DOCUMENTATION

Knowledge Test

SUBMITTED BY

AYANA K

NSTI CALICUT

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.

LIST OF HARDWARE/SOFTWARE REQUIREMENTS:

1. Laptop/Computer with Windows OS
2. Vs code and Browser

Code/Program/Procedure (with comments):

Objective

Image Classification with Pre-trained VGG16 Model

Steps

1. Import Libraries

   - os: For file and directory operations.

   - tensorflow and tensorflow.keras: For deep learning and image preprocessing.

2. Define Paths

   - train_dir: Path to the training dataset.

   - val_dir: Path to the validation dataset.

3. Check and Create Directories

- check_directory(path): Function to verify if a directory exists. If not, it creates the directory.

4. Data Preprocessing

- ImageDataGenerator: Used to perform real-time data augmentation and normalization.
  - train_datagen: Applies augmentation (rotation, shift, flip) and normalization to training images.
  - val_datagen: Normalizes validation images.

5. Load Data

- flow_from_directory: Loads and preprocesses images from directories for training and validation.

6. Model Setup

- Load Pre-trained VGG16 Model:
  - Load VGG16 without the top classification layer, using pre-trained weights from ImageNet.
  - Freeze the layers to prevent them from being updated during training.
- Add Custom Layers:
  - Add a Flatten layer, a Dense layer with 256 units, and a Dense layer with softmax activation for classification.

7. Compile and Train Model

- Compile:
  - Use Adam optimizer with a learning rate of 0.0001.
  - Set loss function to categorical crossentropy and track accuracy.

- Train:

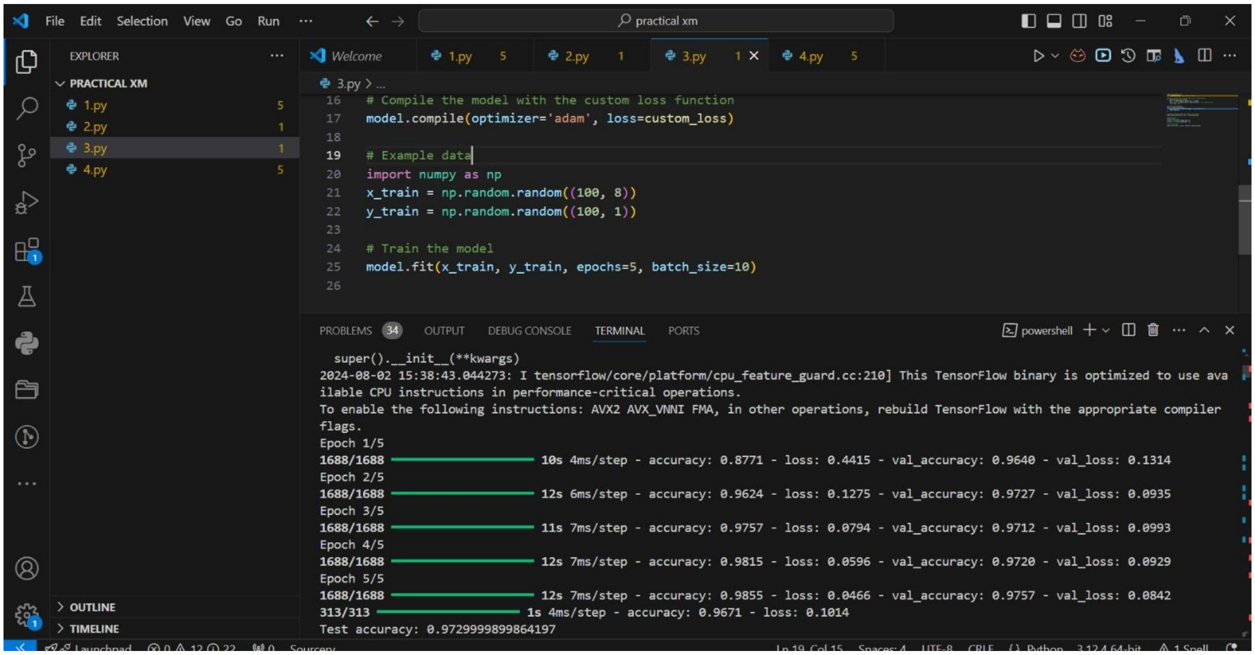
  - Fit the model using the training and validation data for 5 epochs.

Usage

1. Ensure Directories: Place your image data in the data/train and data/val directories, with subdirectories for each class.

