

```
In [1]: print("Fashion MNIST Clothing Classification using CNN")
```

Fashion MNIST Clothing Classification using CNN

```
In [2]: import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical
```

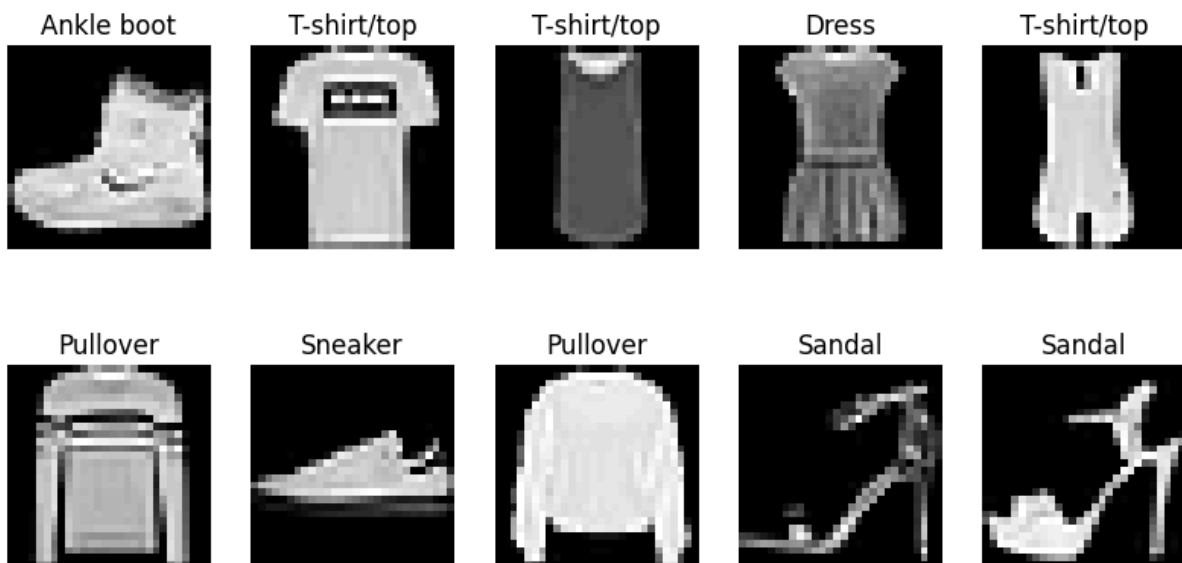
```
In [3]: (X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()
```

```
print("Training data shape:", X_train.shape)
print("Testing data shape:", X_test.shape)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 ————— 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 ————— 3s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 ————— 0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 ————— 0s 0us/step
Training data shape: (60000, 28, 28)
Testing data shape: (10000, 28, 28)
```

```
In [4]: class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
                        'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

```
plt.figure(figsize=(10,5))
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.show()
```



```
In [5]: X_train = X_train / 255.0
X_test = X_test / 255.0

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

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

```
In [6]: model = Sequential()

# Convolution Layer
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)))
model.add(MaxPooling2D((2,2)))

# Second Convolution Layer
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))

# Flatten
model.add(Flatten())

# Fully Connected Layer
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))

# Output Layer
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
D:\DL\.env\lib\site-packages\keras\src\layers\convolutional\base_conv.py:113: UserWarning: Do not pass an `input_shape` or `input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
```

```
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten (Flatten)	(None, 1600)	0
dense (Dense)	(None, 128)	204,928
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

Total params: 225,034 (879.04 KB)

Trainable params: 225,034 (879.04 KB)

Non-trainable params: 0 (0.00 B)

