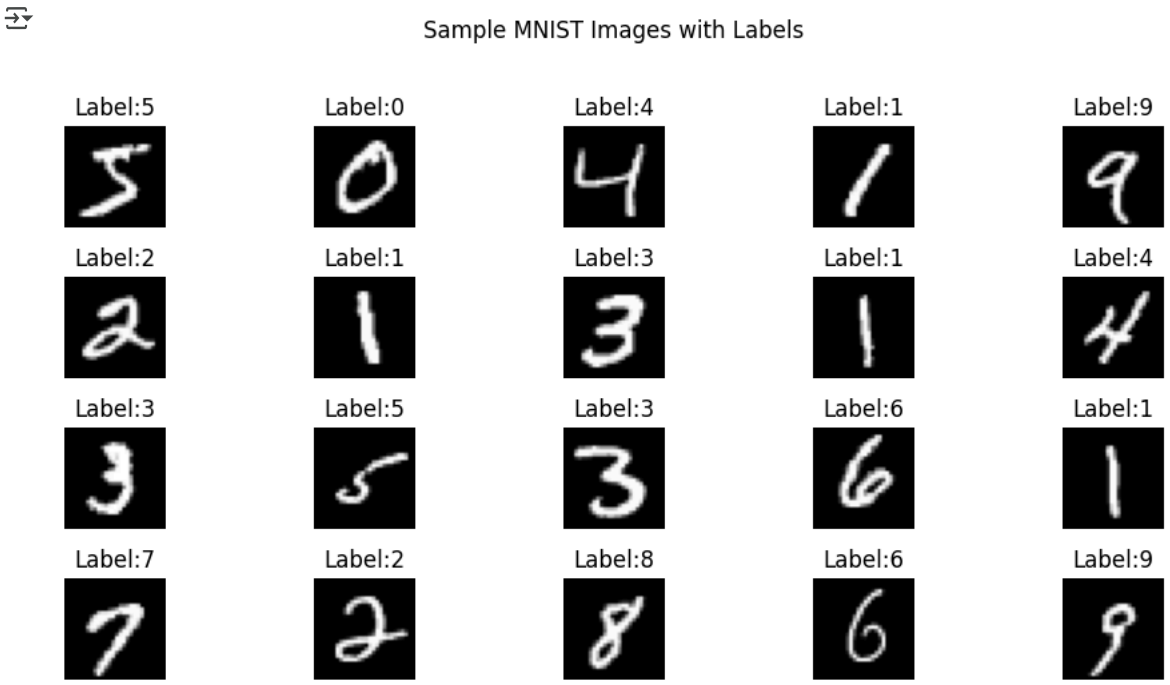
```
#TensorFlow 2.x MNIST Classification with DNN (Only Dense Layers)
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import numpy as np


# Load the MNIST dataset
(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.mnist.load_data()

# Preprocess the data
# - Reshape the images from (60000, 28, 28) to (60000, 784)
# - Normalize the pixel values to range 0–1
train_images = train_images.reshape((60000, 28 * 28)) / 255.0
test_images = test_images.reshape((10000, 28 * 28)) / 255.0
```

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

```
plt.figure(figsize=(10,5))
for i in range(20):
    plt.subplot(4,5,i+1)
    plt.imshow(train_images[i].reshape(28,28),cmap='gray')
    plt.title(f"Label:{train_labels[i]}")
    plt.axis('off')

plt.suptitle("Sample MNIST Images with Labels", y=1.02)
plt.tight_layout()
plt.show()
```




```
model = models.Sequential([
    layers.Dense(1024, activation='relu', input_shape=(28*28,)),
    layers.Dropout(0.3),
    layers.Dense(128, activation='sigmoid'),
    layers.Dropout(0.3),
    layers.Dense(10, activation='softmax')
])


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


from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(
    monitor='val_loss',       # Watch validation loss
    patience=3,               # Wait 3 epochs with no improvement
    restore_best_weights=True  # Go back to the best weights
)

history = model.fit(train_images, train_labels,
```

