human-faces-detection

May 2, 2023

1 1. Library and Data

[1]:
```
import nbconvert
```

[192]:
```
import os
import cv2 as cv
import random
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from sklearn.metrics import mean_absolute_error
```

[193]:
```
! pip install -q kaggle
from google.colab import files
files.upload()
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
```

```
<IPython.core.display.HTML object>

Saving kaggle.json to kaggle (1).json
mkdir: cannot create directory ‘/root/.kaggle’: File exists
```

[ ]:
```
!kaggle datasets download -d sbaghbidi/human-faces-object-detection
```

[ ]:
```
!unzip human-faces-object-detection.zip
```

[196]:
```
df = pd.read_csv('faces.csv')
```

[197]:
```
df.head(7)
```

[197]:

	image_name	width	height	x0	y0	x1	y1
0	00001722.jpg	1333	2000	490	320	687	664
1	00001044.jpg	2000	1333	791	119	1200	436
2	00001050.jpg	667	1000	304	155	407	331

```
3  00001736.jpg    626     417  147   14   519  303
4  00003121.jpg    626     418  462   60   599  166
5  00003121.jpg    626     418  316  157   441  254
6  00003121.jpg    626     418   35   71   160  168
```

[198]:
```
df.shape
```

[198]:
```
(3350, 7)
```

2 2. Preparation

Resizing the images

[199]:
```
data = {}
for i in df['image_name']:
  if i not in data:
    data[i] = []

for idx, img_name in enumerate(df['image_name']):
  width = df["width"][idx]
  height = df["height"][idx]
  x1 = df["x0"][idx]
  y1 = df["y0"][idx]
  x2 = df["x1"][idx]
  y2 = df["y1"][idx]
  new_x1 = int((x1/width)*128)
  new_y1 = int((y1/height)*128)
  new_x2 = int((x2/width)*128)
  new_y2 = int((y2/height)*128)
  data[img_name].append(new_x1)
  data[img_name].append(new_y1)
  data[img_name].append(new_x2)
  data[img_name].append(new_y2)
```

[200]:
```
image_dir = os.listdir('/content/images')
images = []
for image_name in data.keys():
  for itr in image_dir:
    if image_name == itr:
      image_arr = cv.imread(os.path.join('/content/images', image_name), cv.
 ↪IMREAD_GRAYSCALE)
      resized_image = cv.resize(image_arr, (128, 128))
      images.append(resized_image)
```

[201]:
```
images = np.array(images)
images = np.expand_dims(images, axis=3)
```

[202]:
```
print(f"Images shape: {images.shape}")
```

```
Images shape: (2204, 128, 128, 1)
```

Bounding Box

[203]:
```
bbox = []
for boxes in data.keys():
  bbox.append(data[boxes])
```

[204]:
```
maxlen = 0
for i in bbox:
  length = len(i)
  if length > maxlen:
    maxlen = length
bbox = tf.keras.preprocessing.sequence.pad_sequences(bbox, maxlen=max,␣
 ↪padding='post')
```

[205]:
```
bbox = np.array(bbox)
print(f"shape of bbox {bbox.shape}")
```

```
shape of bbox (2204, 48)
```

Rescale

[206]:
```
images = images/255
bbox = bbox/128
```

[207]:
```
plt.figure(figsize=(15,10))
for i in range(25):
  plt.subplot(5, 5, i+1)
  plt.imshow(images[-i], cmap='gray')
  plt.axis("off")
```

[208]:
```
def split_data(X, Y, train_size):
  m, n = Y.shape
  random.seed(42)
  shuffle_index = random.sample(range(m), m)
  train_index = int(m*train_size)

  data_training = X[:train_index]
  data_testing = X[train_index:]
  label_training = Y[:train_index]
  label_testing = Y[train_index:]

  return data_training, label_training, data_testing, label_testing
```

[209]:
```
data_training, label_training, data_testing, label_testing = split_data(images,␣
↪bbox, 0.85)
```

[210]:
```
print(f"Training images shape: {data_training.shape}")
print(f"Training labels shape: {label_training.shape}")
print(f"Testing images shape: {data_testing.shape}")
print(f"Testing labels shape: {label_testing.shape}")
```

```
Training images shape: (1873, 128, 128, 1)
```

```
Training labels shape: (1873, 48)
Testing images shape: (331, 128, 128, 1)
Testing labels shape: (331, 48)
```

3 3. CNN Model

[211]:
```
model1 = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), padding='same', activation='relu',␣
 ↪input_shape=(128,128,1)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(32, (3,3), padding='same', activation='relu',␣
 ↪input_shape=(128,128,1)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), padding='same', activation='relu',␣
 ↪input_shape=(128,128,1)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(48, activation='sigmoid')
])
model1_initial_weights = model1.get_weights()
model1.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_36 (Conv2D)          (None, 128, 128, 16)      160

 max_pooling2d_36 (MaxPoolin  (None, 64, 64, 16)       0
 g2D)

 conv2d_37 (Conv2D)          (None, 64, 64, 32)        4640

 max_pooling2d_37 (MaxPoolin  (None, 32, 32, 32)       0
 g2D)

 conv2d_38 (Conv2D)          (None, 32, 32, 64)        18496

 max_pooling2d_38 (MaxPoolin  (None, 16, 16, 64)       0
 g2D)

 flatten_10 (Flatten)        (None, 16384)             0

 dense_30 (Dense)            (None, 128)               2097280
```

