Hareeshma S

Practical Question

**Knowledge Test**

**Practical Questions (25 marks)**

**Activity-1**

**Aim:**

Build and compile a simple neural network using Keras to classify the MNIST dataset (handwritten digits). The model should include at least one hidden layer. Provide the code and briefly explain each step.

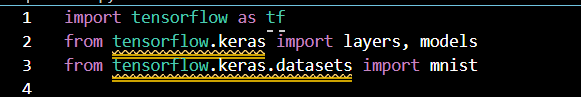
**Requirements:**

* Python
* TensorFlow and Keras libraries (included with TensorFlow)
* MNIST dataset (available directly through Keras)
* VS code

**Procedure**

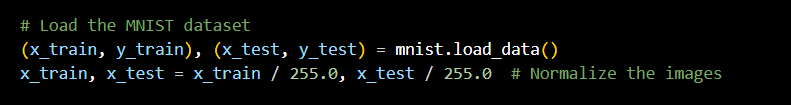
**Step-1**

First, we need to import the required libraries. Keras is part of TensorFlow, and we will use it to build our neural network.



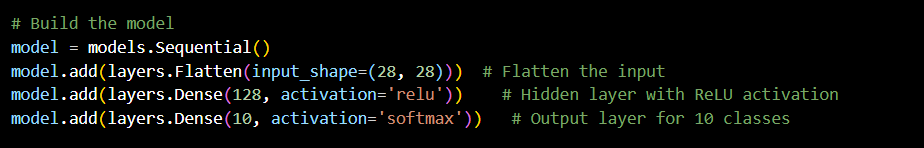
**Step-2**

Next, we load the MNIST dataset and preprocess it. The dataset is divided into training and test sets. We need to normalize the pixel values and convert the labels to categorical format.



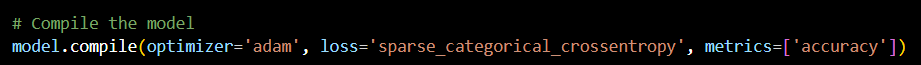
**Step-3**

We define a simple neural network using the Sequential API from Keras. The network includes an input layer (flattening the 2D images), a hidden layer with 128 neurons, and an output layer with 10 neurons (one for each digit class).



**Step-4**

We compile the model by specifying the loss function, optimizer, and metrics to monitor. For classification, we use the categorical crossentropy loss function and the Adam optimizer.



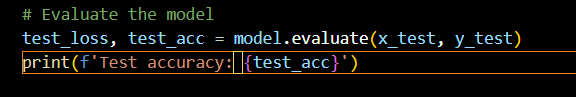
**Step-5**

We train the model using the fit method, specifying the training data, validation data, batch size, and number of epochs.

