In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
from keras.datasets import fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

In [3]:

```
train_images[0]
```

Out[3]:

```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   1,
          0,   0,  13,  73,   0,   0,   1,   4,   0,   0,   0,   0,   1,
          1,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
          0,  36, 136, 127,  62,  54,   0,   0,   0,   1,   3,   4,   0,
          0,   3],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   6,
          0, 102, 204, 176, 134, 144, 123,  23,   0,   0,   0,   0,  12,
         10,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0, 155, 236, 207, 178, 107, 156, 161, 109,  64,  23,  77, 130,
```

In [4]:

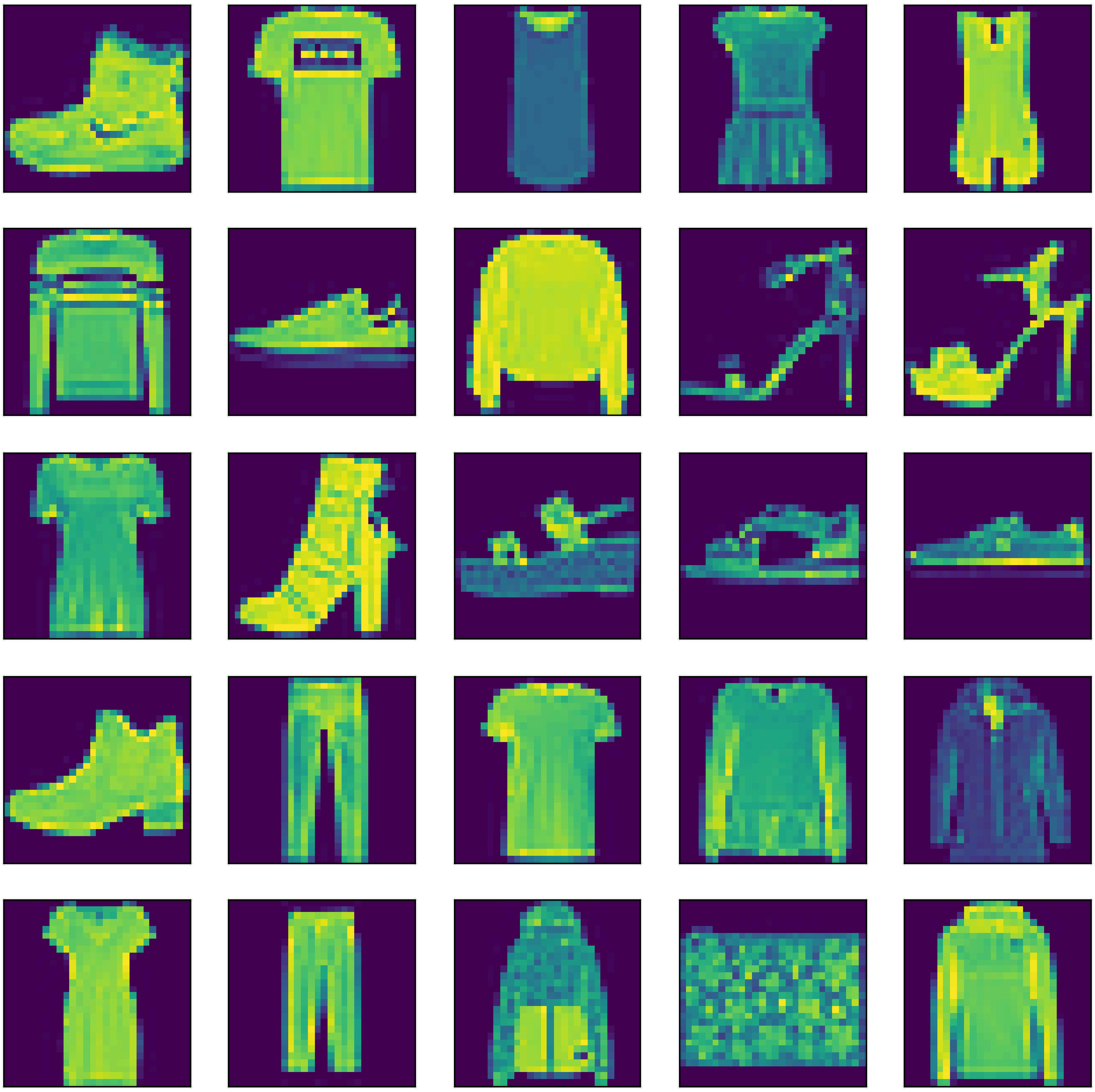
```
train_labels[0]
```

Out[4]:

```
9
```

In [12]:

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

for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.imshow(train_images[i])
    plt.xticks([])
    plt.yticks([])

plt.show()
```



Scale the Data

In [5]:
```
train_images = train_images/255.0
test_images = test_images/255.0
```

Define the model structure

In [6]:
```
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D

model = Sequential()
model
```

Out[6]:
```
<Sequential name=sequential, built=False>
```

In [7]:
```
model.add(Conv2D(64, (3, 3), input_shape=(28, 28, 1), activation="relu"))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation="relu"))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation="relu"))
model.add(Dense(10, activation="softmax"))
```

```
D:\py\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:99: UserWarning: Do not pass an `input_shape`/`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__(
```

Compiler the Model

In [8]:
```
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                 │ (None, 26, 26, 64)     │           640 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 13, 13, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)               │ (None, 11, 11, 64)     │        36,928 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 5, 5, 64)       │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 1600)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 128)            │       204,928 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 10)             │         1,290 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 243,786 (952.29 KB)

 Trainable params: 243,786 (952.29 KB)

 Non-trainable params: 0 (0.00 B)
```

Train the Model

In [9]:
```
history = model.fit(train_images, train_labels, epochs = 2, batch_size = 512, verbose = 1)
```