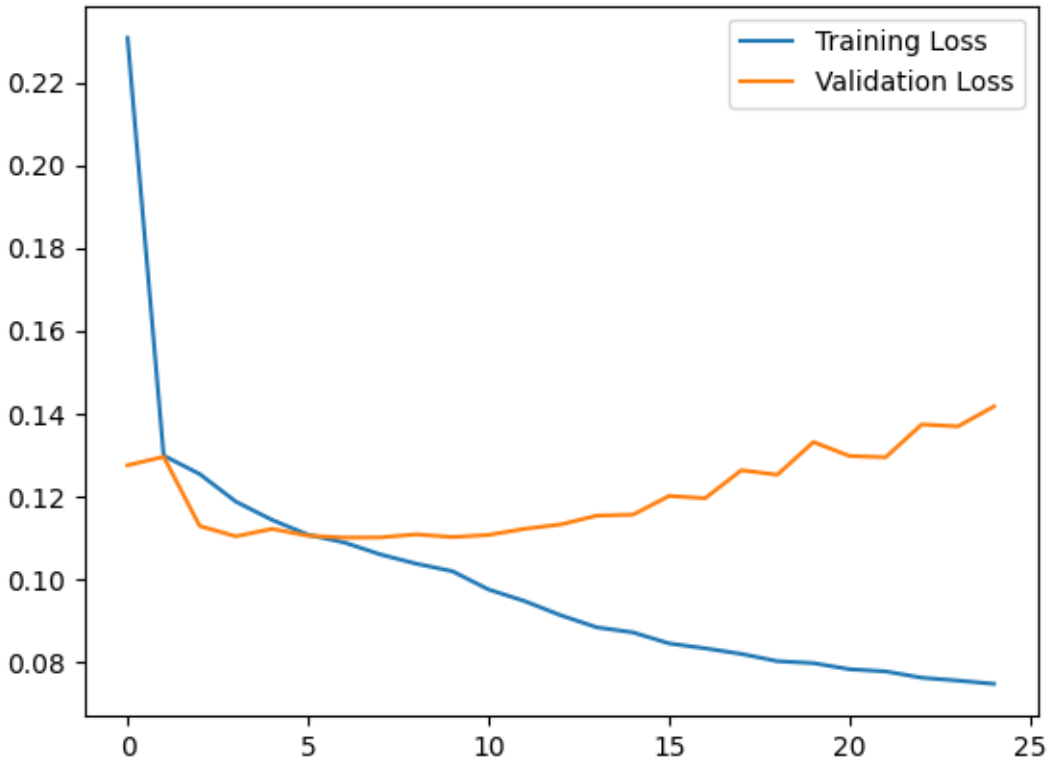
```
 dense_31 (Dense)            (None, 64)                8256

 dense_32 (Dense)            (None, 48)                3120

=================================================================
Total params: 2,131,952
Trainable params: 2,131,952
Non-trainable params: 0
_________________________________________________________________
```

[213]:
```
model1.set_weights(model1_initial_weight)

model1.compile(optimizer = "adam",
               loss="binary_crossentropy")

history1 = model1.fit(data_training, label_training,␣
 ↪validation_data=(data_testing, label_testing), epochs=25)
```

```
Epoch 1/25
59/59 [==============================] - 43s 654ms/step - loss: 0.2307 -
val_loss: 0.1274
Epoch 2/25
59/59 [==============================] - 36s 597ms/step - loss: 0.1298 -
val_loss: 0.1295
Epoch 3/25
59/59 [==============================] - 36s 615ms/step - loss: 0.1253 -
val_loss: 0.1127
Epoch 4/25
59/59 [==============================] - 35s 602ms/step - loss: 0.1186 -
val_loss: 0.1102
Epoch 5/25
59/59 [==============================] - 34s 567ms/step - loss: 0.1142 -
val_loss: 0.1120
Epoch 6/25
59/59 [==============================] - 36s 605ms/step - loss: 0.1106 -
val_loss: 0.1104
Epoch 7/25
59/59 [==============================] - 33s 566ms/step - loss: 0.1088 -
val_loss: 0.1099
Epoch 8/25
59/59 [==============================] - 35s 598ms/step - loss: 0.1059 -
val_loss: 0.1100
Epoch 9/25
59/59 [==============================] - 35s 600ms/step - loss: 0.1036 -
val_loss: 0.1107
Epoch 10/25
59/59 [==============================] - 36s 600ms/step - loss: 0.1018 -
val_loss: 0.1101
```

```
Epoch 11/25
59/59 [==============================] - 35s 595ms/step - loss: 0.0974 -
val_loss: 0.1106
Epoch 12/25
59/59 [==============================] - 34s 572ms/step - loss: 0.0946 -
val_loss: 0.1121
Epoch 13/25
59/59 [==============================] - 40s 685ms/step - loss: 0.0912 -
val_loss: 0.1131
Epoch 14/25
59/59 [==============================] - 35s 600ms/step - loss: 0.0883 -
val_loss: 0.1153
Epoch 15/25
59/59 [==============================] - 34s 572ms/step - loss: 0.0870 -
val_loss: 0.1155
Epoch 16/25
59/59 [==============================] - 35s 603ms/step - loss: 0.0844 -
val_loss: 0.1200
Epoch 17/25
59/59 [==============================] - 37s 627ms/step - loss: 0.0832 -
val_loss: 0.1195
Epoch 18/25
59/59 [==============================] - 35s 593ms/step - loss: 0.0818 -
val_loss: 0.1262
Epoch 19/25
59/59 [==============================] - 35s 597ms/step - loss: 0.0801 -
val_loss: 0.1251
Epoch 20/25
59/59 [==============================] - 34s 570ms/step - loss: 0.0796 -
val_loss: 0.1330
Epoch 21/25
59/59 [==============================] - 35s 595ms/step - loss: 0.0781 -
val_loss: 0.1297
Epoch 22/25
59/59 [==============================] - 35s 598ms/step - loss: 0.0776 -
val_loss: 0.1294
Epoch 23/25
59/59 [==============================] - 34s 577ms/step - loss: 0.0761 -
val_loss: 0.1373
Epoch 24/25
59/59 [==============================] - 37s 627ms/step - loss: 0.0754 -
val_loss: 0.1368
Epoch 25/25
59/59 [==============================] - 35s 593ms/step - loss: 0.0746 -
val_loss: 0.1416
```

[283]:
```
loss = history1.history['loss']
val_loss = history1.history['val_loss']
epoch = range(len(loss))

plt.plot(epoch, loss)
plt.plot(epoch, val_loss)
plt.legend(["Training Loss", "Validation Loss"])
plt.show()
```



[216]:
```
model2 = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), padding='same', activation='relu',␣
 ↪input_shape=(128,128,1)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(32, (3,3), padding='same', activation='relu',␣
 ↪input_shape=(128,128,1)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(64, (3,3), padding='same', activation='relu',␣
 ↪input_shape=(128,128,1)),
```

```
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(128, (3,3), padding='same', activation='relu',␣
 ↪input_shape=(128,128,1)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(48, activation='sigmoid')
])
model2_initial_weights = model2.get_weights()
model2.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_39 (Conv2D)          (None, 128, 128, 16)      160

 dropout_4 (Dropout)         (None, 128, 128, 16)      0

 max_pooling2d_39 (MaxPoolin  (None, 64, 64, 16)       0
 g2D)

 batch_normalization_4 (Batc  (None, 64, 64, 16)       64
 hNormalization)

 conv2d_40 (Conv2D)          (None, 64, 64, 32)        4640

 dropout_5 (Dropout)         (None, 64, 64, 32)        0

 max_pooling2d_40 (MaxPoolin  (None, 32, 32, 32)       0
 g2D)

 batch_normalization_5 (Batc  (None, 32, 32, 32)       128
 hNormalization)

 conv2d_41 (Conv2D)          (None, 32, 32, 64)        18496

 dropout_6 (Dropout)         (None, 32, 32, 64)        0

 max_pooling2d_41 (MaxPoolin  (None, 16, 16, 64)       0
 g2D)
```

```
batch_normalization_6 (Batc  (None, 16, 16, 64)       256
hNormalization)

conv2d_42 (Conv2D)           (None, 16, 16, 128)      73856

dropout_7 (Dropout)          (None, 16, 16, 128)      0

max_pooling2d_42 (MaxPoolin  (None, 8, 8, 128)        0
g2D)

batch_normalization_7 (Batc  (None, 8, 8, 128)        512
hNormalization)

flatten_11 (Flatten)         (None, 8192)             0

dense_33 (Dense)             (None, 128)              1048704

dense_34 (Dense)             (None, 64)               8256

dense_35 (Dense)             (None, 48)               3120

=================================================================
Total params: 1,158,192
Trainable params: 1,157,712
Non-trainable params: 480
_________________________________________________________________
```