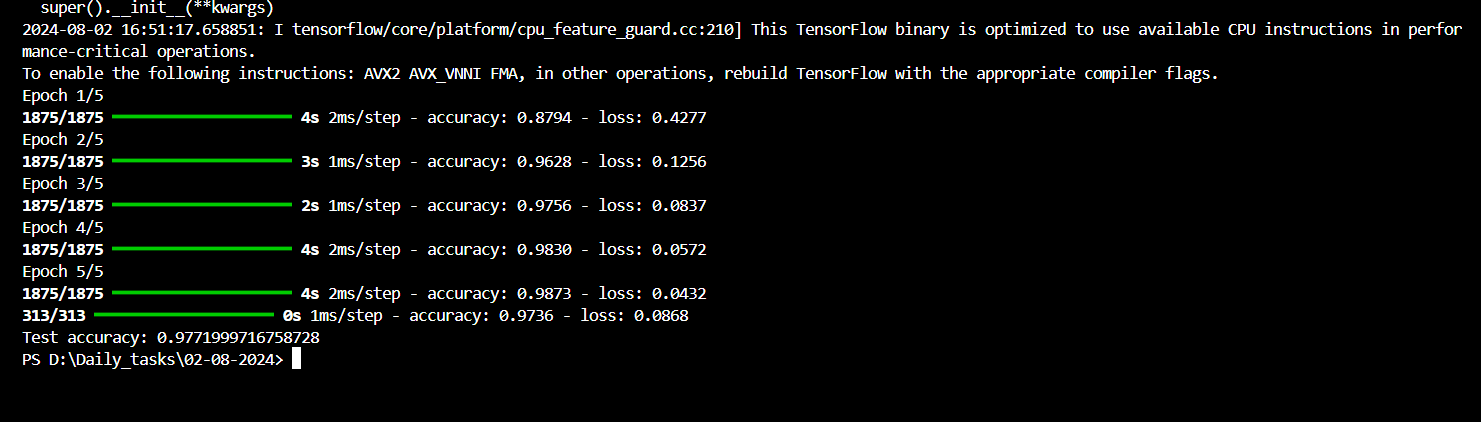


**Step-6**

Evaluate the Model: Assess the model's performance on the test data.

****

**Output**

****

**Activity-2**

**Aim:**

Implement data augmentation on a given image dataset using Keras. Show at least three different augmentation techniques and explain how they help improve model performance.

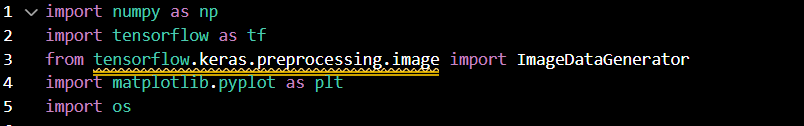
**Requirements:**

* Python
* TensorFlow and Keras libraries (included with TensorFlow)
* VS code

**Procedure**

**Step-1**

Import the necessary libraries for data augmentation.



**Step-2**

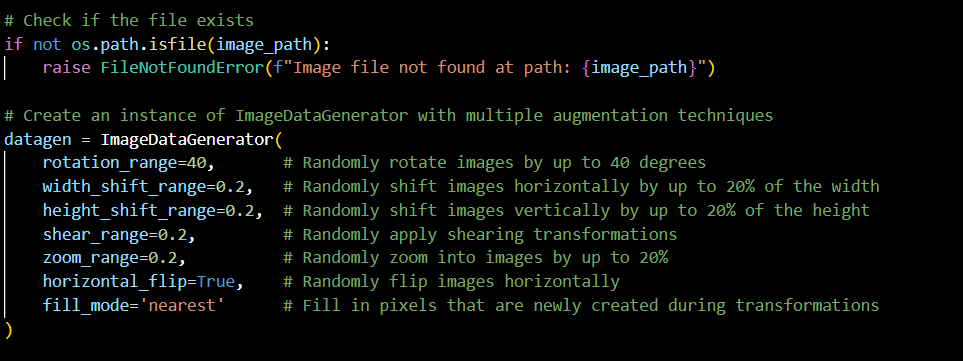
**Load and Preprocess Data**:

* Load a sample image from your dataset. In practice, you would use a directory of images. Here, we use a single image for demonstration.

****

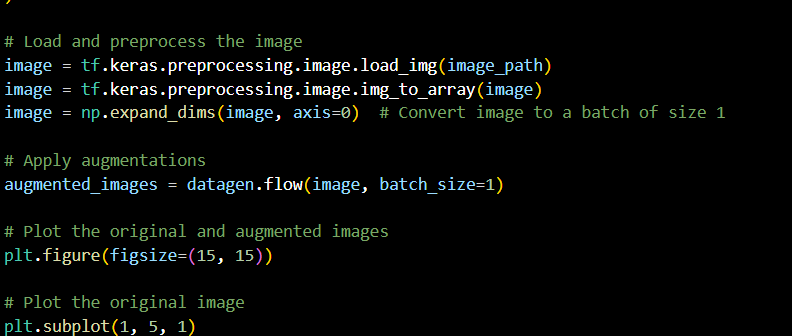
**Step-3**

Define Augmentation Techniques: Set up the ImageDataGenerator with different augmentation parameters.



