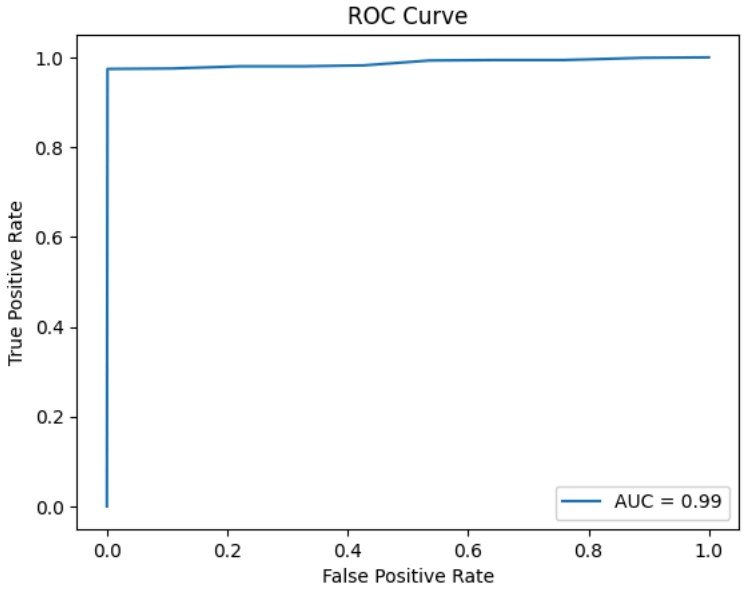
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
import tensorflow as tf
from tensorflow.keras import layers, models

# Image Compression
input_path = 'input.jpeg'
output_path_jpeg = 'compressed.jpg'
output_path_png = 'compressed.png'
quality_jpeg = 50
quality_png = 9


img = cv2.imread(input_path)
cv2.imwrite(output_path_jpeg, img, [int(cv2.IMWRITE_JPEG_QUALITY), quality_jpeg])
cv2.imwrite(output_path_png, img, [int(cv2.IMWRITE_PNG_COMPRESSION), quality_png])

# Display Original and Compressed Images
original = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
jpeg_compressed = cv2.cvtColor(cv2.imread(output_path_jpeg), cv2.COLOR_BGR2RGB)
png_compressed = cv2.cvtColor(cv2.imread(output_path_png), cv2.COLOR_BGR2RGB)

plt.figure(figsize=(12, 4))
plt.subplot(1, 3, 1)
plt.imshow(original)
plt.title("Original Image")
plt.axis("off")

plt.subplot(1, 3, 2)
plt.imshow(jpeg_compressed)
plt.title("JPEG Compressed Image")
plt.axis("off")

plt.subplot(1, 3, 3)
plt.imshow(png_compressed)
plt.title("PNG Compressed Image")
plt.axis("off")

plt.show()
```


```
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc, precision_recall_fscore_support, accuracy_score
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
x_train = x_train[..., np.newaxis]
x_test = x_test[..., np.newaxis]

# CNN Model for MNIST
model_mnist = 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.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

model_mnist.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model_mnist.fit(x_train, y_train, epochs=10, batch_size=128, validation_data=(x_test, y_test))
y_pred = model_mnist.predict(x_test).argmax(axis=1)

accuracy = accuracy_score(y_test, y_pred)
precision, recall, f1, _ = precision_recall_fscore_support(y_test, y_pred, average='macro')
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
```