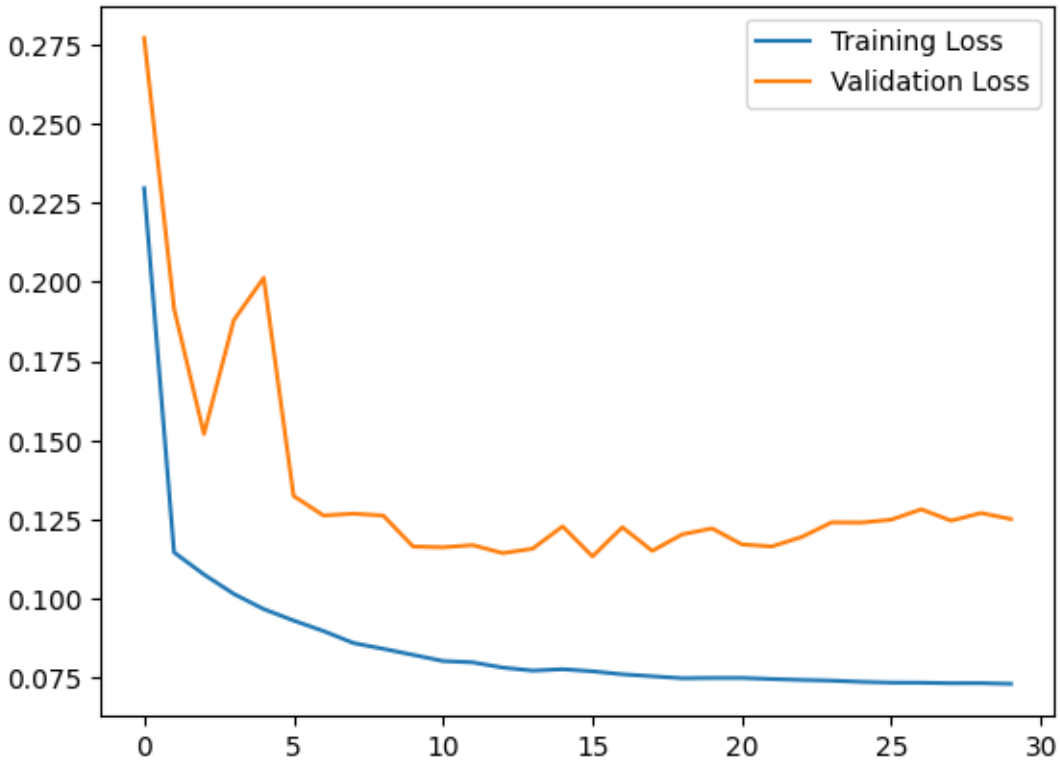
[217]:
```
model2.set_weights(model2_initial_weights)

model2.compile(optimizer = "adam",
               loss="binary_crossentropy")

history2 = model2.fit(data_training, label_training,␣
 ↪validation_data=(data_testing, label_testing), epochs=30)
```

```
Epoch 1/30
59/59 [==============================] - 59s 947ms/step - loss: 0.2295 -
val_loss: 0.2770
Epoch 2/30
59/59 [==============================] - 55s 936ms/step - loss: 0.1146 -
val_loss: 0.1916
Epoch 3/30
59/59 [==============================] - 57s 951ms/step - loss: 0.1077 -
val_loss: 0.1520
Epoch 4/30
59/59 [==============================] - 57s 961ms/step - loss: 0.1015 -
val_loss: 0.1879
Epoch 5/30
```

```
59/59 [==============================] - 55s 932ms/step - loss: 0.0967 -
val_loss: 0.2013
Epoch 6/30
59/59 [==============================] - 55s 936ms/step - loss: 0.0931 -
val_loss: 0.1324
Epoch 7/30
59/59 [==============================] - 57s 968ms/step - loss: 0.0898 -
val_loss: 0.1262
Epoch 8/30
59/59 [==============================] - 56s 941ms/step - loss: 0.0860 -
val_loss: 0.1268
Epoch 9/30
59/59 [==============================] - 58s 981ms/step - loss: 0.0842 -
val_loss: 0.1262
Epoch 10/30
59/59 [==============================] - 56s 953ms/step - loss: 0.0823 -
val_loss: 0.1164
Epoch 11/30
59/59 [==============================] - 57s 953ms/step - loss: 0.0803 -
val_loss: 0.1162
Epoch 12/30
59/59 [==============================] - 55s 940ms/step - loss: 0.0799 -
val_loss: 0.1169
Epoch 13/30
59/59 [==============================] - 57s 972ms/step - loss: 0.0782 -
val_loss: 0.1144
Epoch 14/30
59/59 [==============================] - 56s 943ms/step - loss: 0.0773 -
val_loss: 0.1158
Epoch 15/30
59/59 [==============================] - 56s 958ms/step - loss: 0.0777 -
val_loss: 0.1228
Epoch 16/30
59/59 [==============================] - 56s 944ms/step - loss: 0.0771 -
val_loss: 0.1133
Epoch 17/30
59/59 [==============================] - 57s 965ms/step - loss: 0.0761 -
val_loss: 0.1225
Epoch 18/30
59/59 [==============================] - 57s 976ms/step - loss: 0.0755 -
val_loss: 0.1151
Epoch 19/30
59/59 [==============================] - 57s 959ms/step - loss: 0.0749 -
val_loss: 0.1203
Epoch 20/30
59/59 [==============================] - 56s 944ms/step - loss: 0.0750 -
val_loss: 0.1222
Epoch 21/30
```

```
59/59 [==============================] - 55s 940ms/step - loss: 0.0750 -
val_loss: 0.1171
Epoch 22/30
59/59 [==============================] - 55s 935ms/step - loss: 0.0746 -
val_loss: 0.1165
Epoch 23/30
59/59 [==============================] - 57s 964ms/step - loss: 0.0744 -
val_loss: 0.1195
Epoch 24/30
59/59 [==============================] - 57s 956ms/step - loss: 0.0741 -
val_loss: 0.1241
Epoch 25/30
59/59 [==============================] - 55s 939ms/step - loss: 0.0738 -
val_loss: 0.1240
Epoch 26/30
59/59 [==============================] - 55s 935ms/step - loss: 0.0735 -
val_loss: 0.1249
Epoch 27/30
59/59 [==============================] - 55s 938ms/step - loss: 0.0735 -
val_loss: 0.1282
Epoch 28/30
59/59 [==============================] - 60s 1s/step - loss: 0.0733 - val_loss:
0.1246
Epoch 29/30
59/59 [==============================] - 55s 937ms/step - loss: 0.0733 -
val_loss: 0.1270
Epoch 30/30
59/59 [==============================] - 55s 940ms/step - loss: 0.0731 -
val_loss: 0.1251
```