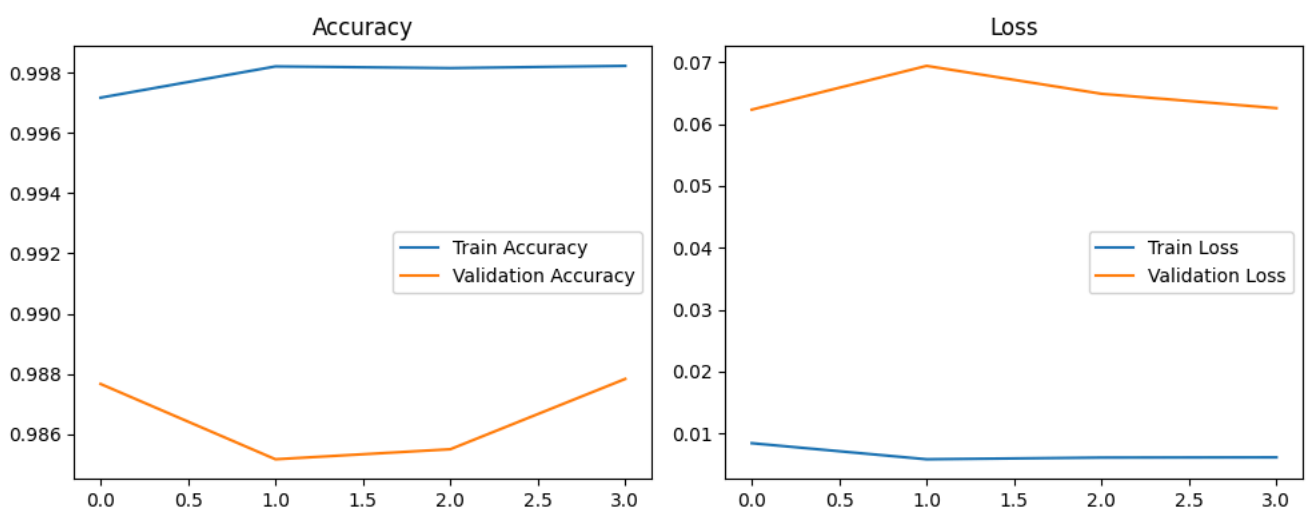
```
                    epochs=30,
                    batch_size=128,
                    validation_split=0.1,
                    callbacks=[early_stop])
```

```
Epoch 1/30
422/422 ━━━━━━━━━━━━━━━━━━━━ 10s 23ms/step - accuracy: 0.9969 - loss: 0.0094 - val_accuracy: 0.9877 - val_loss: 0.0623
Epoch 2/30
422/422 ━━━━━━━━━━━━━━━━━━━━ 10s 23ms/step - accuracy: 0.9981 - loss: 0.0061 - val_accuracy: 0.9852 - val_loss: 0.0694
Epoch 3/30
422/422 ━━━━━━━━━━━━━━━━━━━━ 10s 24ms/step - accuracy: 0.9980 - loss: 0.0064 - val_accuracy: 0.9855 - val_loss: 0.0649
Epoch 4/30
422/422 ━━━━━━━━━━━━━━━━━━━━ 9s 21ms/step - accuracy: 0.9985 - loss: 0.0053 - val_accuracy: 0.9878 - val_loss: 0.0626
```

```
test_loss,test_acc=model.evaluate(test_images,test_labels)
print(f"\nTest Accuracy is:{test_acc:.4f}")
```