```
Epoch 1/2
118/118 ━━━━━━━━━━━━━━━━━━━━ 40s 330ms/step - accuracy: 0.6328 - loss: 1.1061
Epoch 2/2
118/118 ━━━━━━━━━━━━━━━━━━━━ 44s 376ms/step - accuracy: 0.8348 - loss: 0.4569
```

Evaluate the Model

In [10]:
```
results = model.evaluate(test_images, test_labels)
results
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - accuracy: 0.8505 - loss: 0.4134
```

Out[10]:
```
[0.41731637716293335, 0.848800003528595]
```

Make Predictions

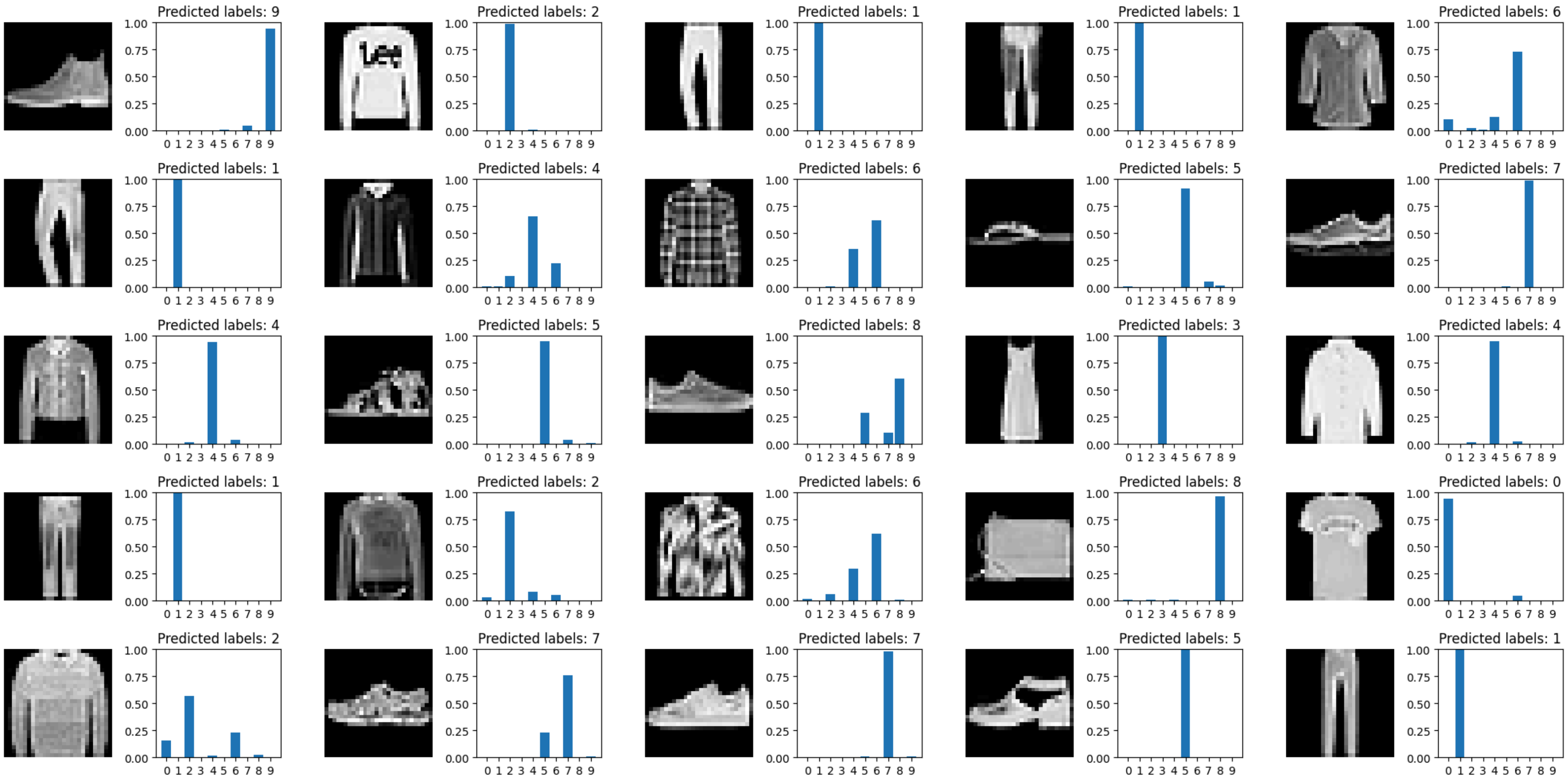
In [13]:
```
predictions = model.predict(test_images)
predicted_labels = np.argmax(predictions, axis = 1)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step
```

Display the Predictions

In [14]:

```
rows = 5
cols = 5
num_images = rows * cols

plt.figure(figsize=(2 * 2 * cols, 2 * rows))

for i in range(num_images):

    # plot the images
    plt.subplot(rows, 2 * cols, 2 * i + 1)
    plt.imshow(test_images[i], cmap='gray')
    plt.axis('off')

    # plot the bar chart
    plt.subplot(rows, 2 * cols, 2 * i + 2)
    plt.bar(range(10), predictions[i])
    plt.xticks((range(10)))
    plt.ylim([0, 1])
    plt.title(f"Predicted labels: {predicted_labels[i]}")
    plt.tight_layout()

plt.show()
```



Predicted labels: 9
Predicted labels: 2
Predicted labels: 1
Predicted labels: 1
Predicted labels: 6
Predicted labels: 1
Predicted labels: 4
Predicted labels: 6
Predicted labels: 5
Predicted labels: 7
Predicted labels: 4
Predicted labels: 5
Predicted labels: 8
Predicted labels: 3
Predicted labels: 4
Predicted labels: 1
Predicted labels: 2
Predicted labels: 6
Predicted labels: 8
Predicted labels: 0
Predicted labels: 2
Predicted labels: 7
Predicted labels: 7
Predicted labels: 5
Predicted labels: 1

In [ ]: