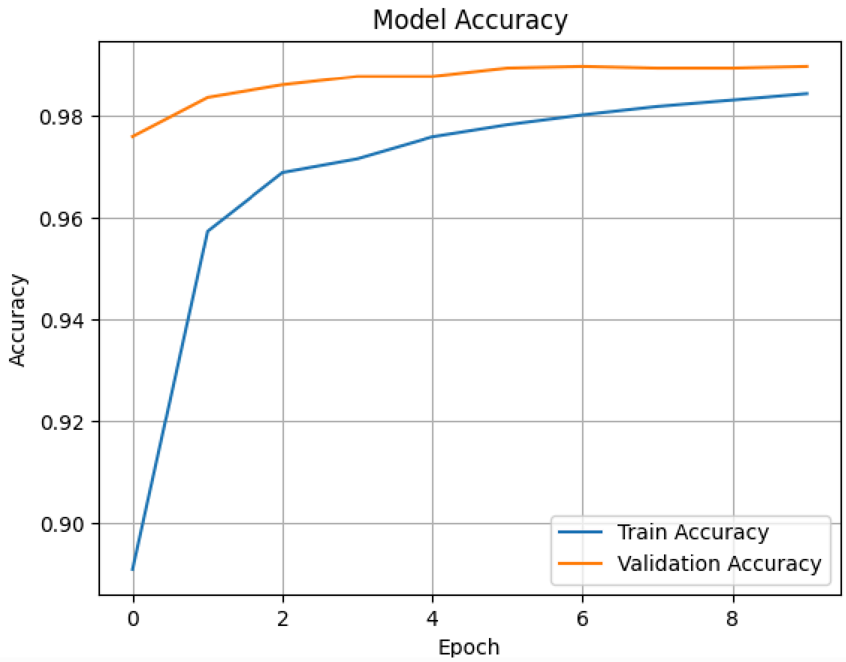
```
# Handwritten Digit Recognition in Google Colab

# 1. Setup Environment and Imports
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.datasets import mnist

# 2. Load MNIST Dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
print(f"Train shape: {X_train.shape}, Test shape: {X_test.shape}")

# 3. Data Preprocessing

# 3.1 Check for Missing Values
assert not np.isnan(X_train).any(), "Missing values in X_train"
assert not np.isnan(X_test).any(), "Missing values in X_test"

# 3.2 Remove Duplicate Records
X_train_flat = X_train.reshape(X_train.shape[0], -1)
df_train = pd.DataFrame(X_train_flat)
df_train['label'] = y_train
df_train_nodup = df_train.drop_duplicates()
print(f"Removed {len(df_train) - len(df_train_nodup)} duplicate rows")
y_train = df_train_nodup['label'].values
X_train = df_train_nodup.drop('label', axis=1).values.reshape(-1, 28, 28)

# 3.3 Convert Types and Normalize
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0

# 3.4 Encode Labels
y_train_cat = to_categorical(y_train, 10)
y_test_cat = to_categorical(y_test, 10)

# 4. Reshape Data for CNN
X_train = X_train.reshape(-1, 28, 28, 1)
X_test = X_test.reshape(-1, 28, 28, 1)

# 5. Build CNN Model
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    MaxPooling2D(2,2),
    Dropout(0.25),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

# 6. Train the Model
history = model.fit(X_train, y_train_cat, epochs=10, batch_size=128, validation_split=0.1)

# 7. Evaluate and Visualize
test_loss, test_acc = model.evaluate(X_test, y_test_cat)
print(f"\nTest Accuracy: {test_acc:.4f}")

plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title("Model Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(True)
plt.show()
```

```
Train shape: (60000, 28, 28), Test shape: (10000, 28, 28)
Removed 0 duplicate rows
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential_1"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d_1 (Conv2D)                │ (None, 26, 26, 32)     │           320 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)   │ (None, 13, 13, 32)     │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_2 (Dropout)              │ (None, 13, 13, 32)     │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten_1 (Flatten)              │ (None, 5408)           │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                  │ (None, 128)            │       692,352 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_3 (Dropout)              │ (None, 128)            │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_3 (Dense)                  │ (None, 10)             │         1,290 │
└──────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 693,962 (2.65 MB)
 Trainable params: 693,962 (2.65 MB)
 Non-trainable params: 0 (0.00 B)
Epoch 1/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 31s 70ms/step - accuracy: 0.8060 - loss: 0.6287 - val_accuracy: 0.9758 - val_loss: 0.08
Epoch 2/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 29s 69ms/step - accuracy: 0.9543 - loss: 0.1562 - val_accuracy: 0.9835 - val_loss: 0.06
Epoch 3/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 30s 72ms/step - accuracy: 0.9677 - loss: 0.1087 - val_accuracy: 0.9860 - val_loss: 0.05
Epoch 4/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 29s 69ms/step - accuracy: 0.9718 - loss: 0.0924 - val_accuracy: 0.9877 - val_loss: 0.04
Epoch 5/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 40s 67ms/step - accuracy: 0.9754 - loss: 0.0805 - val_accuracy: 0.9877 - val_loss: 0.04
Epoch 6/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 41s 68ms/step - accuracy: 0.9786 - loss: 0.0699 - val_accuracy: 0.9893 - val_loss: 0.04
Epoch 7/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 42s 70ms/step - accuracy: 0.9795 - loss: 0.0673 - val_accuracy: 0.9897 - val_loss: 0.04
Epoch 8/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 29s 69ms/step - accuracy: 0.9818 - loss: 0.0558 - val_accuracy: 0.9893 - val_loss: 0.04
Epoch 9/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 29s 69ms/step - accuracy: 0.9835 - loss: 0.0507 - val_accuracy: 0.9893 - val_loss: 0.04
Epoch 10/10
422/422 ━━━━━━━━━━━━━━━━━━━━ 41s 69ms/step - accuracy: 0.9848 - loss: 0.0467 - val_accuracy: 0.9897 - val_loss: 0.04
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.9845 - loss: 0.0451

Test Accuracy: 0.9876
```


Model Accuracy
Accuracy
Epoch
Train Accuracy
Validation Accuracy