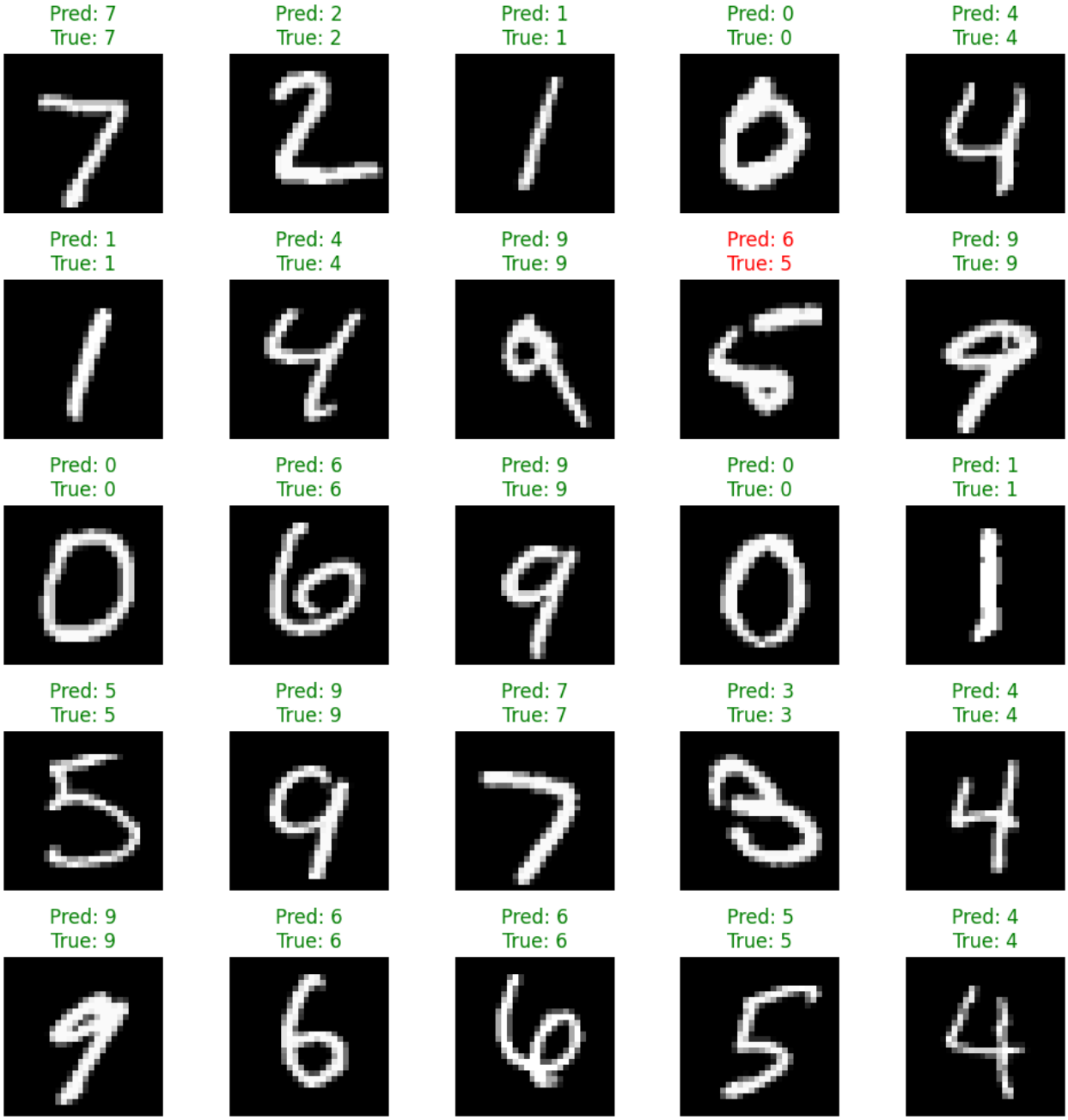
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9830 - loss: 0.0751

Test Accuracy is:0.9859
```

```
# Plot training history
plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy')
plt.legend()

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




```
predictions = model.predict(test_images)
plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i+1)
    plt.imshow(test_images[i].reshape(28, 28), cmap='gray')
    pred_label = np.argmax(predictions[i])
    true_label = test_labels[i]
    color = 'green' if pred_label == true_label else 'red'
    plt.title(f"Pred: {pred_label}\nTrue: {true_label}", color=color)
    plt.axis('off')
plt.tight_layout()
plt.show()
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
```



Start coding or generate with AI.