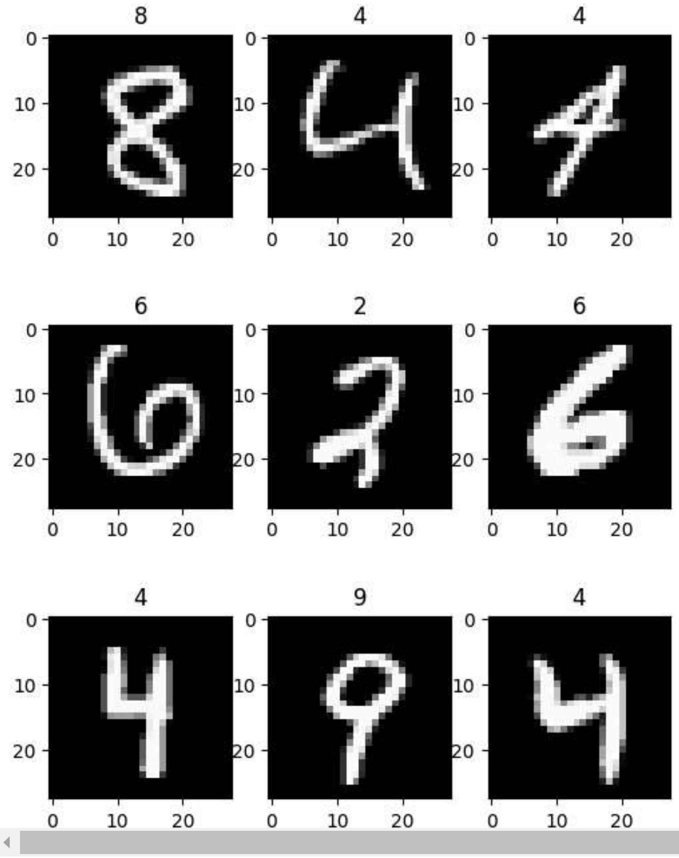
```
import os
import time
import numpy as np
import random
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from sklearn.metrics import confusion_matrix, classification_report
from matplotlib import pyplot as plt


model_directory = 'models'

class RandomIntegers:
    def __init__(self):
        pass

    def generate(self, n, length):
        random_integers = random.sample(range(length), n)
        return random_integers

(x_train, y_train), (x_test, y_test) = mnist.load_data()

def display_images():
    random_integers = RandomIntegers().generate(9, len(x_train))
    plt.figure(figsize=(6, 8))
    counter = 0
    for i in random_integers:
        plt.subplot(330 + 1 + counter)
        counter += 1
        plt.imshow(x_train[i], cmap='gray')
        plt.title(str(y_train[i]))
    plt.show()

display_images()
```



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

x_train = x_train.reshape(x_train.shape[0],28,28,1)
x_test = x_test.reshape(x_test.shape[0],28,28,1)
```

```
model = 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(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

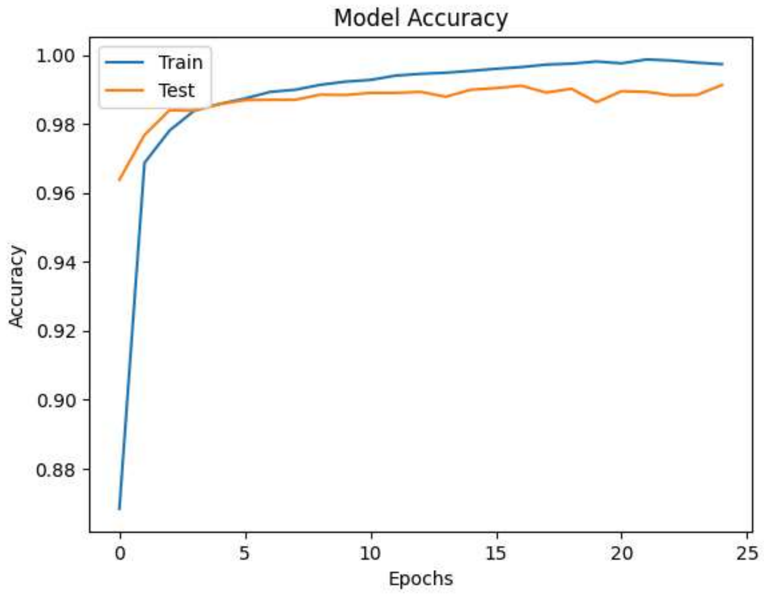
```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
model.compile(
    optimizer=tf.keras.optimizers.Adam(0.001),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=[tf.keras.metrics.SparseCategoricalAccuracy()]
)

start_time = time.time()
history = model.fit(
    x_train,
    y_train,
    epochs=25,
    batch_size=512,
    validation_data=(x_test, y_test)
)
end_time = time.time()
time_taken = end_time - start_time
print("Time taken for training: ", time_taken, "seconds")
```

