

Dataprocess

April 29, 2021

```
[ ]: from google.colab import drive  
drive.mount('/content/drive')
```

```
[ ]: import os  
os.chdir("/content/drive/My Drive/7011gp/")  
!ls
```

```
cleanphoto-wiki.csv  genderpre-cnn.ipynb  wiki_crop  
Dataprocess.ipynb    wikicleanphoto      wiki_crop.tar
```

```
[ ]: !wget https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/static/wiki_crop.tar
```

```
--2021-04-13 03:07:36-- https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-  
wiki/static/wiki_crop.tar  
Resolving data.vision.ee.ethz.ch (data.vision.ee.ethz.ch)... 129.132.52.178,  
2001:67c:10ec:36c2::178  
Connecting to data.vision.ee.ethz.ch  
(data.vision.ee.ethz.ch)|129.132.52.178|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 811315200 (774M) [application/x-tar]  
Saving to: 'wiki_crop.tar'
```

```
wiki_crop.tar      100%[=====] 773.73M 27.2MB/s   in 30s
```

```
2021-04-13 03:08:06 (25.9 MB/s) - 'wiki_crop.tar' saved [811315200/811315200]
```

```
[ ]: !tar -xf wiki_crop.tar
```

```
[ ]: def imshow(path):  
    import cv2  
    import matplotlib.pyplot as plt  
    %matplotlib inline  
  
    image = cv2.imread(path)  
    height, width = image.shape[:2]  
    resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.  
    ↪INTER_CUBIC)
```

```
fig = plt.gcf()
fig.set_size_inches(18, 10)
plt.axis("off")
#plt.rcParams['figure.figsize'] = [10, 5]
plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
```

```
[ ]: def imShowCV(image):
    import cv2
    import matplotlib.pyplot as plt
    %matplotlib inline

    height, width = image.shape[:2]
    resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)

    fig = plt.gcf()
    fig.set_size_inches(18, 10)
    plt.axis("off")
    #plt.rcParams['figure.figsize'] = [10, 5]
    plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
```

```
[ ]: import os

ages = os.scandir('wiki_crop')
length = 0
for age in ages:

    if '.mat' not in str(age.path):

        length+=len(os.listdir(age.path))

print(length)
```

62328

```
[ ]: from shutil import copy2
import cv2
import scipy.io
import csv
wikiMat = scipy.io.loadmat('wiki_crop/wiki.mat')
wikiPlace = wikiMat['wiki'][0][0]
place = wikiPlace
where='wiki_crop'
total = 0
listyList = []
```

```

# firstline=['AGE', 'GENDER', 'NAME', 'PATH']
content_sec=[]
# content_sec.append(firstline)
for i in range(62328):
# for i in range(5):
    if i % 10000 ==0:
        print(i)

    bYear = int(place[0][0][i]/365) #birth year
    taken = place[1][0][i] #photo taken
    path = place[2][0][i][0]
    gender = place[3][0][i] # Female/Male
    name = place[4][0][i] # Name
    faceBox= place[5][0][i] # Face coords
    faceScore = place[6][0][i] #Face score
    secFaceScore = place[7][0][i] #Sec face score

    #Calculating shit
    age = taken - bYear

    faceScore = str(faceScore)
    secFaceScore = str(secFaceScore)

    if 'n' not in faceScore: # n as in Inf; if true, implies that there isn't a ↵
    ↵face in the image
        if float(faceScore)<4 and float(faceScore)>2.5:
            if 'a' in secFaceScore: #a as in NaN; implies that no second face was ↵
    ↵found
            if age >= 0:
                try:
                    gender = int(gender)
                    total += 1

                    # if i > 5000:
                    # if i > 0:
                    thisline=[]
                    thisline.append(age)
                    thisline.append(gender)
                    thisline.append(name[0])
                    thisline.append(path)
                    content_sec.append(thisline)

    # print('-----')

```

```
# print(bYear)
# print(taken)
# print("AGE", age)
# print("NAME", name)
# print("GENDER", gender)
# print(path)
# print(faceBox)
# print(faceScore)
# print(secFaceScore)

# imshow(os.path.join(where, path))
# break

except:
    print('Failed with gender')
    continue

print(total)
```

```
0
Failed with gender
```



```
Failed with gender
Failed with gender
Failed with gender
Failed with gender
20000
Failed with gender
```



```
Failed with gender
60000
Failed with gender
15713
```

```
[ ]: len(content_sec)
```

```
[ ]: 15713
```

```
[ ]: with open('cleanphoto-wiki-2.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(content2)
```

```
[ ]: from google.colab import files
files.download('cleanphoto-wiki-2.csv')
```

