```
import tensorflow as tf
from tensorflow import keras
from keras import layers
from keras.datasets import fashion_mnist


import ssl
ssl._create_default_https_context = ssl._create_unverified_context
# Load the dataset
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

```
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 [==============================] - 1s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

```
# Normalize the pixel values to be between 0 and 1
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255


# Convert the labels to one-hot encoded vectors
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)


# Build the model
model = keras.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.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(num_classes, activation='softmax')
])


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


# Train the model
model.fit(x_train.reshape(-1,28,28,1), y_train, epochs=10, batch_size=32, validation_data=(x_test.reshape(-1,28,28,1), y_test))
```

```
Epoch 1/10
1875/1875 [==============================] - 68s 35ms/step - loss: 0.4466 - accuracy: 0.8388 - val_loss: 0.3437 - val_accuracy: 0.87
Epoch 2/10
1875/1875 [==============================] - 60s 32ms/step - loss: 0.2997 - accuracy: 0.8903 - val_loss: 0.3048 - val_accuracy: 0.89
Epoch 3/10
1875/1875 [==============================] - 57s 30ms/step - loss: 0.2545 - accuracy: 0.9064 - val_loss: 0.2665 - val_accuracy: 0.90
Epoch 4/10
1875/1875 [==============================] - 57s 30ms/step - loss: 0.2208 - accuracy: 0.9176 - val_loss: 0.2608 - val_accuracy: 0.90
Epoch 5/10
1875/1875 [==============================] - 57s 30ms/step - loss: 0.1948 - accuracy: 0.9276 - val_loss: 0.2586 - val_accuracy: 0.90
Epoch 6/10
1875/1875 [==============================] - 56s 30ms/step - loss: 0.1730 - accuracy: 0.9355 - val_loss: 0.2470 - val_accuracy: 0.91
Epoch 7/10
1875/1875 [==============================] - 58s 31ms/step - loss: 0.1509 - accuracy: 0.9421 - val_loss: 0.2539 - val_accuracy: 0.91
Epoch 8/10
1875/1875 [==============================] - 56s 30ms/step - loss: 0.1334 - accuracy: 0.9496 - val_loss: 0.2782 - val_accuracy: 0.90
Epoch 9/10
1875/1875 [==============================] - 58s 31ms/step - loss: 0.1198 - accuracy: 0.9549 - val_loss: 0.2894 - val_accuracy: 0.90
Epoch 10/10
1875/1875 [==============================] - 56s 30ms/step - loss: 0.1045 - accuracy: 0.9606 - val_loss: 0.2891 - val_accuracy: 0.90
<keras.src.callbacks.History at 0x79f263659b70>
```

```
# Evaluate the model
test_loss, test_acc = model.evaluate(x_test.reshape(-1,28,28,1), y_test, verbose=2)
print('Test accuracy:', test_acc)
```

```
313/313 - 2s - loss: 0.2891 - accuracy: 0.9077 - 2s/epoch - 8ms/step
Test accuracy: 0.9077000021934509
```