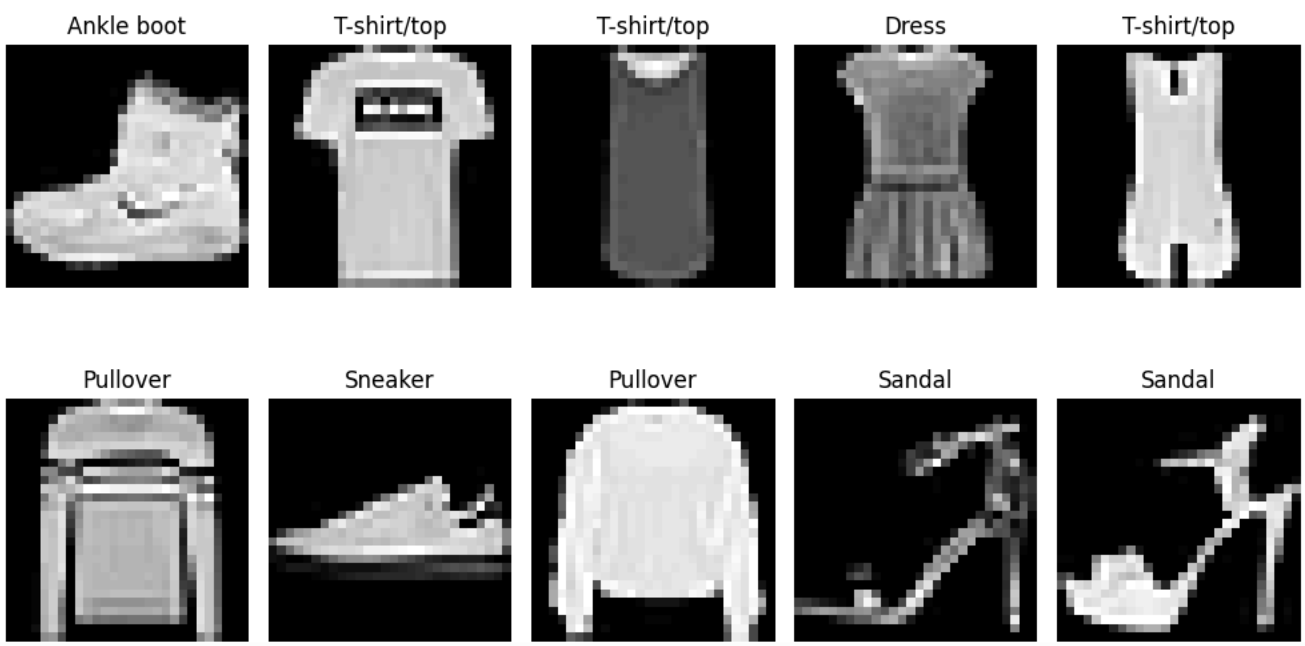
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
```

```
# 1. Import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import classification_report, confusion_matrix
```

```
# 2. Load and Visualize Dataset
fashion_mnist = tf.keras.datasets.fashion_mnist
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()

# Class labels
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

# Plot sample images
plt.figure(figsize=(10, 6))
for i in range(10):
    plt.subplot(2, 5, i + 1)
    plt.imshow(X_train[i], cmap='gray')
    plt.title(class_names[y_train[i]])
    plt.axis('off')
plt.tight_layout()
plt.show()
```




```
# 3. Preprocessing
X_train = X_train / 255.0
X_test = X_test / 255.0

# One-hot encoding
y_train_cat = to_categorical(y_train, 10)
```

```
y_test_cat = to_categorical(y_test, 10)
```

```
# 4. Define Neural Network
model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])

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

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_dim`
  super().__init__(**kwargs)
```