[218]:
```
loss = history2.history['loss']
val_loss = history2.history['val_loss']
epoch = range(len(loss))

plt.plot(epoch, loss)
plt.plot(epoch, val_loss)
plt.legend(["Training Loss", "Validation Loss"])
plt.show()
```



Trained Weights

[220]:
```
model1_trained_weights = model1.get_weights()
model2_trained_weights = model2.get_weights()
```

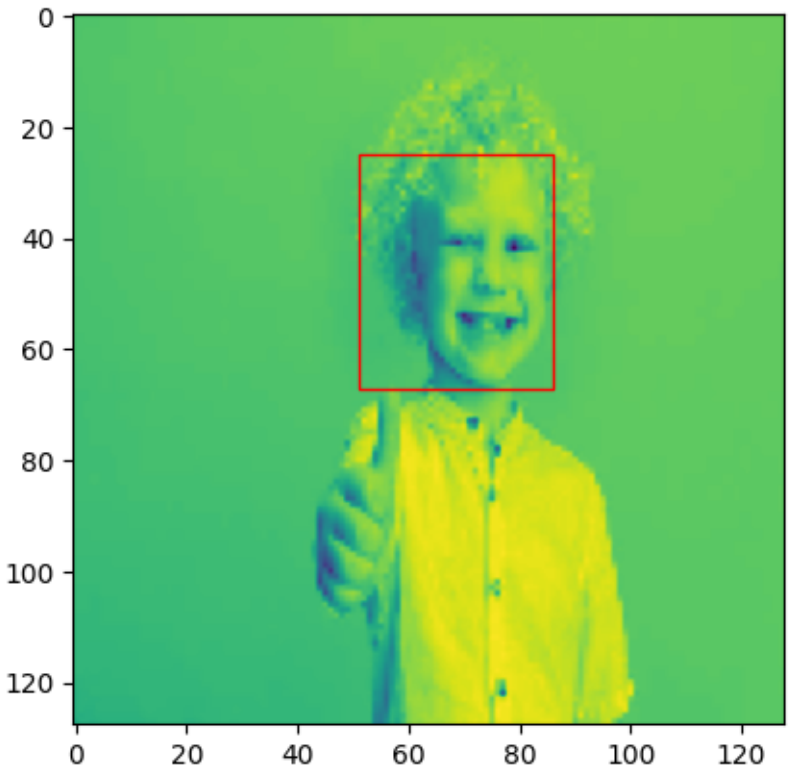
4 4. Prediction

[268]:
```
def model_prediction(num, model):
  data = data_testing[num]
  prediction = model.predict(data.reshape(1, 128, 128, 1))
  fig, ax = plt.subplots(1)
  ax.imshow(data)
  x1 = int(prediction[0][0]*128)
  y1 = int(prediction[0][1]*128)
  x2 = int(prediction[0][2]*128)
  y2 = int(prediction[0][3]*128)
  rect = patches.Rectangle((x1, y1), x2-x1, y2-y1, linewidth=1,␣
 ↪edgecolor='red', facecolor="none")
  ax.add_patch(rect)
  plt.show()
```

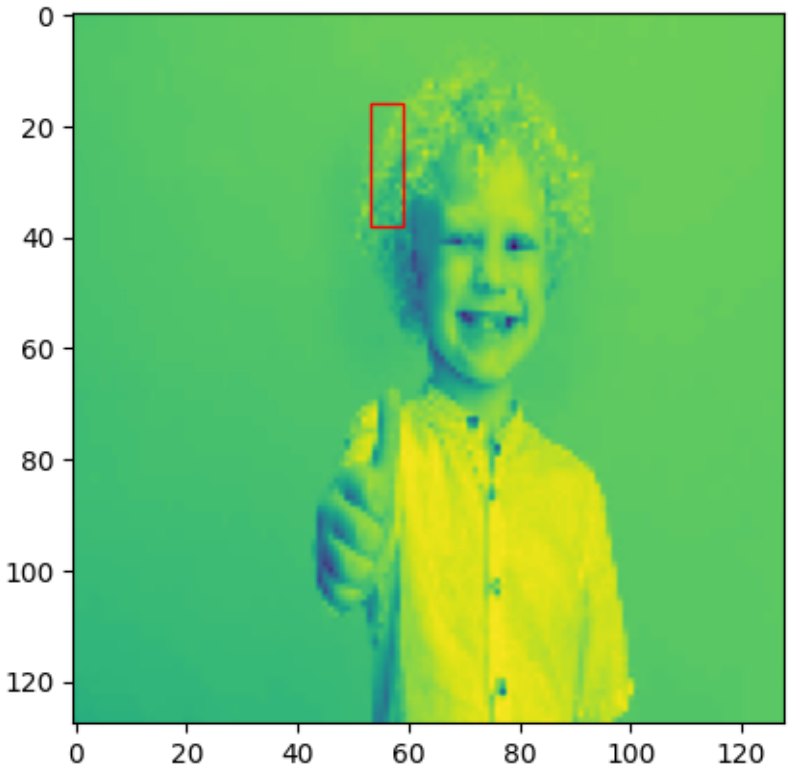
[266]:
```
def model_prediction(num, model):
  for i in range(num):
    data = data_testing[i]
    prediction = model.predict(data.reshape(1, 128, 128, 1))
    num_sqrt = int(np.sqrt(num))
    fig, ax = plt.subplot(num_sqrt, num_sqrt, i+1)
    ax.imshow(data)
    x1 = int(prediction[0][0]*128)
    y1 = int(prediction[0][1]*128)
    x2 = int(prediction[0][2]*128)
    y2 = int(prediction[0][3]*128)
    rect = patches.Rectangle((x1, y1), x2-x1, y2-y1, linewidth=1,␣
 ↪edgecolor='red', facecolor="none")
    ax.add_patch(rect)
    plt.show()
```