```
In [7]: model.compile(optimizer='adam',
                      loss='categorical_crossentropy',
                      metrics=['accuracy'])
```

WARNING:tensorflow:TensorFlow GPU support is not available on native Windows for TensorFlow >= 2.11. Even if CUDA/cuDNN are installed, GPU will not be used. Please use WSL2 or the TensorFlow-DirectML plugin.

[illegible]

Epoch 1/8
 469/469 ————— 10s 19ms/step - accuracy: 0.7562 - loss: 0.6803 - val_accuracy: 0.8392 - val_loss: 0.4409
 Epoch 2/8
 469/469 ————— 11s 22ms/step - accuracy: 0.8418 - loss: 0.4408 - val_accuracy: 0.8680 - val_loss: 0.3648
 Epoch 3/8
 469/469 ————— 12s 25ms/step - accuracy: 0.8627 - loss: 0.3830 - val_accuracy: 0.8771 - val_loss: 0.3352
 Epoch 4/8
 469/469 ————— 13s 28ms/step - accuracy: 0.8752 - loss: 0.3463 - val_accuracy: 0.8851 - val_loss: 0.3124
 Epoch 5/8
 469/469 ————— 12s 26ms/step - accuracy: 0.8830 - loss: 0.3241 - val_accuracy: 0.8956 - val_loss: 0.2919
 Epoch 6/8
 469/469 ————— 11s 23ms/step - accuracy: 0.8910 - loss: 0.3015 - val_accuracy: 0.8972 - val_loss: 0.2892
 Epoch 7/8
 469/469 ————— 10s 22ms/step - accuracy: 0.8971 - loss: 0.2837 - val_accuracy: 0.9021 - val_loss: 0.2734
 Epoch 8/8
 469/469 ————— 12s 25ms/step - accuracy: 0.9020 - loss: 0.2709 - val_accuracy: 0.9033 - val_loss: 0.2640

In [9]: test_loss, test_acc = model.evaluate(X_test, y_test)

