Close

```
Title

Deep Learning Model for Image Classification using TensorFlow

1. Problem Statement

Image classification is an important task in computer vision whe
machines automatically identify objects or patterns from images.
real-world applications, image classification is used in healthc
security systems, handwriting recognition, and automation.

In this project, we implement a Deep Learning model using Tensor
classify handwritten digit images using the MNIST dataset.

2. Objective

Build a functional Deep Learning model

Perform image classification

Train and test the model

Visualize model performance using graphs

Evaluate accuracy of predictions
3. Conclusion

In this project, we successfully built a Deep Learning Image
Classification model using TensorFlow. The model was trained on
handwritten digit images and achieved good accuracy. Visualizati
helped analyze model performance and learning behavior.
```

Title

Deep Learning Model for Image Classification using TensorFlow

1. Problem Statement

Image classification is an important task in computer vision where machines automatically identify objects or patterns from images. In real-world applications, image classification is used in healthcare, security systems, handwriting recognition, and automation.

In this project, we implement a Deep Learning model using TensorFlow to classify handwritten digit images using the MNIST dataset.

2. Objective

Build a functional Deep Learning model

Perform image classification

Train and test the model

Visualize model performance using graphs

Evaluate accuracy of predictions 3. Conclusion

In this project, we successfully built a Deep Learning Image Classification model using TensorFlow. The model was trained on handwritten digit images and achieved good accuracy. Visualization graphs helped analyze model performance and learning behavior.

```
import tensorflow as tf
print(tf.__version__)
```

```
2.19.0
```

```
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import numpy as np
```

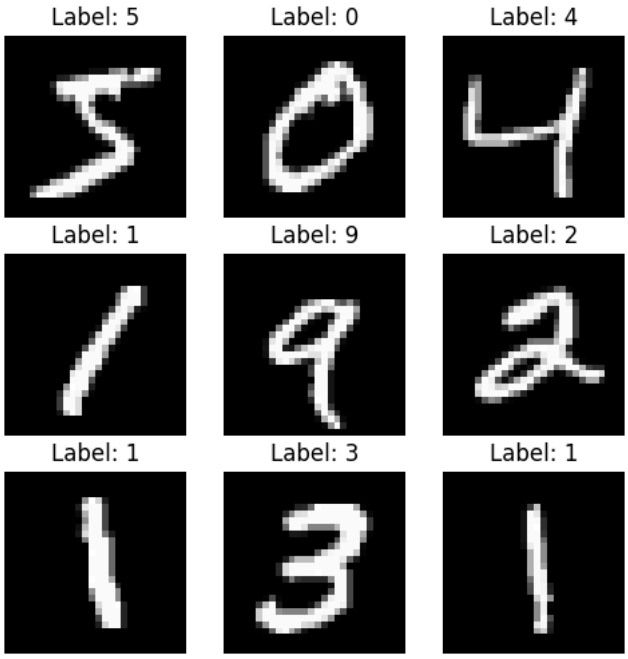
```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
x_train = x_train / 255.0
x_test = x_test / 255.0

x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)
```

```
plt.figure(figsize=(6,6))
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(x_train[i].reshape(28,28), cmap='gray')
    plt.title(f"Label: {y_train[i]}")
    plt.axis('off')
plt.show()
```




```
model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

```
history = model.fit(
    x_train, y_train,
    epochs=5,
    validation_data=(x_test, y_test)
)
```

```
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 54s 28ms/step - accuracy: 0.5812 - loss: 1.2110 - val_accuracy: 0.9057 - val_loss: 0.2995
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 52s 28ms/step - accuracy: 0.9121 - loss: 0.2854 - val_accuracy: 0.9383 - val_loss: 0.1997
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 52s 28ms/step - accuracy: 0.9409 - loss: 0.1958 - val_accuracy: 0.9597 - val_loss: 0.1351
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 52s 28ms/step - accuracy: 0.9561 - loss: 0.1448 - val_accuracy: 0.9669 - val_loss: 0.1086
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 82s 28ms/step - accuracy: 0.9658 - loss: 0.1108 - val_accuracy: 0.9745 - val_loss: 0.0839
```

```
plt.figure(figsize=(12,4))

plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
```

```
plt.legend()
plt.title('Accuracy')

plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Loss')

plt.show()
```