```
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

fpr, tpr, _ = roc_curve(y_test, y_pred, pos_label=y_test.max())
auc_score = auc(fpr, tpr)
plt.plot(fpr, tpr, label=f'AUC = {auc_score:.2f}')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```

```
Epoch 1/2
469/469 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 0.8370 - loss: 0.5647 - val_accuracy: 0.9825 - val_loss: 0.0576
Epoch 2/2
469/469 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9805 - loss: 0.0617 - val_accuracy: 0.9860 - val_loss: 0.0437
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
Accuracy: 0.9860
Precision: 0.9864
Recall: 0.9859
F1-Score: 0.9861
Confusion Matrix:
 [[ 977    0    0    0    0    0    1    1    1    0]
 [   0 1135    0    0    0    0    0    0    0    0]
 [   2   13 1004    0    4    0    2    4    3    0]
 [   0    1    2  997    0    3    0    5    2    0]
 [   0    1    0    0  979    0    0    0    0    2]
 [   1    0    0    5    1  877    4    1    2    1]
 [   4    4    0    0    2    1  947    0    0    0]
 [   0   11    4    0    0    0    0 1009    1    3]
 [   4    3    2    1    3    0    3    3  952    3]
 [   1    5    0    1   11    2    0    5    1  983]]
```




```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

# CNN Model for CIFAR-10
model_cifar10 = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

model_cifar10.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model_cifar10.fit(x_train, y_train, epochs=50, batch_size=64, validation_data=(x_test, y_test))
y_pred = model_cifar10.predict(x_test).argmax(axis=1)

accuracy = accuracy_score(y_test, y_pred)
precision, recall, f1, _ = precision_recall_fscore_support(y_test, y_pred, average='macro')
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
```

```
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

fpr, tpr, _ = roc_curve(y_test, y_pred, pos_label=y_test.max())
auc_score = auc(fpr, tpr)
plt.plot(fpr, tpr, label=f'AUC = {auc_score:.2f}')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 7s 7ms/step - accuracy: 0.3343 - loss: 1.7919 - val_accuracy: 0.5374 - val_loss: 1.2826
Epoch 2/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.5545 - loss: 1.2494 - val_accuracy: 0.5803 - val_loss: 1.1846
Epoch 3/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.6212 - loss: 1.0775 - val_accuracy: 0.6081 - val_loss: 1.1061
Epoch 4/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.6599 - loss: 0.9744 - val_accuracy: 0.6580 - val_loss: 0.9803
Epoch 5/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.6841 - loss: 0.9018 - val_accuracy: 0.6718 - val_loss: 0.9412
Epoch 6/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.7036 - loss: 0.8484 - val_accuracy: 0.6720 - val_loss: 0.9329
Epoch 7/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.7203 - loss: 0.7984 - val_accuracy: 0.6846 - val_loss: 0.9136
Epoch 8/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.7386 - loss: 0.7507 - val_accuracy: 0.6710 - val_loss: 0.9612
Epoch 9/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - accuracy: 0.7489 - loss: 0.7211 - val_accuracy: 0.6686 - val_loss: 0.9544
Epoch 10/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.7673 - loss: 0.6704 - val_accuracy: 0.7039 - val_loss: 0.8743
Epoch 11/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.7704 - loss: 0.6468 - val_accuracy: 0.7042 - val_loss: 0.8701
Epoch 12/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.7905 - loss: 0.5996 - val_accuracy: 0.7143 - val_loss: 0.8383
Epoch 13/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.7975 - loss: 0.5748 - val_accuracy: 0.6950 - val_loss: 0.9251
Epoch 14/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - accuracy: 0.8066 - loss: 0.5537 - val_accuracy: 0.7119 - val_loss: 0.8620
Epoch 15/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8148 - loss: 0.5255 - val_accuracy: 0.7188 - val_loss: 0.8581
Epoch 16/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8241 - loss: 0.4992 - val_accuracy: 0.7150 - val_loss: 0.8918
Epoch 17/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8387 - loss: 0.4600 - val_accuracy: 0.7204 - val_loss: 0.8923
Epoch 18/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.8436 - loss: 0.4423 - val_accuracy: 0.7131 - val_loss: 0.9097
Epoch 19/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.8552 - loss: 0.4137 - val_accuracy: 0.7065 - val_loss: 1.0173
Epoch 20/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - accuracy: 0.8629 - loss: 0.3868 - val_accuracy: 0.7043 - val_loss: 1.0175
Epoch 21/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.8714 - loss: 0.3661 - val_accuracy: 0.7131 - val_loss: 1.0213
Epoch 22/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.8787 - loss: 0.3426 - val_accuracy: 0.7069 - val_loss: 1.0671
Epoch 23/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - accuracy: 0.8852 - loss: 0.3268 - val_accuracy: 0.7137 - val_loss: 1.0770
Epoch 24/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.8927 - loss: 0.3043 - val_accuracy: 0.7019 - val_loss: 1.1577
Epoch 25/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - accuracy: 0.8986 - loss: 0.2881 - val_accuracy: 0.7019 - val_loss: 1.1985
Epoch 26/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9021 - loss: 0.2759 - val_accuracy: 0.6905 - val_loss: 1.2338
Epoch 27/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9083 - loss: 0.2615 - val_accuracy: 0.6972 - val_loss: 1.2770
Epoch 28/50
```

Start coding or generate with AI.

```
Epoch 30/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 6s 5ms/step - accuracy: 0.9287 - loss: 0.2012 - val_accuracy: 0.6956 - val_loss: 1.4044
Epoch 31/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9286 - loss: 0.1994 - val_accuracy: 0.6905 - val_loss: 1.4807
Epoch 32/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9324 - loss: 0.1906 - val_accuracy: 0.6979 - val_loss: 1.5195
Epoch 33/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - accuracy: 0.9379 - loss: 0.1760 - val_accuracy: 0.6946 - val_loss: 1.5482
Epoch 34/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.9368 - loss: 0.1823 - val_accuracy: 0.6937 - val_loss: 1.6470
Epoch 35/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9444 - loss: 0.1549 - val_accuracy: 0.6882 - val_loss: 1.7196
Epoch 36/50
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 6s 8ms/step - accuracy: 0.9449 - loss: 0.1563 - val_accuracy: 0.6909 - val_loss: 1.6943
Epoch 37/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.9458 - loss: 0.1515 - val_accuracy: 0.6944 - val_loss: 1.7636
Epoch 38/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 3s 4ms/step - accuracy: 0.9447 - loss: 0.1506 - val_accuracy: 0.6821 - val_loss: 1.7891
Epoch 39/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 4s 4ms/step - accuracy: 0.9481 - loss: 0.1448 - val_accuracy: 0.6899 - val_loss: 1.8759
Epoch 40/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.9562 - loss: 0.1242 - val_accuracy: 0.6846 - val_loss: 1.9507
Epoch 41/50
```