step involves visualizing edges using the first filter, facilitated by the visualize\_filters function. This function retrieves the model's layers and weights, allowing the examination of each filter. Through the use of imshow, the filters are displayed and added to the output visualization, providing a comprehensive view of the model's edge-detection capabilities.



In the next piece we visualize the feature maps and how they correspond to the actual MNIST data information. This can be seen in the image below.



Subsequently, the focus shifts to visualizing the feature maps and their correlation with actual MNIST data. The resulting image vividly showcases the highlighted regions representing edges of the number 2 within the MNIST dataset. This visualization offers valuable insights into how the model perceives and comprehends the distinctive features that define the numeral '2'.

**GitHub Resources**:

<https://github.com/ZZUTK/An-Example-of-CNN-on-MNIST-dataset#kernels>

* This repository provides an example of a CNN on the MNIST dataset, with visualizations of the learned kernels in the first and second convolutional layers.
* It showcases the effect of using a wider or deeper CNN for better classification accuracy​.
* You can leverage the Github code, but please describe what kind of work you have done based on it.

**Assessment Criteria**:

1. **Code Quality** (30%): Readability, modularity, and efficiency.
2. **Model Performance** (10%): Achieving a high accuracy on the test dataset.
3. **Visualization Quality & Analysis** (30%): Clarity and relevance of visualizations, and the depth of analysis on what the CNN might be learning.
4. **Report** (30%): Clear documentation on implementation decisions, model architecture, and analysis of visualizations.