```
# 5. Train Model
history = model.fit(X_train, y_train_cat,
                    epochs=15,
                    batch_size=64,
                    validation_split=0.1,
                    verbose=2)
```

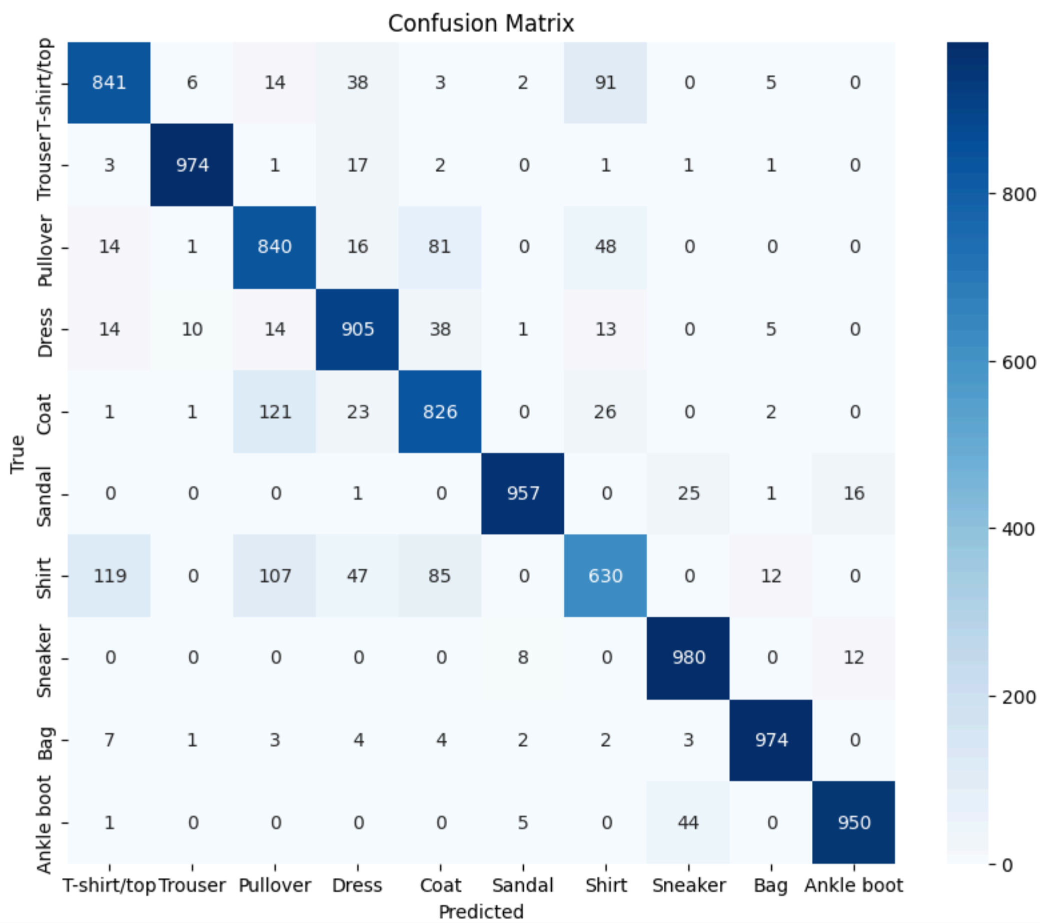
```
Epoch 1/15
844/844 - 6s - 7ms/step - accuracy: 0.8167 - loss: 0.5236 - val_accuracy: 0.8555 - val_loss: 0.4060
Epoch 2/15
844/844 - 4s - 4ms/step - accuracy: 0.8631 - loss: 0.3785 - val_accuracy: 0.8667 - val_loss: 0.3678
Epoch 3/15
844/844 - 3s - 4ms/step - accuracy: 0.8773 - loss: 0.3398 - val_accuracy: 0.8682 - val_loss: 0.3558
Epoch 4/15
844/844 - 6s - 7ms/step - accuracy: 0.8825 - loss: 0.3200 - val_accuracy: 0.8762 - val_loss: 0.3439
Epoch 5/15
844/844 - 4s - 5ms/step - accuracy: 0.8911 - loss: 0.2967 - val_accuracy: 0.8747 - val_loss: 0.3321
Epoch 6/15
844/844 - 6s - 8ms/step - accuracy: 0.8961 - loss: 0.2844 - val_accuracy: 0.8762 - val_loss: 0.3314
Epoch 7/15
844/844 - 4s - 5ms/step - accuracy: 0.8990 - loss: 0.2726 - val_accuracy: 0.8810 - val_loss: 0.3325
Epoch 8/15
844/844 - 3s - 4ms/step - accuracy: 0.9022 - loss: 0.2610 - val_accuracy: 0.8842 - val_loss: 0.3130
Epoch 9/15
844/844 - 4s - 5ms/step - accuracy: 0.9068 - loss: 0.2493 - val_accuracy: 0.8818 - val_loss: 0.3324
Epoch 10/15
844/844 - 4s - 5ms/step - accuracy: 0.9098 - loss: 0.2410 - val_accuracy: 0.8857 - val_loss: 0.3208
Epoch 11/15
844/844 - 5s - 6ms/step - accuracy: 0.9130 - loss: 0.2344 - val_accuracy: 0.8845 - val_loss: 0.3239
Epoch 12/15
844/844 - 5s - 5ms/step - accuracy: 0.9156 - loss: 0.2260 - val_accuracy: 0.8842 - val_loss: 0.3401
Epoch 13/15
844/844 - 4s - 5ms/step - accuracy: 0.9184 - loss: 0.2176 - val_accuracy: 0.8822 - val_loss: 0.3422
Epoch 14/15
844/844 - 5s - 6ms/step - accuracy: 0.9206 - loss: 0.2127 - val_accuracy: 0.8888 - val_loss: 0.3131
Epoch 15/15
844/844 - 5s - 5ms/step - accuracy: 0.9239 - loss: 0.2028 - val_accuracy: 0.8880 - val_loss: 0.3324
```

```
# 6. Evaluate Model
test_loss, test_acc = model.evaluate(X_test, y_test_cat, verbose=0)
print(f"\nTest Accuracy: {test_acc:.4f}")

# Predictions and Evaluation
preds = model.predict(X_test)
y_pred = np.argmax(preds, axis=1)

print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=class_names))
```

```
Test Accuracy: 0.8877
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step