[282]:
```
num = 75 # The first model is better
model_prediction(num, model1)
model_prediction(num, model2)
```

```
1/1 [==============================] - 0s 30ms/step
```

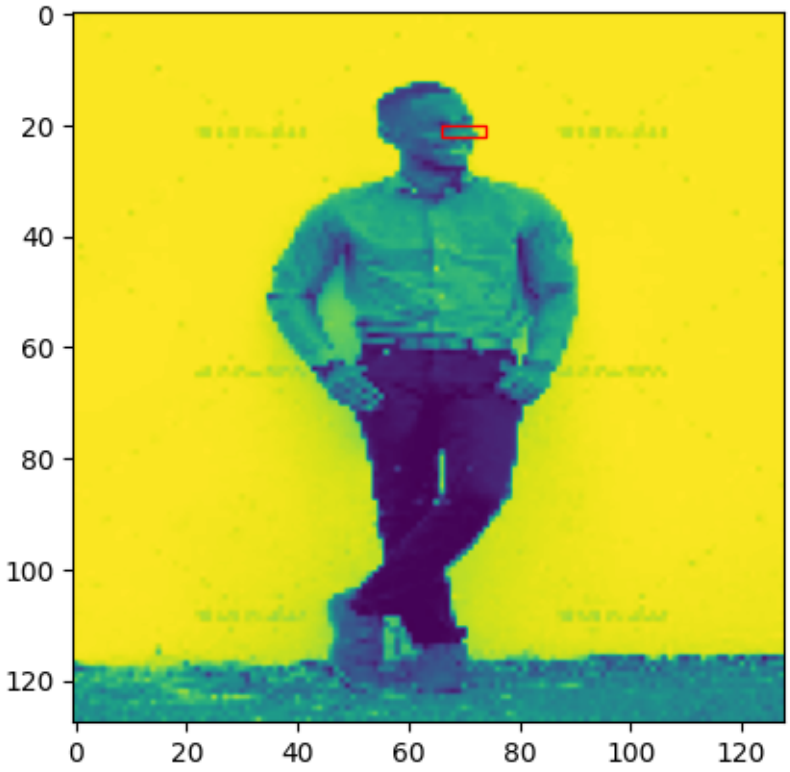


```
1/1 [==============================] - 0s 53ms/step
```

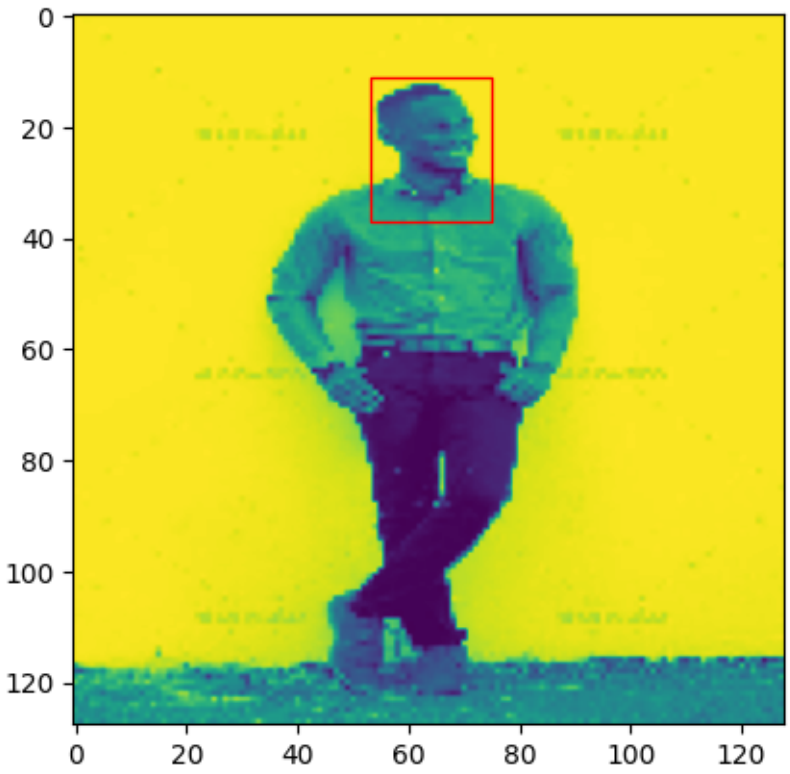


[278]:
```
num = 15# The second model is better
model_prediction(num, model1)
model_prediction(num, model2)
```

```
1/1 [==============================] - 0s 78ms/step
```

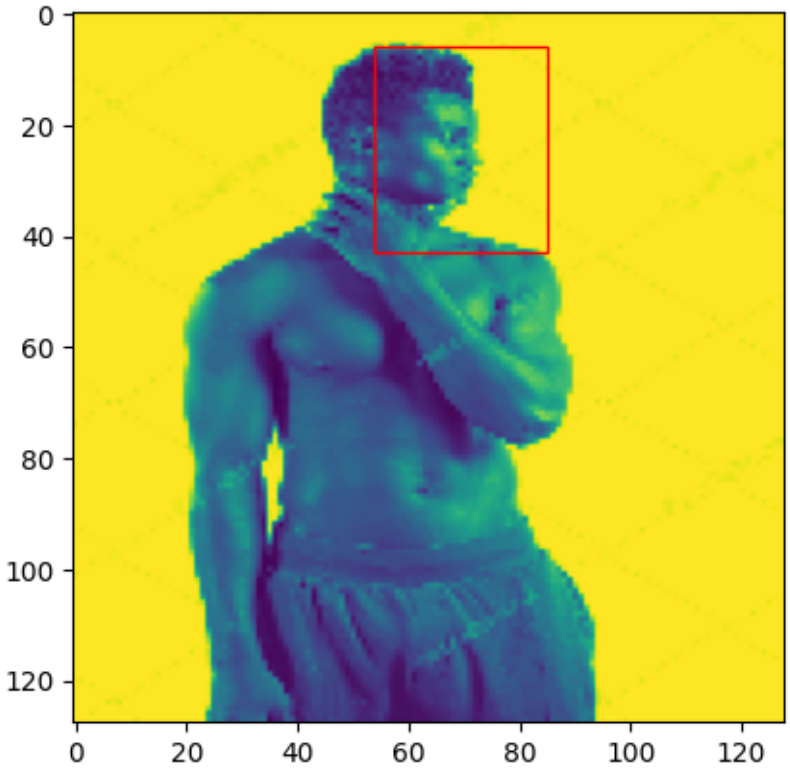



```
1/1 [==============================] - 0s 90ms/step
```