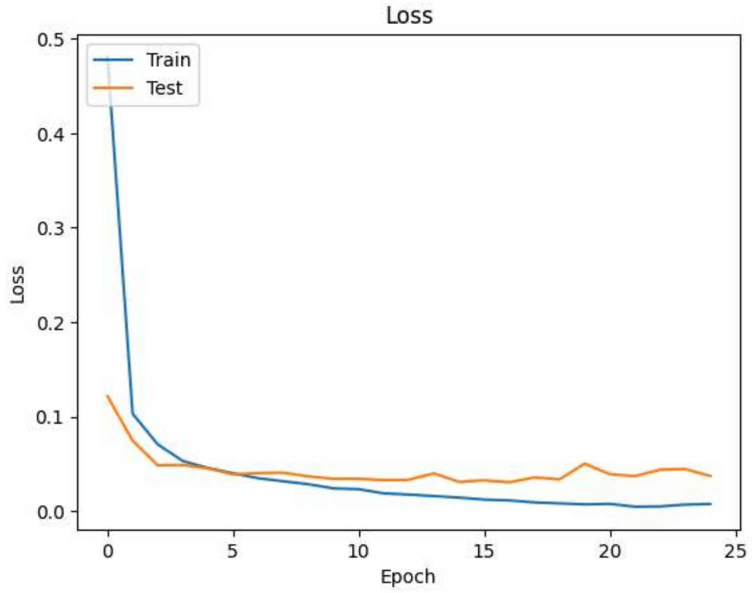
```
Epoch 1/25
/usr/local/lib/python3.10/dist-packages/keras/src/backend/tensorflow/nn.py:609: UserWarning: "`sparse_categorical_crossentropy` receivec
  output, from_logits = _get_logits(
118/118 ━━━━━━━━━━━━━━━━━━━━ 52s 432ms/step - loss: 0.9703 - sparse_categorical_accuracy: 0.7368 - val_loss: 0.1216 - val_sparse_categor
Epoch 2/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 82s 435ms/step - loss: 0.1141 - sparse_categorical_accuracy: 0.9656 - val_loss: 0.0749 - val_sparse_categor
Epoch 3/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 49s 419ms/step - loss: 0.0780 - sparse_categorical_accuracy: 0.9749 - val_loss: 0.0486 - val_sparse_categor
Epoch 4/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 82s 421ms/step - loss: 0.0550 - sparse_categorical_accuracy: 0.9833 - val_loss: 0.0490 - val_sparse_categor
Epoch 5/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 83s 429ms/step - loss: 0.0488 - sparse_categorical_accuracy: 0.9850 - val_loss: 0.0453 - val_sparse_categor
Epoch 6/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 82s 429ms/step - loss: 0.0416 - sparse_categorical_accuracy: 0.9868 - val_loss: 0.0390 - val_sparse_categor
Epoch 7/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 82s 425ms/step - loss: 0.0354 - sparse_categorical_accuracy: 0.9893 - val_loss: 0.0404 - val_sparse_categor
Epoch 8/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 50s 422ms/step - loss: 0.0345 - sparse_categorical_accuracy: 0.9896 - val_loss: 0.0408 - val_sparse_categor
Epoch 9/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 84s 437ms/step - loss: 0.0278 - sparse_categorical_accuracy: 0.9914 - val_loss: 0.0369 - val_sparse_categor
Epoch 10/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 83s 447ms/step - loss: 0.0251 - sparse_categorical_accuracy: 0.9919 - val_loss: 0.0343 - val_sparse_categor
Epoch 11/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 49s 414ms/step - loss: 0.0231 - sparse_categorical_accuracy: 0.9928 - val_loss: 0.0345 - val_sparse_categor
Epoch 12/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 51s 432ms/step - loss: 0.0192 - sparse_categorical_accuracy: 0.9937 - val_loss: 0.0330 - val_sparse_categor
Epoch 13/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 82s 431ms/step - loss: 0.0183 - sparse_categorical_accuracy: 0.9942 - val_loss: 0.0334 - val_sparse_categor
Epoch 14/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 50s 423ms/step - loss: 0.0168 - sparse_categorical_accuracy: 0.9948 - val_loss: 0.0401 - val_sparse_categor
Epoch 15/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 83s 433ms/step - loss: 0.0151 - sparse_categorical_accuracy: 0.9953 - val_loss: 0.0311 - val_sparse_categor
Epoch 16/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 82s 432ms/step - loss: 0.0117 - sparse_categorical_accuracy: 0.9965 - val_loss: 0.0328 - val_sparse_categor
Epoch 17/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 82s 431ms/step - loss: 0.0104 - sparse_categorical_accuracy: 0.9971 - val_loss: 0.0308 - val_sparse_categor
Epoch 18/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 82s 432ms/step - loss: 0.0088 - sparse_categorical_accuracy: 0.9975 - val_loss: 0.0357 - val_sparse_categor
Epoch 19/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 82s 434ms/step - loss: 0.0078 - sparse_categorical_accuracy: 0.9975 - val_loss: 0.0338 - val_sparse_categor
Epoch 20/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 49s 418ms/step - loss: 0.0062 - sparse_categorical_accuracy: 0.9985 - val_loss: 0.0504 - val_sparse_categor
Epoch 21/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 86s 453ms/step - loss: 0.0092 - sparse_categorical_accuracy: 0.9973 - val_loss: 0.0393 - val_sparse_categor
Epoch 22/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 79s 428ms/step - loss: 0.0051 - sparse_categorical_accuracy: 0.9985 - val_loss: 0.0371 - val_sparse_categor
Epoch 23/25
```

```
118/118 ━━━━━━━━━━━━━━━━━━━━ 83s 433ms/step - loss: 0.0040 - sparse_categorical_accuracy: 0.9988 - val_loss: 0.0441 - val_sparse_categor
Epoch 24/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 81s 425ms/step - loss: 0.0062 - sparse_categorical_accuracy: 0.9980 - val_loss: 0.0447 - val_sparse_categor
Epoch 25/25
118/118 ━━━━━━━━━━━━━━━━━━━━ 51s 431ms/step - loss: 0.0068 - sparse_categorical_accuracy: 0.9978 - val_loss: 0.0374 - val_sparse_categor
Time taken for training:  1801.0017359256744 seconds
```

```
plt.plot(history.history['sparse_categorical_accuracy'])
plt.plot(history.history['val_sparse_categorical_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



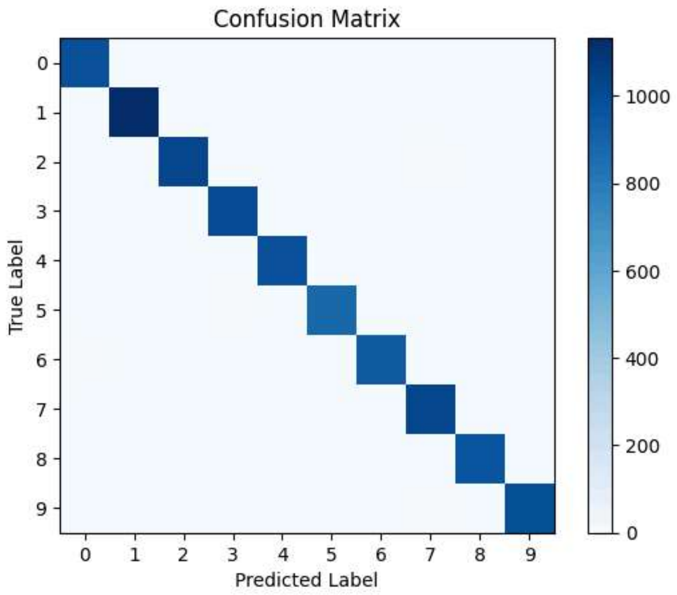
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