Classification Report:
              precision    recall  f1-score   support

 T-shirt/top       0.84      0.84      0.84      1000
     Trouser       0.98      0.97      0.98      1000
```

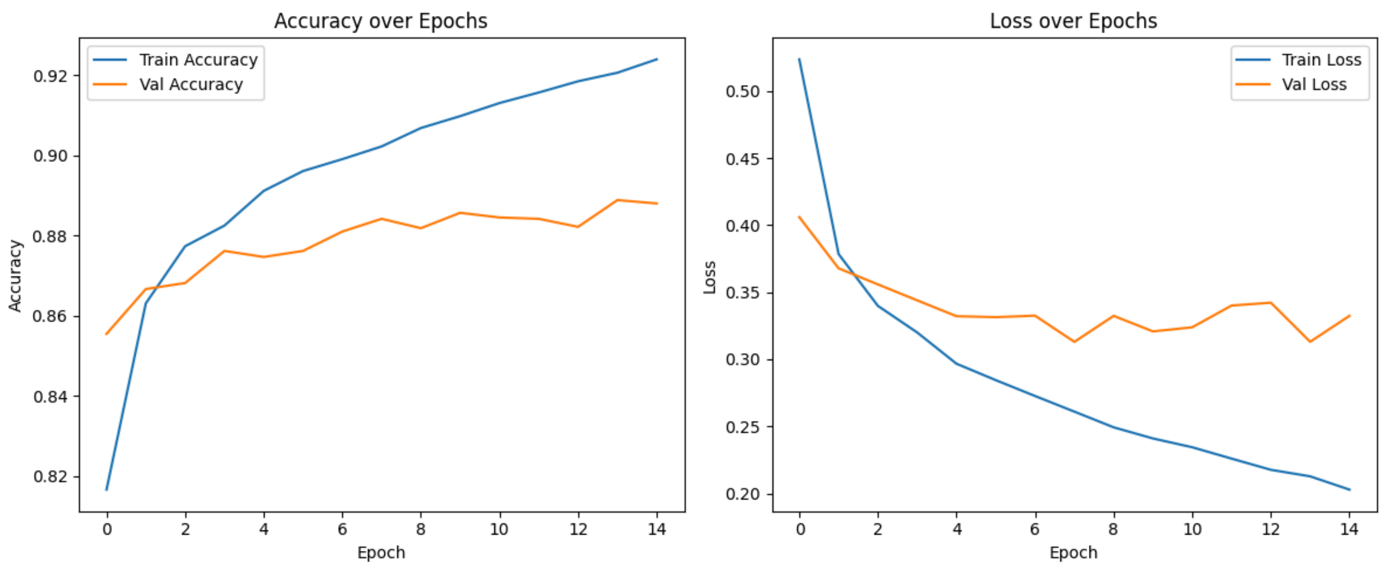
```
    Pullover       0.76      0.84      0.80      1000
       Dress       0.86      0.91      0.88      1000
        Coat       0.79      0.83      0.81      1000
      Sandal       0.98      0.96      0.97      1000
       Shirt       0.78      0.63      0.70      1000
     Sneaker       0.93      0.98      0.95      1000
         Bag       0.97      0.97      0.97      1000
  Ankle boot       0.97      0.95      0.96      1000

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```

```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```




```
# 7. Plot Training History
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.title('Accuracy over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title('Loss over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
```

```
plt.legend()
plt.tight_layout()
plt.show()
```




```
# 8. Deployment Scenario Discussion
print("""
Deployment Scenario:
This model could be deployed in a fashion e-commerce app to auto-classify products based on images uploaded by sellers.
It would streamline tagging and cataloging. Challenges include real-time processing speed,
integration with cloud platforms, and maintaining accuracy on varied image backgrounds.
""")
```

```
Deployment Scenario:
This model could be deployed in a fashion e-commerce app to auto-classify products based on images uploaded by sellers.
It would streamline tagging and cataloging. Challenges include real-time processing speed,
integration with cloud platforms, and maintaining accuracy on varied image backgrounds.
```

```
# 3. Preprocessing
X_train = X_train / 255.0
X_test = X_test / 255.0

# One-hot encoding
y_train_cat = to_categorical(y_train, 10)
y_test_cat = to_categorical(y_test, 10)
```

```
PART 5: Application Demonstration
Hypothetical Use Case
Fashion Retail:

Model used to auto-tag clothing items uploaded by sellers.

Can improve product search and reduce manual tagging.

Challenges:

Needs real-time inference at scale.

Variability in real-world images (lighting, angle) → consider transfer
in production.

Edge deployment possible on mobile for camera-based apps.
```

PART 5: Application Demonstration Hypothetical Use Case Fashion Retail:

Model used to auto-tag clothing items uploaded by sellers.

Can improve product search and reduce manual tagging.

Challenges:

Needs real-time inference at scale.

Variability in real-world images (lighting, angle) → consider transfer learning in production.

Edge deployment possible on mobile for camera-based apps.