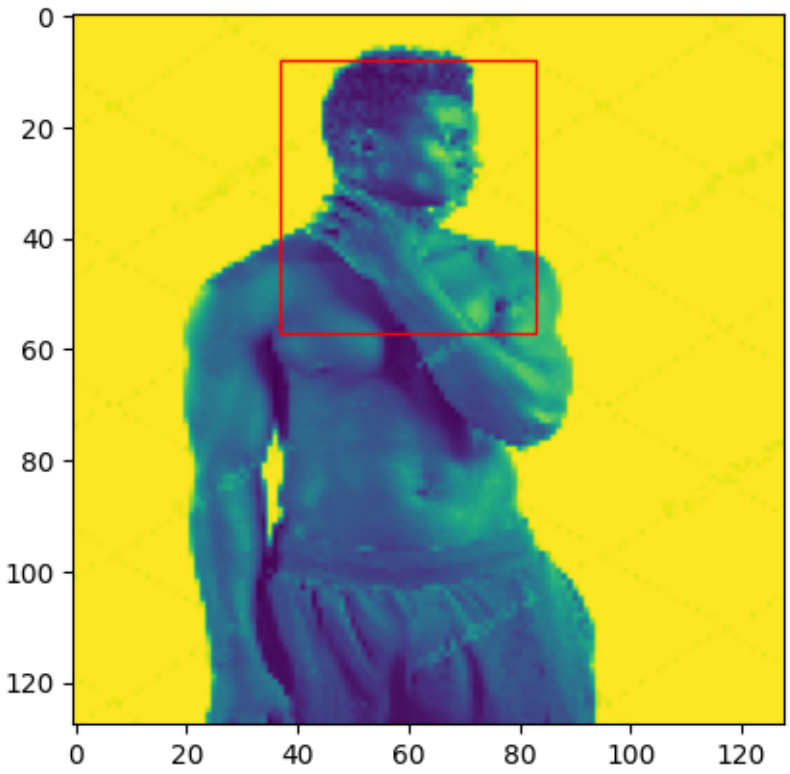


[303]:
```
num = 91 # Both models are good, but the first model is slightly better
model_prediction(num, model1)
model_prediction(num, model2)
```

```
1/1 [==============================] - 0s 48ms/step
```

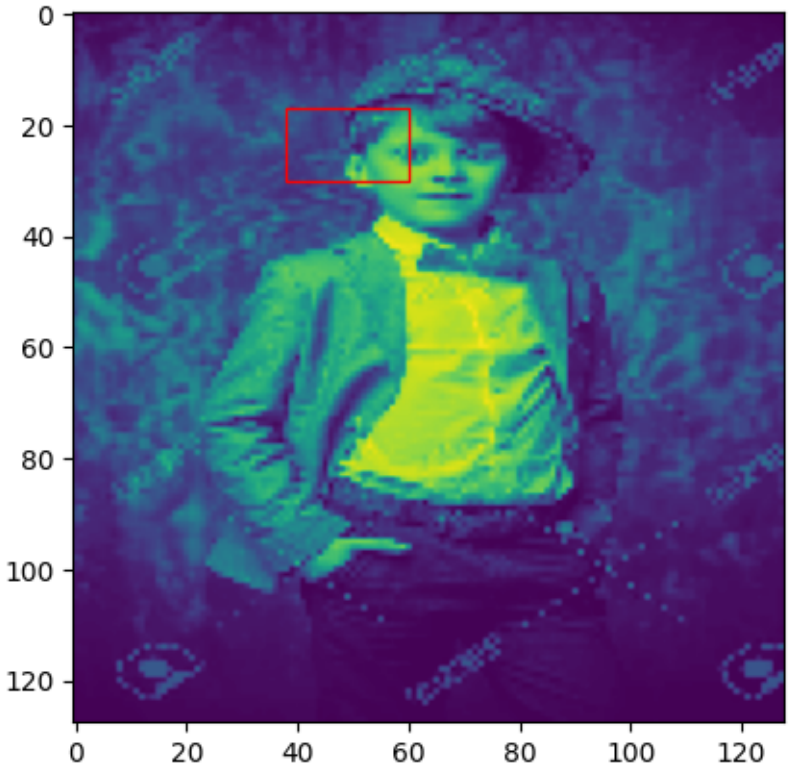



```
1/1 [==============================] - 0s 52ms/step
```

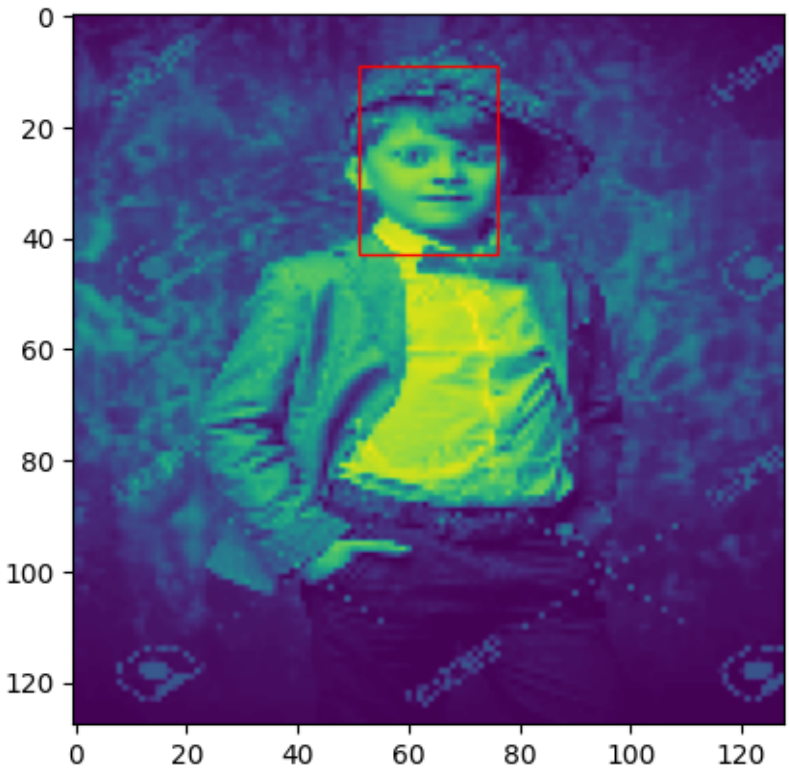


[287]:
```
num = 7 # The second model is better
model_prediction(num, model1)
model_prediction(num, model2)
```

```
1/1 [==============================] - 0s 34ms/step
```

