```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from keras.utils import to_categorical
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns


# Load the training and testing data
train_data = pd.read_csv('/content/drive/MyDrive/fashion-mnist_train.csv')
test_data = pd.read_csv('/content/drive/MyDrive/fashion-mnist_test.csv')


X_train = train_data.iloc[:, 1:].values  # Assuming the first column is the label
y_train = train_data.iloc[:, 0].values


X_test = test_data.iloc[:, 1:].values
y_test = test_data.iloc[:, 0].values
```

+ Code + Text

```
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')


X_train = X_train / 255.0
X_test = X_test / 255.0


y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)


# Define the model
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:10
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
# Train the model
history = model.fit(X_train, y_train, epochs=10, batch_size=64, validation_split=0.2, verb
```

```
Epoch 1/10
750/750 - 51s - 68ms/step - accuracy: 0.7182 - loss: 0.7859 - val_accuracy: 0.8232 - v
Epoch 2/10
750/750 - 86s - 115ms/step - accuracy: 0.8229 - loss: 0.4900 - val_accuracy: 0.8609 -
Epoch 3/10
750/750 - 78s - 104ms/step - accuracy: 0.8533 - loss: 0.4158 - val_accuracy: 0.8800 -
Epoch 4/10
750/750 - 81s - 108ms/step - accuracy: 0.8698 - loss: 0.3680 - val_accuracy: 0.8820 -
Epoch 5/10
750/750 - 89s - 119ms/step - accuracy: 0.8797 - loss: 0.3389 - val_accuracy: 0.8918 -
Epoch 6/10
750/750 - 47s - 63ms/step - accuracy: 0.8877 - loss: 0.3160 - val_accuracy: 0.8971 - v
Epoch 7/10
750/750 - 84s - 112ms/step - accuracy: 0.8961 - loss: 0.2946 - val_accuracy: 0.9031 -
Epoch 8/10
750/750 - 80s - 107ms/step - accuracy: 0.9005 - loss: 0.2786 - val_accuracy: 0.9018 -
Epoch 9/10
750/750 - 81s - 108ms/step - accuracy: 0.9057 - loss: 0.2631 - val_accuracy: 0.8975 -
Epoch 10/10
750/750 - 47s - 63ms/step - accuracy: 0.9093 - loss: 0.2510 - val_accuracy: 0.9050 - v
```

```
# Evaluate the model
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print(f'\nTest accuracy: {test_acc:.2f}')
```

```
313/313 - 2s - 8ms/step - accuracy: 0.9101 - loss: 0.2567

Test accuracy: 0.91
```

```
# Make predictions
predictions = model.predict(X_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 5s 15ms/step
```

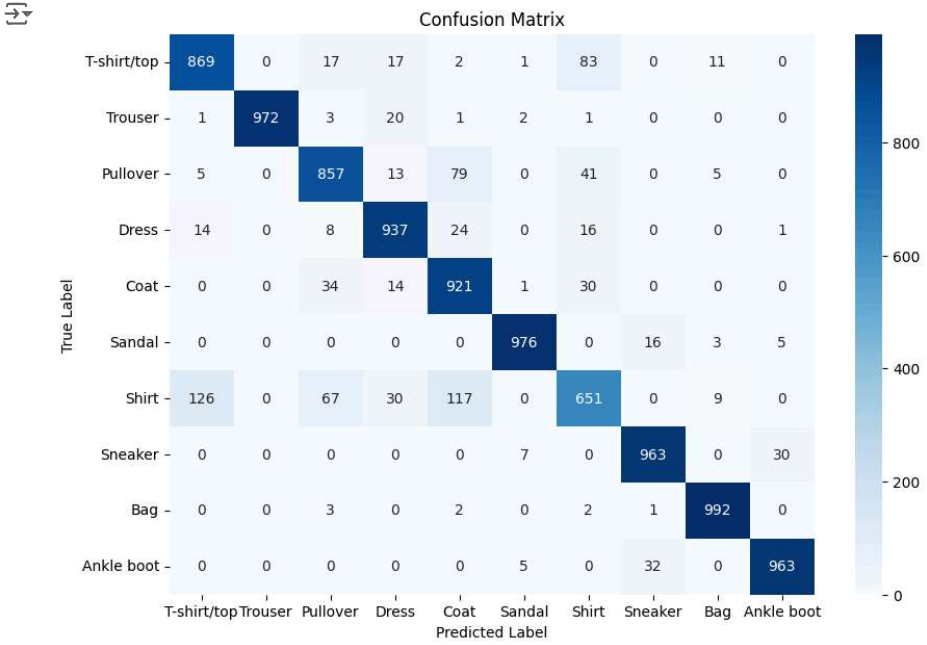
```
# Convert predictions and actual labels from one-hot encoding to integers
y_pred = np.argmax(predictions, axis=1)
y_true = np.argmax(y_test, axis=1)
```

```
print("Classification Report:")
print(classification_report(y_true, y_pred, target_names=[
    'T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', '
```

```
Classification Report:
              precision    recall  f1-score   support

 T-shirt/top       0.86      0.87      0.86      1000
     Trouser       1.00      0.97      0.99      1000
    Pullover       0.87      0.86      0.86      1000
       Dress       0.91      0.94      0.92      1000
        Coat       0.80      0.92      0.86      1000
      Sandal       0.98      0.98      0.98      1000
       Shirt       0.79      0.65      0.71      1000
```

