```
import tensorflow as tf
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.utils import to_categorical
import numpy as np



# Load the MNIST Fashion dataset
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/tra
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/tra
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10
4422102/4422102 [==============================] - 0s 0us/step
```

```
# Normalize pixel values to the range [0, 1]
train_images = train_images.astype('float32') / 255.0
test_images = test_images.astype('float32') / 255.0


# Convert labels to one-hot encoding
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)


# Create CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')  # 10 classes for fashion items
])


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

```
# Train the model
history = model.fit(
    train_images,
    train_labels,
    epochs=10,
    batch_size=128,
    validation_data=(test_images, test_labels)
)
```

```
Epoch 1/10
469/469 [==============================] - 53s 111ms/step - loss: 0.6809 - accuracy:
Epoch 2/10
469/469 [==============================] - 52s 111ms/step - loss: 0.4450 - accuracy:
Epoch 3/10
469/469 [==============================] - 52s 111ms/step - loss: 0.3884 - accuracy:
Epoch 4/10
469/469 [==============================] - 53s 113ms/step - loss: 0.3521 - accuracy:
Epoch 5/10
469/469 [==============================] - 53s 113ms/step - loss: 0.3257 - accuracy:
Epoch 6/10
469/469 [==============================] - 51s 109ms/step - loss: 0.3003 - accuracy:
Epoch 7/10
469/469 [==============================] - 53s 113ms/step - loss: 0.2820 - accuracy:
Epoch 8/10
469/469 [==============================] - 53s 113ms/step - loss: 0.2639 - accuracy:
Epoch 9/10
469/469 [==============================] - 52s 112ms/step - loss: 0.2489 - accuracy:
Epoch 10/10
469/469 [==============================] - 51s 109ms/step - loss: 0.2308 - accuracy:
```

```
results = model.evaluate(test_images, test_labels)
```

```
313/313 [==============================] - 3s 9ms/step - loss: 0.3089 - accuracy: 0.8
```

```
results
```

```
[0.30885961651802063, 0.8891000151634216]
```

Start coding or generate with AI.