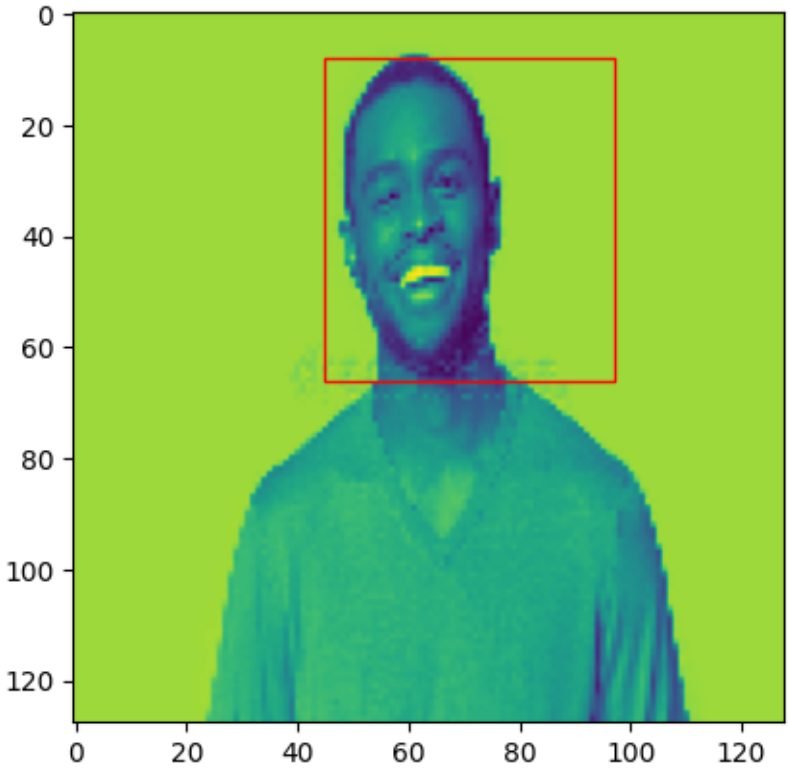



```
1/1 [==============================] - 0s 40ms/step
```

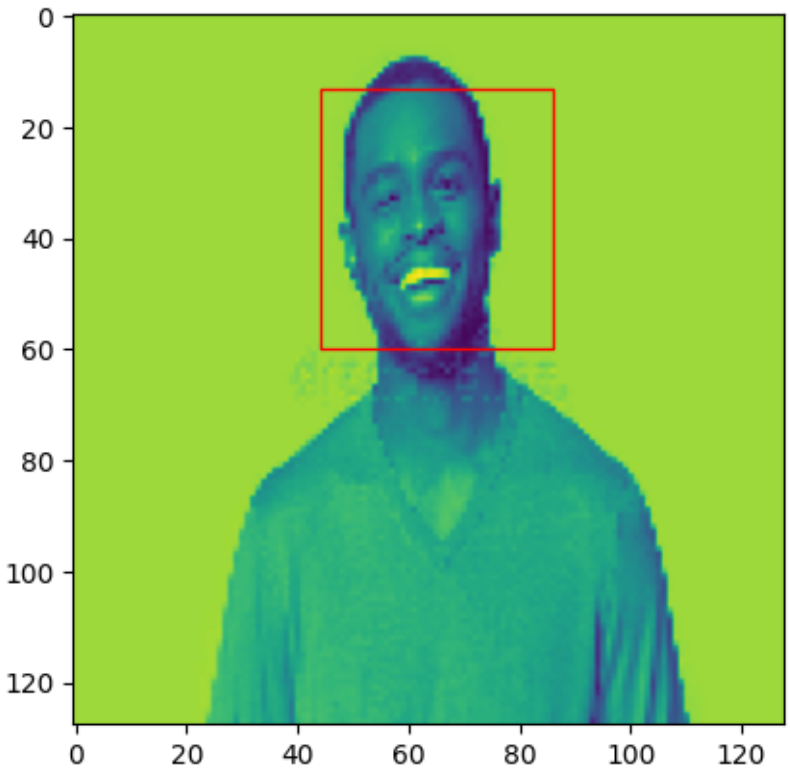


[291]:
```
num = 11 # Both models are good
model_prediction(num, model1)
model_prediction(num, model2)
```

```
1/1 [==============================] - 0s 29ms/step
```

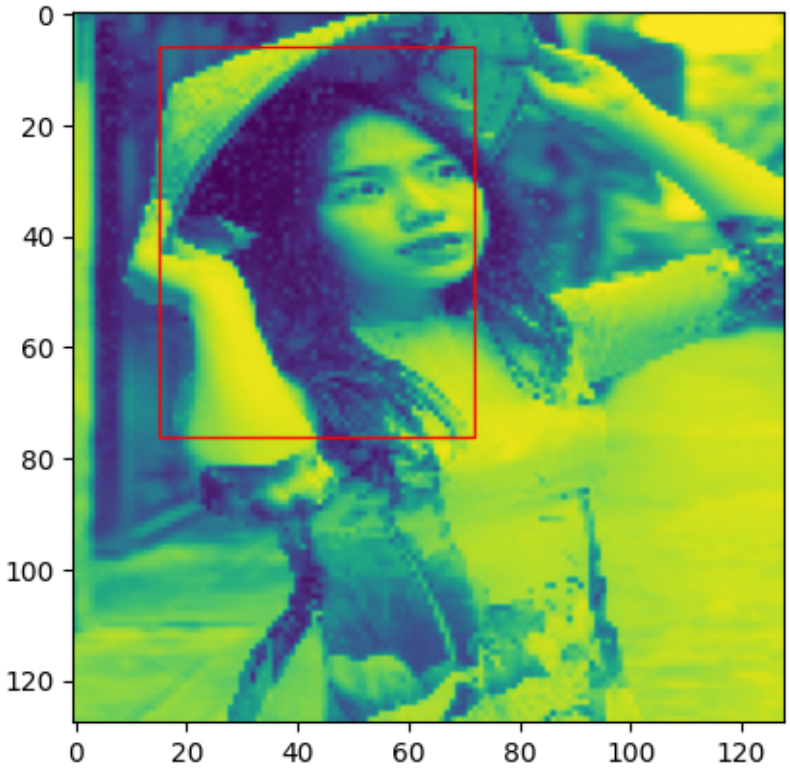



```
1/1 [==============================] - 0s 36ms/step
```