```
print("Test Accuracy:", test_acc)
```

313/313 ————— 2s 5ms/step - accuracy: 0.9033 - loss: 0.2640

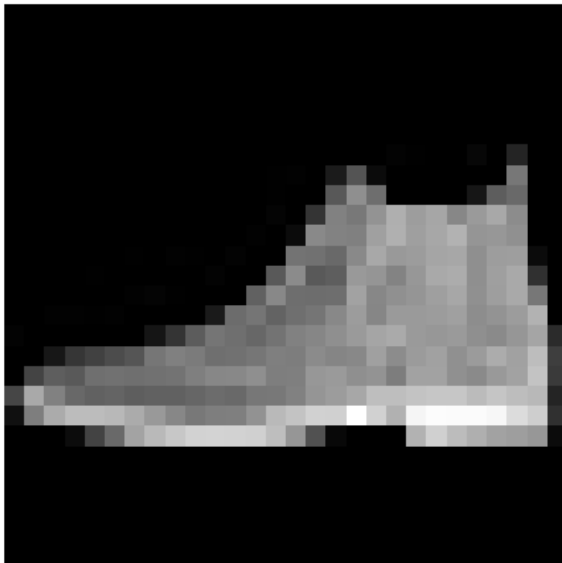
Test Accuracy: 0.9032999873161316

In [14]: predictions = model.predict(X_test)

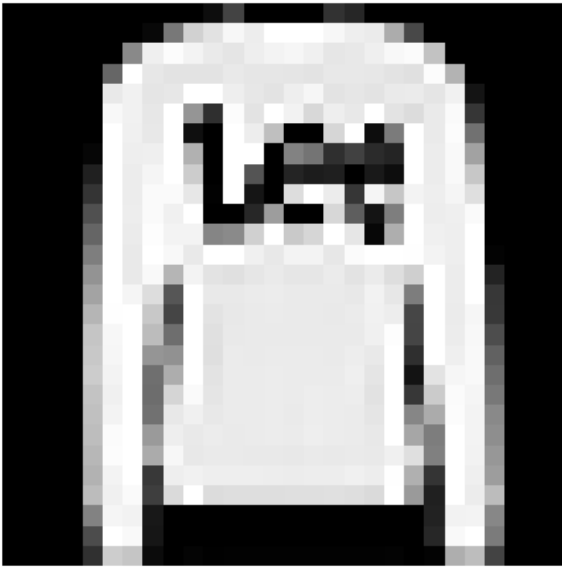
```
for i in range(5):
    plt.imshow(X_test[i].reshape(28,28), cmap='gray')
    plt.title("Predicted: " + class_names[np.argmax(predictions[i])])
    plt.axis('off')
    plt.show()
```

313/313 ————— 1s 5ms/step

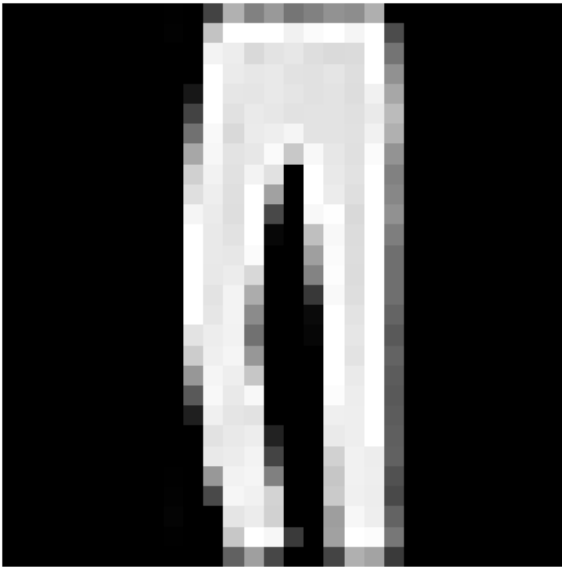
Predicted: Ankle boot



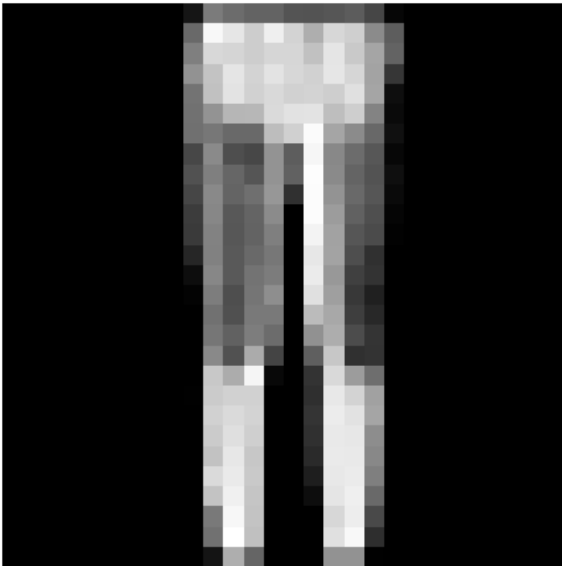
Predicted: Pullover



Predicted: Trouser



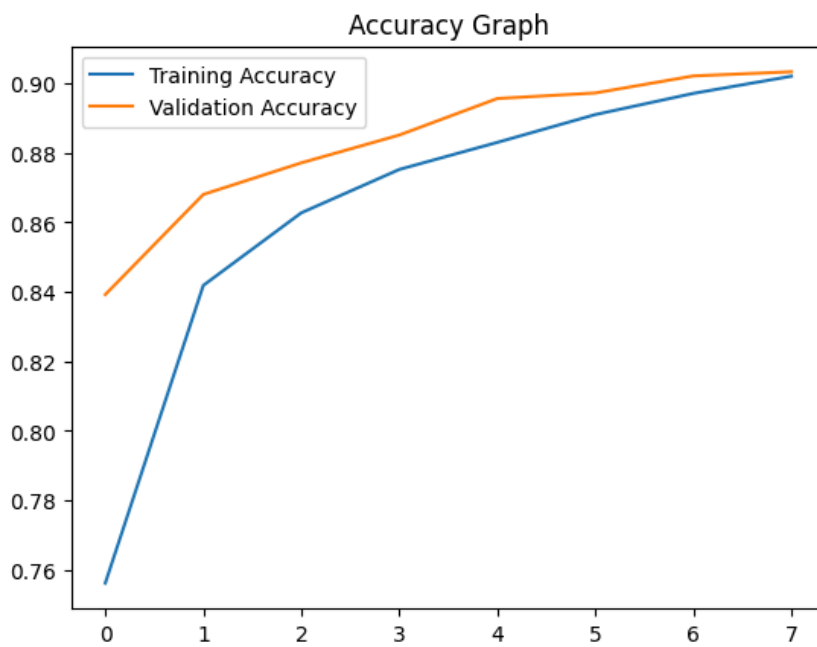
Predicted: Trouser



Predicted: Shirt



```
In [15]: plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title("Accuracy Graph")
plt.show()
```



```
In [ ]:
```