In [ ]:

```
import tensorflow as tf
from tensorflow.keras import layers, models

# Load Fashion-MNIST dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()

# Normalize pixel values to be between 0 and 1
x_train, x_test = x_train / 255.0, x_test / 255.0

# Reshape the data to (28, 28, 1) as Fashion-MNIST consists of grayscale images
x_train = x_train.reshape((x_train.shape[0], 28, 28, 1))
x_test = x_test.reshape((x_test.shape[0], 28, 28, 1))

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

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

# Train the model
history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))

# Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_acc}')
```

```
Epoch 1/10
1875/1875 [==============================] - 29s 15ms/step - loss: 0.5117 - accurac
y: 0.8116 - val_loss: 0.3982 - val_accuracy: 0.8527
Epoch 2/10
1875/1875 [==============================] - 29s 16ms/step - loss: 0.3308 - accurac
y: 0.8802 - val_loss: 0.3428 - val_accuracy: 0.8785
Epoch 3/10
1875/1875 [==============================] - 31s 16ms/step - loss: 0.2804 - accurac
y: 0.8971 - val_loss: 0.3107 - val_accuracy: 0.8898
Epoch 4/10
1875/1875 [==============================] - 31s 16ms/step - loss: 0.2512 - accurac
y: 0.9081 - val_loss: 0.2820 - val_accuracy: 0.9002
Epoch 5/10
1875/1875 [==============================] - 30s 16ms/step - loss: 0.2282 - accurac
y: 0.9158 - val_loss: 0.2574 - val_accuracy: 0.9063
Epoch 6/10
1875/1875 [==============================] - 31s 16ms/step - loss: 0.2081 - accurac
y: 0.9224 - val_loss: 0.2965 - val_accuracy: 0.8958
Epoch 7/10
1875/1875 [==============================] - 30s 16ms/step - loss: 0.1905 - accurac
y: 0.9293 - val_loss: 0.2695 - val_accuracy: 0.9051
Epoch 8/10
1875/1875 [==============================] - 30s 16ms/step - loss: 0.1748 - accurac
y: 0.9346 - val_loss: 0.2643 - val_accuracy: 0.9082
Epoch 9/10
1875/1875 [==============================] - 32s 17ms/step - loss: 0.1614 - accurac
y: 0.9385 - val_loss: 0.2949 - val_accuracy: 0.9058
Epoch 10/10
1875/1875 [==============================] - 32s 17ms/step - loss: 0.1487 - accurac
y: 0.9446 - val_loss: 0.2715 - val_accuracy: 0.9134
313/313 [==============================] - 2s 6ms/step - loss: 0.2715 - accuracy: 0.
9134
Test accuracy: 0.9133999943733215
```

In [ ]:

```python
# print some samples
import matplotlib.pyplot as plt
import numpy as np

# get the predictions
predictions = model.predict(x_test)

# get the index of the largest probability
predictions = np.argmax(predictions, axis=1)

# get the first 25 samples
x_test = x_test[:25]
y_test = y_test[:25]
predictions = predictions[:25]

# plot the samples
plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
```

```
    plt.imshow(x_test[i].reshape(28, 28), cmap=plt.cm.binary)
    plt.xlabel(f'Actual: {y_test[i]}, Predicted: {predictions[i]}')
plt.show()
```

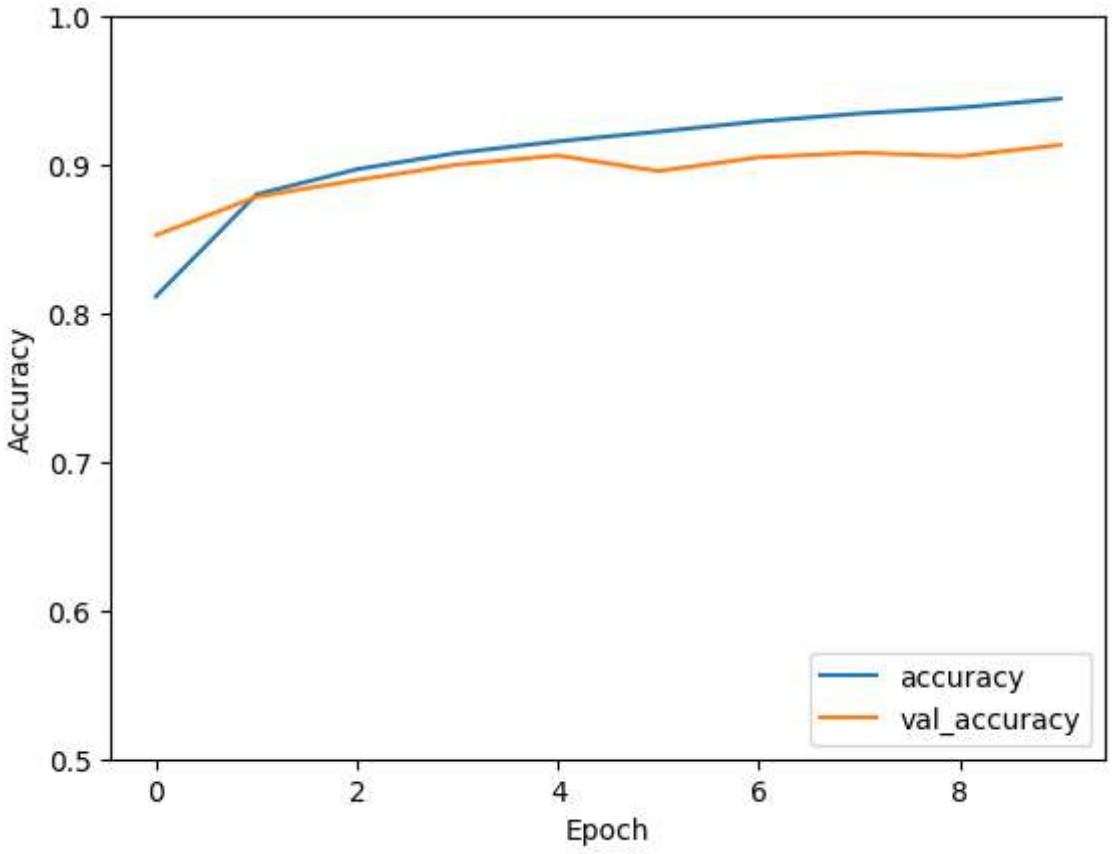
```
313/313 [==============================] - 2s 5ms/step
```

In [ ]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 64)          0
 g2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

=================================================================
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 64)          0
 g2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

 flatten (Flatten)           (None, 576)               0

 dense (Dense)               (None, 64)                36928

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 93322 (364.54 KB)
Trainable params: 93322 (364.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

plt.show()
```



In [ ]:
```
# print the accuracy
print(f'Test accuracy: {test_acc}')
```

```
Test accuracy: 0.9133999943733215
```