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

```
[ ]: a=os.path.join('wikicleanphoto',str(w)+'.jpg')
print(a)
myimg = cv2.imread(os.path.join('wikicleanphoto',str(9)+'.jpg'))
print(myimg)
# # cv2.namedWindow('Image')
# import cv2
# imshow(os.path.join('wikicleanphoto',str(11440)+'.jpg'))
```

```
wikicleanphoto/0.jpg
[[[200 140 94]
 [198 138 92]
 [197 137 91]
 ...
 [190 127 83]
 [191 128 84]
 [191 128 84]]]

[[200 140 94]
 [198 138 92]
 [196 136 90]
 ...
 [191 128 84]
 [191 128 84]
 [191 128 84]]]

[[202 142 96]
 [200 140 94]
 [198 138 92]
 ...
 [191 128 84]
 [191 128 84]
 [191 128 84]]]

...
[[131 100 251]
 [137 105 253]
 [127 92 233]
 ...
 [ 33  93  83]
 [ 32  92  82]
 [ 32  92  82]]]

[[134 102 251]
 [132  98 246]
 [124  89 230]
 ...]
```

```
[ 23  82  74]
[ 19  79  69]
[ 17  77  67]]
```

```
[[133 101 250]
 [128  95 240]
 [126  88 230]
 ...
 [ 25  84  76]
 [ 19  78  70]
 [ 16  76  66]]]
```

```
[ ]: content2=content_sec
for j in range(11447,27160):
    content2[j-11447][3]='wikicleanphoto/'+str(j)+'.jpg'
```

```
[ ]: for w in range(len(content_sec)):
    img = cv2.imread(os.path.join('wiki_crop',content_sec[w][3]))
    cv2.imwrite(os.path.join('wikicleanphoto',str(w+11447)+'.jpg'), img)
```

face detection

April 29, 2021

```
[ ]: import os
os.chdir('/content/gdrive/My Drive/7011/group project/')

[ ]: from mtcnn import MTCNN
import cv2
from PIL import Image

[ ]: from os import listdir
from os.path import isfile, join
# mypath = '/content/gdrive/My Drive/7011/group project/photos/wikicleanphoto/'
# onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]

path2 = '/content/gdrive/My Drive/7011/group project/photos/crop/'
already = [f for f in listdir(path2) if isfile(join(path2, f))]

[ ]: '10703'_crop.jpg' in already

[ ]: True

[ ]: import warnings

[ ]: import os
import tensorflow as tf
tf.get_logger().setLevel('INFO')
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"

[ ]: for i in range(len(onlyfiles)):
    import warnings
    warnings.filterwarnings('ignore')
    name = onlyfiles[i].strip('.jpg')
    if name+'_crop.jpg' in already:
        continue
    else:
        detector = MTCNN()
        image = cv2.cvtColor(cv2.imread("/content/gdrive/My Drive/7011/group_
→project/photos/wikicleanphoto/" + onlyfiles[i]), cv2.COLOR_BGR2RGB)
```

```

result = detector.detect_faces(image)

# Result is an array with all the bounding boxes detected. We know that ↵
→for 'ivan.jpg' there is only one.
if len(result) != 0:
    bounding_box = result[0]['box']
else:
    continue

im = Image.open("/content/gdrive/My Drive/7011/group project/photos/
→wikicleanphoto/" + onlyfiles[i])

crop_rectangle = (bounding_box[0], bounding_box[1], ↵
→bounding_box[0] + bounding_box[2], bounding_box[1] + bounding_box[3])
cropped_im = im.crop(crop_rectangle)

cropped_im.save("/content/gdrive/My Drive/7011/group project/photos/
→crop/" + name + '_crop.jpg')

```

Streaming output truncated to the last 5000 lines.

WARNING:tensorflow:7 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f445863f200> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:7 out of the last 13 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4462e23710> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:8 out of the last 14 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4462e23680> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument

shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:8 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4458189950> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:9 out of the last 12 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4458189950> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:9 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4458189950> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:5 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4458e374d0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:5 out of the last 14 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f44655d44d0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of

tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:6 out of the last 15 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f44655d44d0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:6 out of the last 12 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f446548b680> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:7 out of the last 13 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4459f1e320> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:8 out of the last 14 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4461ca98c0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:8 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4461ca98c0> triggered tf.function retracing. Tracing is expensive and the excessive number

of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:8 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4461ca98c0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:7 out of the last 13 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4458c16320> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:7 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4424faecb0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:7 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f445852b8c0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:7 out of the last 11 calls to <function