2. Run the Script: Execute the script to train the model.

Output :



```python
# Compile the model with the custom loss function
model.compile(optimizer='adam', loss=custom_loss)

# Example data
import numpy as np
x_train = np.random.random((100, 8))
y_train = np.random.random((100, 1))

# Train the model
model.fit(x_train, y_train, epochs=5, batch_size=10)
```

```
  super().__init__(**kwargs)
2024-08-02 15:38:43.044273: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use ava
ilable CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler
flags.
Epoch 1/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.8771 - loss: 0.4415 - val_accuracy: 0.9640 - val_loss: 0.1314
Epoch 2/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 12s 6ms/step - accuracy: 0.9624 - loss: 0.1275 - val_accuracy: 0.9727 - val_loss: 0.0935
Epoch 3/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.9757 - loss: 0.0794 - val_accuracy: 0.9712 - val_loss: 0.0993
Epoch 4/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 12s 7ms/step - accuracy: 0.9815 - loss: 0.0596 - val_accuracy: 0.9720 - val_loss: 0.0929
Epoch 5/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 12s 7ms/step - accuracy: 0.9855 - loss: 0.0466 - val_accuracy: 0.9757 - val_loss: 0.0842
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9671 - loss: 0.1014
Test accuracy: 0.9729999899864197
```

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.

LIST OF HARDWARE/SOFTWARE REQUIREMENTS:

1. Laptop/Computer with Windows OS
2. Vs code and Browser

Code/Program/Procedure (with comments):

Objective

This script demonstrates how to use TensorFlow/Keras' ImageDataGenerator to apply data augmentation techniques to a sample image from the CIFAR-10 dataset. It generates augmented versions of the image and displays them using matplotlib.

Steps

1. Import Libraries

  - tensorflow: For loading and processing images.

  - ImageDataGenerator: For data augmentation.

  - matplotlib.pyplot: For displaying images.

2. Define Data Augmentation Techniques

  - ImageDataGenerator is configured with:

    - rotation_range=40: Random rotation up to 40 degrees.

    - width_shift_range=0.2: Horizontal shifts up to 20% of the image width.

    - horizontal_flip=True: Random horizontal flipping.

3. Load and Prepare a Sample Image

- Load CIFAR-10 dataset using tf.keras.datasets.cifar10.load_data().
- Normalize the sample image to the range [0, 1].
- Reshape the image to match the input shape expected by ImageDataGenerator.

4. Generate and Display Augmented Images

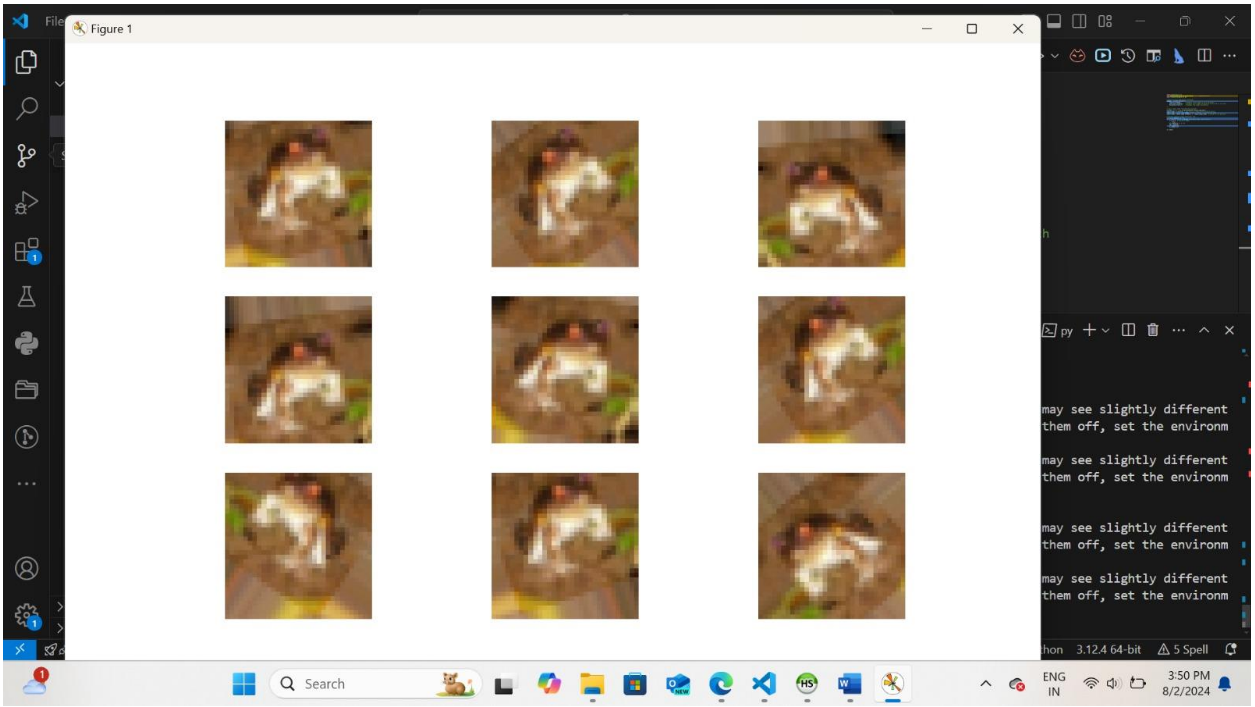
- Generate augmented images using datagen.flow().
- Display the first 9 augmented images in a 3x3 grid.

Usage

1. Run the Script: Execute the script to see the original image and its augmented versions.

2.View Results: The output will display a 3x3 grid of augmented images, showing various transformations applied to the original sample image.