**Step-4**

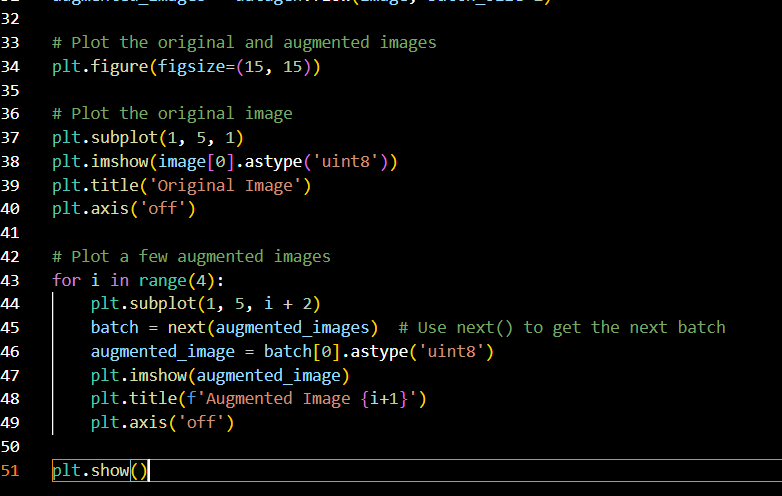
Apply Augmentations: Generate augmented images from the original image.

****

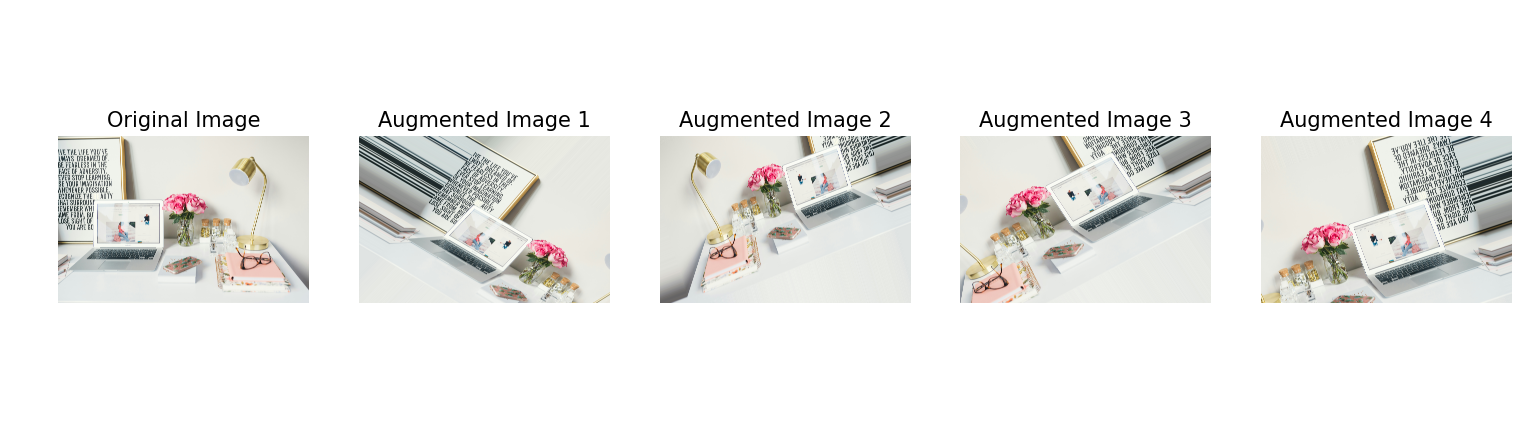
**Step-5**

Visualize Results:

* Plot the original image and several augmented images using matplotlib to visualize the effects of the applied augmentations.



**Out put**

****

**Activity-3**

**Aim:**

Implement a custom loss function in TensorFlow/Keras. Explain the purpose of the loss function and provide an example scenario where it would be useful.

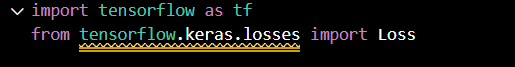
**Requirements:**

* Python
* TensorFlow and Keras libraries (included with TensorFlow)
* VS code

**Procedure**

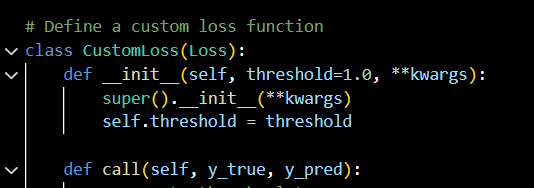
**Step-1**

Import Libraries: Import TensorFlow and other necessary libraries.