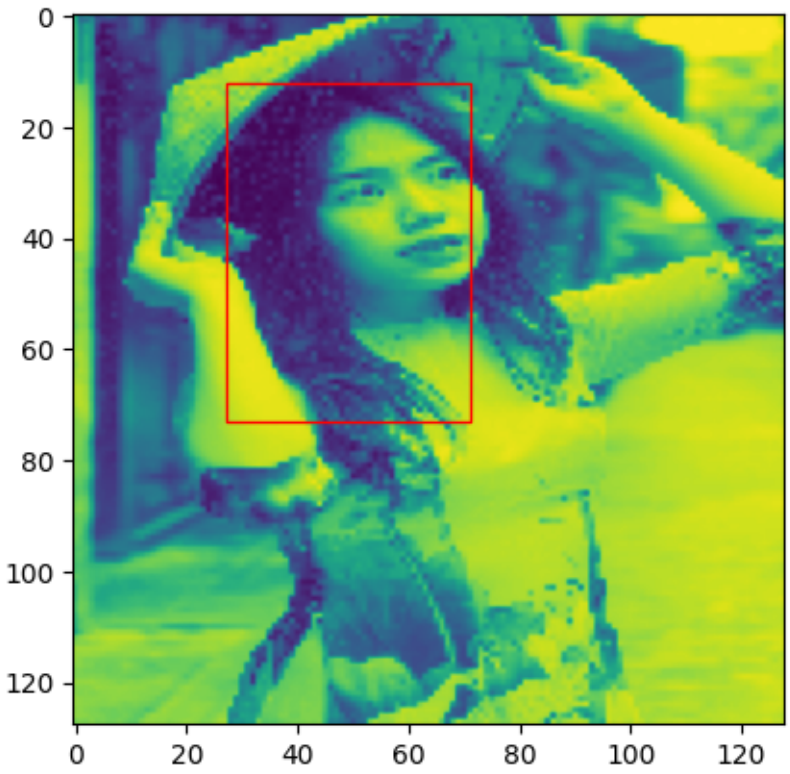


[298]:
```
num = 18 # Both models are good
model_prediction(num, model1)
model_prediction(num, model2)
```

```
1/1 [==============================] - 0s 28ms/step
```

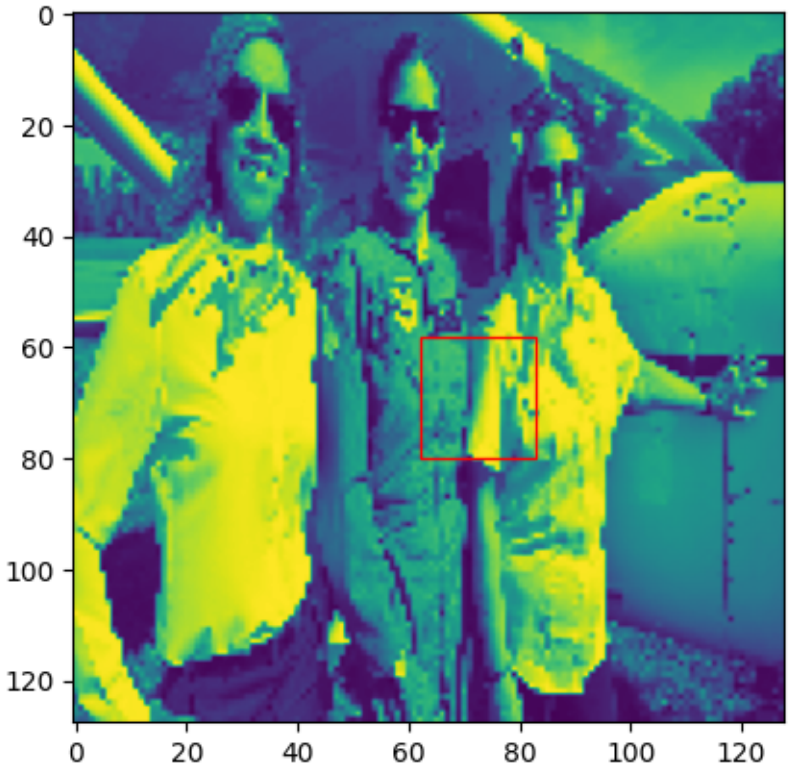



```
1/1 [==============================] - 0s 32ms/step
```

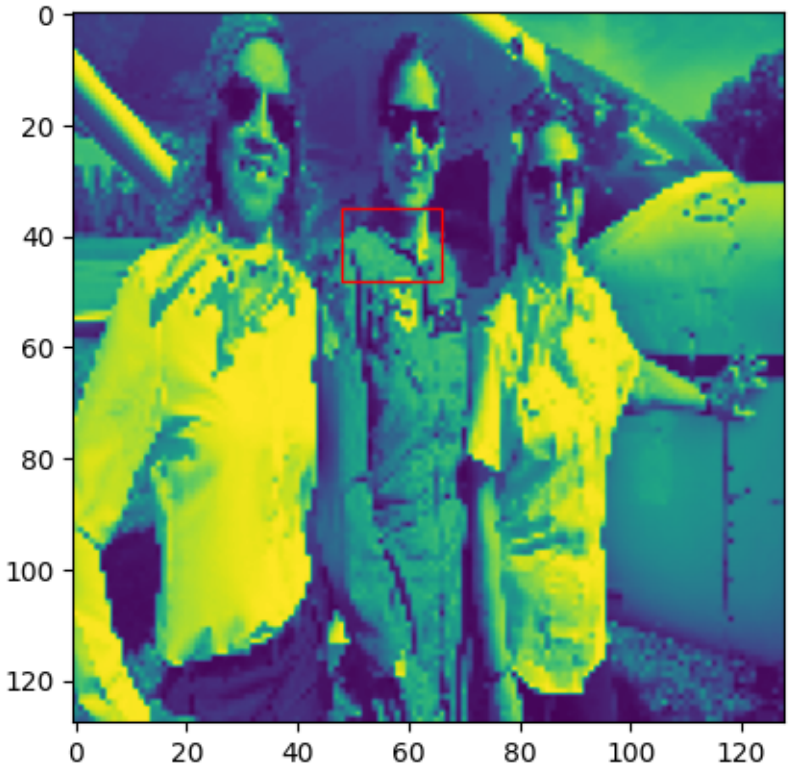


[305]:
```
num = 10 # Both models failed.
model_prediction(num, model1)
model_prediction(num, model2)
```

```
1/1 [==============================] - 0s 35ms/step
```




```
1/1 [==============================] - 0s 56ms/step
```


0
20
40
60
80
100
120
0
20
40
60
80
100
120