Output :



```
may see slightly different
them off, set the environm

may see slightly different
them off, set the environm

may see slightly different
them off, set the environm

may see slightly different
them off, set the environm
```

Activity - 3

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.

LIST OF HARDWARE/SOFTWARE REQUIREMENTS:

1. Laptop/Computer with Windows OS
2. Vs code and Browser

Code/Program/Procedure (with comments):

Objective

Custom Loss Function in TensorFlow/Keras

Overview

This script demonstrates how to define and use a custom loss function with TensorFlow/Keras. It includes creating a simple neural network model, compiling it with the custom loss function, and training it on random example data.

Steps

1. Import Libraries

   - tensorflow: For building and training the model.

   - numpy: For generating example data.

2. Define Custom Loss Function

   - custom_loss(y_true, y_pred):

     - *MAE (Mean Absolute Error)*: Measures the average magnitude of errors.

     - *L2 Regularization*: Adds a penalty proportional to the square of the predictions to help prevent overfitting.

3. Create and Compile the Model

   - Model Architecture:

      - Input layer with 8 units and ReLU activation.

      - Output layer with 1 unit (no activation function).

   - *Compile*:

      - Optimizer: Adam.

      - Loss function: custom_loss.

4. Generate Example Data

   - Randomly generate x_train and y_train for demonstration purposes.

5. Train the Model

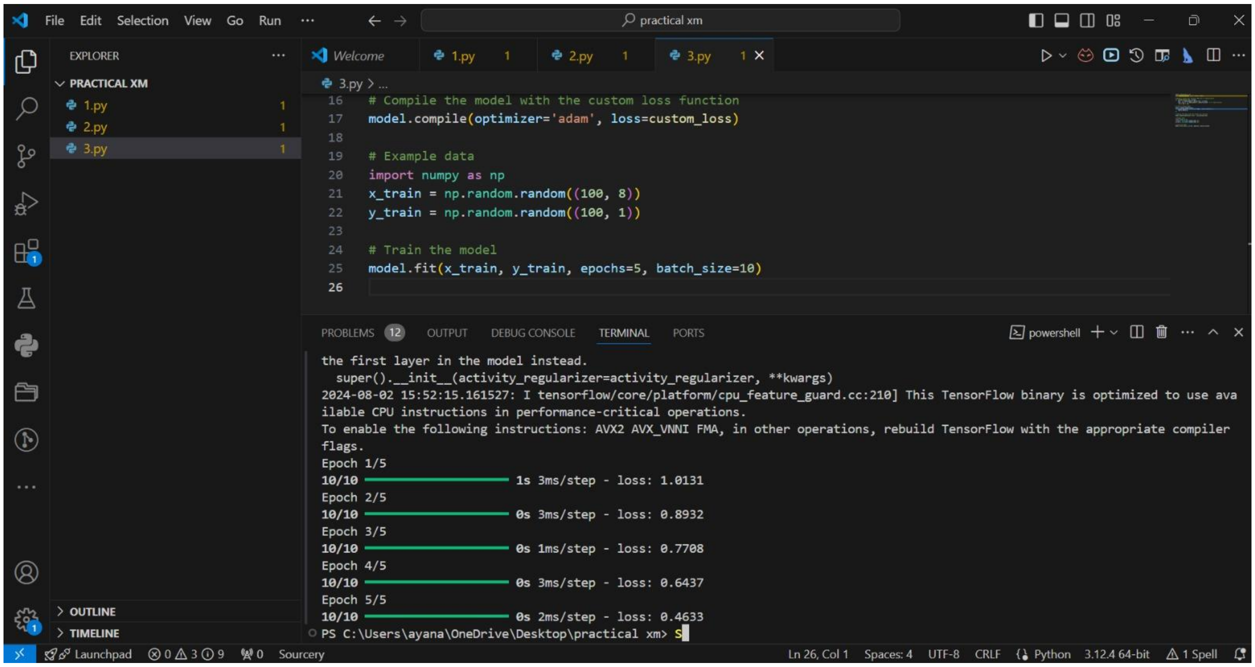
   - Fit the model on the example data for 5 epochs with a batch size of 10.

Usage

1. Run the Script: Execute the script to train the model using the custom loss function.

2. *Results*: The model will train on the synthetic data and you can observe the effect of the custom loss function during training.

Output :



```python
# Compile the model with the custom loss function
model.compile(optimizer='adam', loss=custom_loss)

# Example data
import numpy as np
x_train = np.random.random((100, 8))
y_train = np.random.random((100, 1))

# Train the model
model.fit(x_train, y_train, epochs=5, batch_size=10)

```

```
the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
2024-08-02 15:52:15.161527: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use ava
ilable CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler
flags.
Epoch 1/5
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 1.0131
Epoch 2/5
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.8932
Epoch 3/5
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.7708
Epoch 4/5
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.6437
Epoch 5/5
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.4633
PS C:\Users\ayana\OneDrive\Desktop\practical xm> S
```