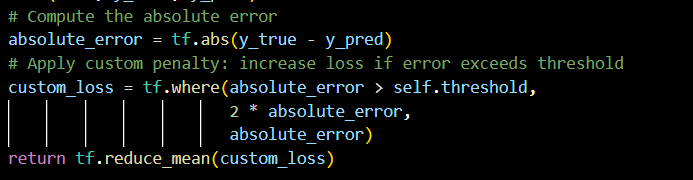
**Step-2**

Define the Custom Loss Function: Create a custom loss function by subclassing tf.keras.losses.Loss.



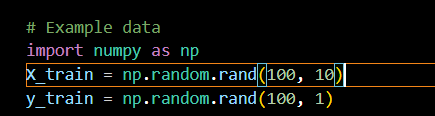
**Step-3**

Create and Compile the Model: Define a simple neural network model and compile it using the custom loss function.



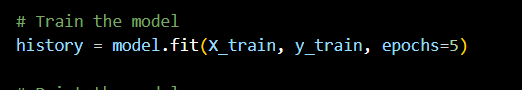
**Step-4**

**Generate Example Data**: Create synthetic data for training the model.



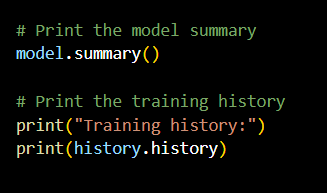
**Step-5**

Train the Model: Train the model using the example data and the custom loss function.

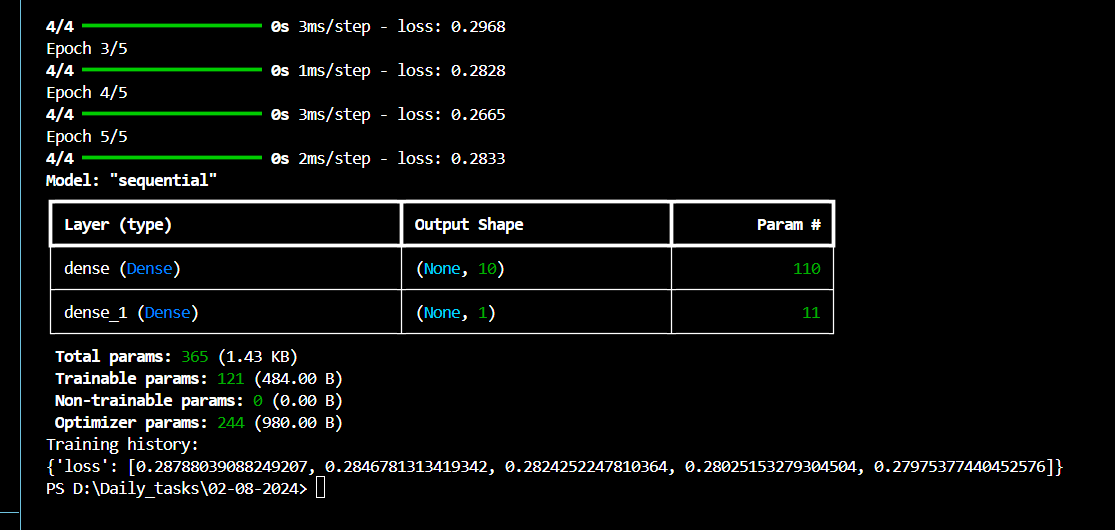


**Step-6**

Review Results: Print the model summary and training history.



**Output**

****

**Activity-4**

**Aim:**

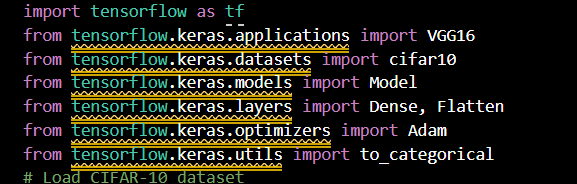
Use a pre-trained model (such as VGG16 or ResNet) available in Keras for a simple image classification task. Fine-tune the model for a new dataset and describe the steps taken**Requirements:**

* Python
* TensorFlow and Keras libraries (included with TensorFlow)
* VS code

**Procedure**

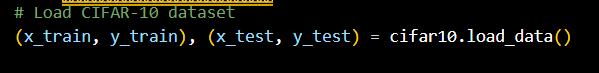
**Step-1**

Import Libraries: Import TensorFlow and other necessary libraries.



