



# PRACTICAL TECHNICAL ASSESSMENT

SHARON P. A

# Activity 1

## Building a Simple Neural Network

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

- ✚ Personal computer/laptop
- ✚ Google Collab

## Procedure

1. Create a folder name as exam
2. Open vs code
3. Create a py file in that folder
4. Write the code in that file

### 1. Import Necessary Libraries

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical
```

## 2. Load and Preprocess the Data

```
# Load the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize the pixel values (0-255) to the range (0-1)
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# One-hot encode the labels
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

## 3. Build the Neural Network Model

```
# Build the neural network model
model = Sequential([
    Flatten(input_shape=(28, 28)), # Flattens the 2D image array into a 1D array
    Dense(128, activation='relu'), # Hidden layer with 128 neurons and ReLU activation
    Dense(10, activation='softmax') # Output layer with 10 neurons and softmax activation
])
```

## 4. Compile the Model

```
# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

## 5. Train and evaluate the Model

```
# Train the model
model.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))

# Evaluate the model
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_accuracy}')
```

## 6. Make predictions

```
# Make predictions (optional)
predictions = model.predict(x_test)
print(f'Predicted label for the first test sample: {np.argmax(predictions[0])}')

```

## 5.output

```
PS C:\Users\USER\Desktop\exam> py q1.py
2024-08-02 15:25:08.828951: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results
rn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-08-02 15:25:13.168426: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results
rn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
C:\Users\USER\AppData\Roaming\Python\Python311\site-packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an `input_shape`/
Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
2024-08-02 15:25:24.818302: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instruct
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Epoch 1/5
1875/1875 ————— 7s 3ms/step - accuracy: 0.8795 - loss: 0.4331 - val_accuracy: 0.9577 - val_loss: 0.1387
Epoch 2/5
1875/1875 ————— 5s 3ms/step - accuracy: 0.9638 - loss: 0.1257 - val_accuracy: 0.9701 - val_loss: 0.0972
Epoch 3/5
1875/1875 ————— 5s 3ms/step - accuracy: 0.9762 - loss: 0.0805 - val_accuracy: 0.9726 - val_loss: 0.0855
Epoch 4/5
1875/1875 ————— 5s 3ms/step - accuracy: 0.9832 - loss: 0.0551 - val_accuracy: 0.9746 - val_loss: 0.0810
Epoch 5/5
1875/1875 ————— 5s 3ms/step - accuracy: 0.9861 - loss: 0.0454 - val_accuracy: 0.9741 - val_loss: 0.0809
313/313 ————— 1s 2ms/step - accuracy: 0.9697 - loss: 0.0947
Test accuracy: 0.9740999937057495
313/313 ————— 1s 2ms/step
Predicted label for the first test sample: 7
PS C:\Users\USER\Desktop\exam> |
```

# Activity 2

## Data Augmentation

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

- + Personal computer/laptop
- + Google Collab

## Procedure

### **Import Necessary Libraries**

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import os
```

### **Define path**

```
# Define the path to your image
image_path = 'C:/Users/USER/Desktop/exam/images.jpg' # Replace with your image path
```

## check the file

```
# Check if the file exists
if not os.path.isfile(image_path):
    raise FileNotFoundError(f"Image file not found at path: {image_path}")
```

## Create an instance of imageDataGenerator with multiple augmentation

```
# Create an instance of ImageDataGenerator with multiple augmentation techniques
datagen = ImageDataGenerator(
    rotation_range=40,      # Randomly rotate images by up to 40 degrees
    width_shift_range=0.2,  # Randomly shift images horizontally by up to 20% of the width
    height_shift_range=0.2, # Randomly shift images vertically by up to 20% of the height
    shear_range=0.2,       # Randomly apply shearing transformations
    zoom_range=0.2,        # Randomly zoom into images by up to 20%
    horizontal_flip=True,   # Randomly flip images horizontally
    fill_mode='nearest'     # Fill in pixels that are newly created during transformations
)
```

### 1. load and preprocess the image

```
# Load and preprocess the image
image = tf.keras.preprocessing.image.load_img(image_path)
image = tf.keras.preprocessing.image.img_to_array(image)
image = np.expand_dims(image, axis=0) # Convert image to a batch of size 1
```

### 2. apply the argumentaion

```
# Apply augmentations
augmented_images = datagen.flow(image, batch_size=1)
```

### 3. plot the original and argumental images

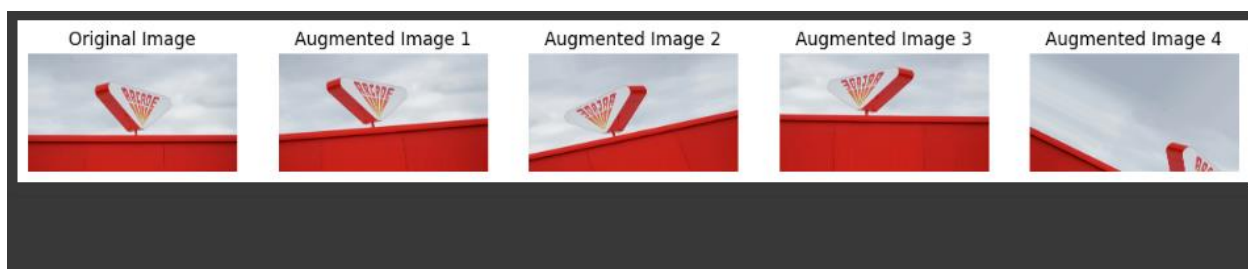
```
# Plot the original and augmented images
plt.figure(figsize=(15, 15))

# Plot the original image
plt.subplot(1, 5, 1)
plt.imshow(image[0].astype('uint8'))
plt.title('Original Image')
plt.axis('off')

# Plot a few augmented images
for i in range(4):
    plt.subplot(1, 5, i + 2)
    batch = next(augmented_images) # Use next() to get the next batch
    augmented_image = batch[0].astype('uint8')
    plt.imshow(augmented_image)
    plt.title(f'Augmented Image {i+1}')
    plt.axis('off')

plt.show()
```

## OUTPUT



# Activity 3

## Custom Loss Function

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

- ✚ Personal computer/laptop
- ✚ Google Collab

## Procedure

### Import Necessary Libraries

```
import tensorflow as tf
from tensorflow.keras.losses import Loss
```

### Function

```
class CustomLoss(Loss):
    def __init__(self, alpha=0.1, **kwargs):
        super().__init__(**kwargs)
        self.alpha = alpha # Regularization strength

    def call(self, y_true, y_pred):
        # Mean Squared Error
        mse = tf.reduce_mean(tf.square(y_true - y_pred))

        # Regularization Term: Penalizes predictions deviating from the mean of y_true
        y_true_mean = tf.reduce_mean(y_true)
        regularization_term = tf.reduce_mean(tf.square(y_pred - y_true_mean))

        # Combine MSE with the regularization term
        loss = mse + self.alpha * regularization_term
        return loss
```



## example usage

```
# Example Usage
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, activation='relu', input_shape=(5,)),
    tf.keras.layers.Dense(1)
])

model.compile(optimizer='adam', loss=CustomLoss(alpha=0.5))
```

## Output

```
PS C:\Users\USER\Desktop\exam> py q3.py
2024-08-02 16:09:36.752481: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
2024-08-02 16:09:38.485330: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
C:\Users\USER\AppData\Roaming\Python\Python311\site-packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an 'input_shape'/'input_dim' argument to a layer. When using Sequential models, prefer using an 'Input(shape)' object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
2024-08-02 16:09:42.181044: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
```