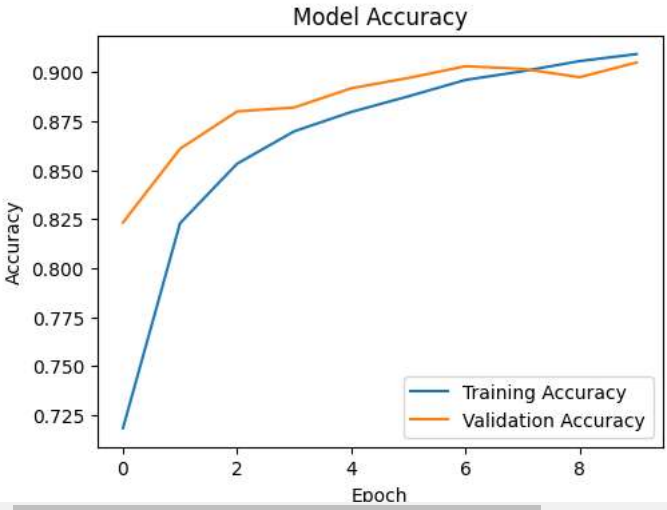
```
           Sneaker       0.95      0.96      0.96      1000
               Bag       0.97      0.99      0.98      1000
        Ankle boot       0.96      0.96      0.96      1000

          accuracy                           0.91     10000
         macro avg       0.91      0.91      0.91     10000
      weighted avg       0.91      0.91      0.91     10000
```

```
# Confusion matrix
cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=[
    'T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', '
    yticklabels=[
    'T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', '
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```

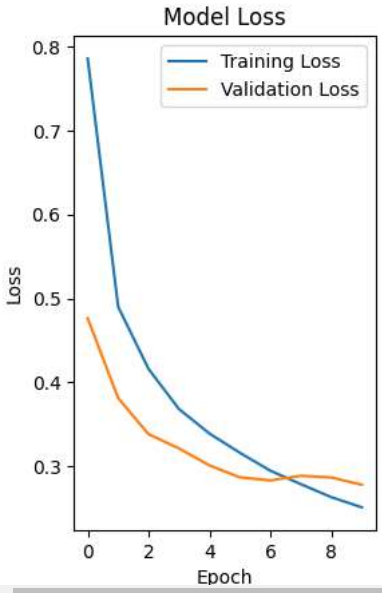


```
# Plot training & validation accuracy values
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
```

```
<matplotlib.legend.Legend at 0x7ba6d7ed3880>
```



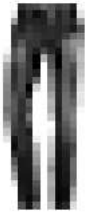

```
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.show()
```



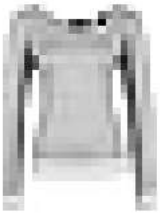

```
# Display a few test images with predictions
labels = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneak
num_images_to_show = 10  # Number of images to display

for i in range(num_images_to_show):
    plt.figure(figsize=(2, 2))

    # Plot the image
    plt.imshow(X_test[i].reshape(28, 28), cmap=plt.cm.binary)
    plt.axis('off')  # Hide axes

    # Show the predicted and actual labels
    predicted_label = labels[y_pred[i]]
    actual_label = labels[y_true[i]]
    plt.title(f'Predicted: {predicted_label}\nActual: {actual_label}', fontsize=10)

    plt.show()
```

Predicted: T-shirt/top
Actual: T-shirt/top

Predicted: Trouser
Actual: Trouser



Predicted: Pullover
Actual: Pullover



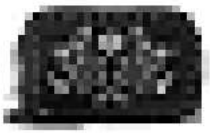
Predicted: Shirt
Actual: Pullover

Predicted: Dress
Actual: Dress

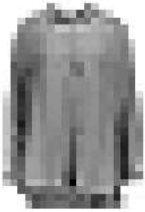
Predicted: Shirt

Actual: Pullover

Predicted: Bag
Actual: Bag



Predicted: Shirt
Actual: Shirt



Predicted: Sandal
Actual: Sandal

Predicted: T-shirt/top
Actual: T-shirt/top