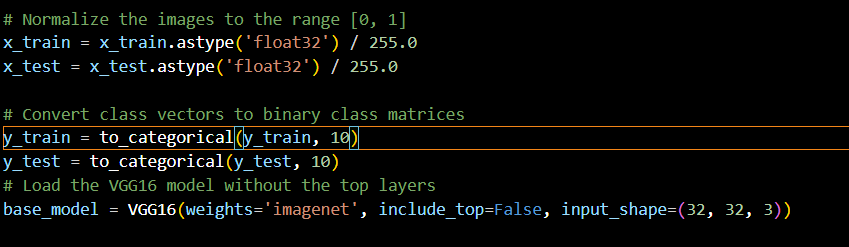
**Step-2**

Load and Prepare the Dataset: Load and preprocess the new dataset.



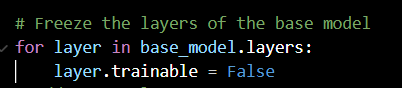
**Step-3**

Load the Pre-Trained Model: Load the pre-trained VGG16 model without the top layers.



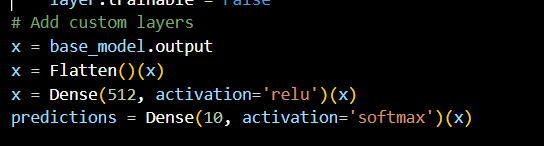
**Step-4**

**Freeze the Pre-Trained Layers**: Prevent the pre-trained layers from being updated during training.



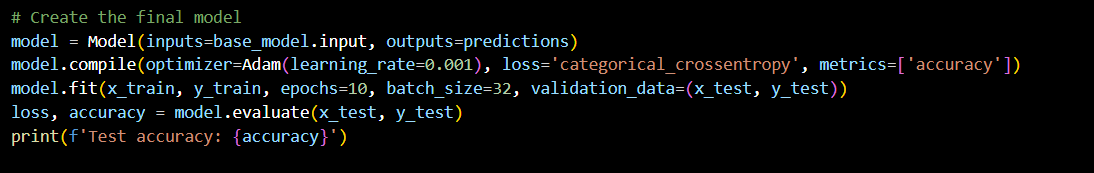
**Step-5**

Add Custom Layers: Add new layers to adapt the model to the new dataset

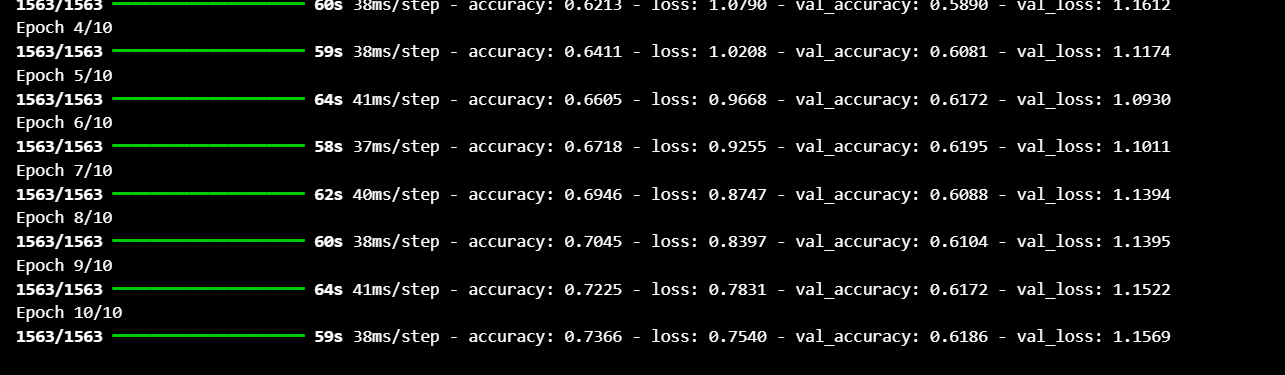


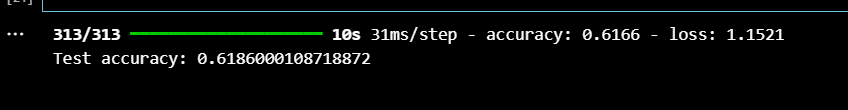
**Step-6**

Compile the Model: Define the optimizer, loss function, and evaluation metrics. Train the model on the new dataset. Assess the performance of the model on the test set.



**Output**

****

****