```
if not os.path.exists(model_directory):
    os.makedirs(model_directory)
model_path = os.path.join(model_directory, 'model_mnist_cnn.h5')
model.save(model_path)
model.summary()

true_labels = y_test
predicted_labels = np.argmax(model.predict(x_test), axis=-1)
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is consi
Model: "sequential_1"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d_2 (Conv2D)                    │ (None, 26, 26, 32)          │             320 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_2 (MaxPooling2D)       │ (None, 13, 13, 32)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_3 (Conv2D)                    │ (None, 11, 11, 64)          │          18,496 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_3 (MaxPooling2D)       │ (None, 5, 5, 64)            │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten_1 (Flatten)                  │ (None, 1600)                │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_2 (Dense)                      │ (None, 64)                  │         102,464 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_3 (Dense)                      │ (None, 10)                  │             650 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 365,792 (1.40 MB)
 Trainable params: 121,930 (476.29 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 243,862 (952.59 KB)
313/313 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step
```

```
cm = confusion_matrix(true_labels, predicted_labels)

plt.imshow(cm, cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(np.arange(10))
plt.yticks(np.arange(10))
plt.show()
```



```
# Generate and print classification report
report = classification_report(true_labels, predicted_labels)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       1.00      1.00      1.00      1135
           2       0.99      0.99      0.99      1032
           3       0.99      1.00      0.99      1010
           4       0.99      0.99      0.99       982
           5       0.99      0.99      0.99       892
           6       1.00      0.98      0.99       958
           7       0.99      1.00      0.99      1028
           8       0.99      0.99      0.99       974
           9       0.99      0.99      0.99      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```