Model.make_predict_function.<locals>.predict_function at 0x7f445852b8c0>
triggered tf.function retracing. Tracing is expensive and the excessive number
of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2)
passing tensors with different shapes, (3) passing Python objects instead of
tensors. For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for more details.
WARNING:tensorflow:7 out of the last 11 calls to <function
Model.make_predict_function.<locals>.predict_function at 0x7f445852b8c0>
triggered tf.function retracing. Tracing is expensive and the excessive number
of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2)
passing tensors with different shapes, (3) passing Python objects instead of
tensors. For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for more details.
WARNING:tensorflow:7 out of the last 14 calls to <function
Model.make_predict_function.<locals>.predict_function at 0x7f4461d0db00>
triggered tf.function retracing. Tracing is expensive and the excessive number
of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2)
passing tensors with different shapes, (3) passing Python objects instead of
tensors. For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for more details.
WARNING:tensorflow:7 out of the last 11 calls to <function
Model.make_predict_function.<locals>.predict_function at 0x7f4458ccf560>
triggered tf.function retracing. Tracing is expensive and the excessive number
of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2)
passing tensors with different shapes, (3) passing Python objects instead of
tensors. For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for more details.
WARNING:tensorflow:8 out of the last 12 calls to <function
Model.make_predict_function.<locals>.predict_function at 0x7f445891c170>
triggered tf.function retracing. Tracing is expensive and the excessive number
of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2)
passing tensors with different shapes, (3) passing Python objects instead of
tensors. For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has experimental_relax_shapes=True option that relaxes argument
shapes that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and

https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:8 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f445891c170> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:8 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f445891c170> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:5 out of the last 16 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4458372dd0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:6 out of the last 17 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4458372dd0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:5 out of the last 13 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4464f54c20> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument

shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:6 out of the last 14 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4464f54c20> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:5 out of the last 13 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f445e246320> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:6 out of the last 14 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f445e246320> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:6 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4459f1e290> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:6 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f4464720440> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of

tensors. For (1), please define your `@tf.function` outside of the loop. For (2), `@tf.function` has `experimental_relax_shapes=True` option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

[]:

Machine Learning

Preparing Data

```
In [1]: # This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import math
import cv2
import matplotlib.pyplot as plt
import os
import seaborn as sns
import umap
from PIL import Image
from scipy import misc
from os import listdir
from os.path import isfile, join
import numpy as np
from scipy import misc
from random import shuffle
from collections import Counter
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.utils.np_utils import to_categorical
```

```
In [2]: os.chdir('crop')
```

```
In [3]: os.listdir()[:5]
```

```
Out[3]: ['18030_crop.jpg',
 '20423_crop.jpg',
 '2378_crop.jpg',
 '25683_crop.jpg',
 '22742_crop.jpg']
```

```
In [4]: im =Image.open('18030_crop.jpg').resize((128,128))
im
```



```
In [5]: imageinfo = pd.read_csv("../wikifinal_specificage.csv")
```

```
In [6]: len(imageinfo)
```

```
Out[6]: 27123
```

```
In [7]: imageinfo[:5]
```

```
Out[7]:
```

	AGE	GENDER	NAME	PATH
0	27	1	Sami JauhojÄorvi	1_crop.jpg
1	59	1	Marc Okrand	2_crop.jpg
2	50	0	Krista Tippett	3_crop.jpg
3	32	1	Bernie Whitebear	4_crop.jpg
4	41	1	Carl Greenberg	5_crop.jpg

```
In [8]: from sklearn.utils import shuffle
imageinfo = shuffle(imageinfo)
```

```
In [9]: imageinfo[:5]
```

```
Out[9]:
```

	AGE	GENDER	NAME	PATH
1537	14	0	Miwa Fukuhara	1538_crop.jpg
166	46	1	Max Keiser	167_crop.jpg
18385	33	0	Dolcenera	18403_crop.jpg
4484	26	1	Mieczysław Wilczewski	4485_crop.jpg
18267	36	1	Larry Blyden	18284_crop.jpg

```
In [10]: class_label = [(0,18), (18,30), (30,40), (40,60), (60,infinity)]
```

```
In [11]: age_classes = []
Y_age = imageinfo["AGE"]
for i in Y_age:
    if i <= 18:
        age_classes.append(0)
    elif i >18 and i <= 30:
        age_classes.append(1)
    elif i >30 and i <= 40:
        age_classes.append(2)
    elif i >40 and i <= 60:
        age_classes.append(3)
    elif i > 60:
        age_classes.append(4)
```

```
In [12]: age_classes[:10]
```

```
Out[12]: [0, 3, 2, 1, 2, 2, 1, 1, 3, 4]
```

```
In [13]: gender_classes = list(imageinfo["GENDER"])
```

```
In [14]: gender_classes[:10]
```

```
Out[14]: [0, 1, 0, 1, 1, 0, 0, 1, 1, 0]
```

Convert Images to Vectors

```
In [15]: import imageio
```

```
In [16]: def convertImage(filename):
    face = imageio.imread(filename)
    face = cv2.resize(face, (32, 32))
    return face
X_data = list(map(convertImage, imageinfo["PATH"]))
```

```
In [17]: X = np.squeeze(X_data)
```

```
In [18]: X.shape
```

```
Out[18]: (27123, 32, 32, 3)
```

```
In [19]: # normalize data
X = X.astype('float32')
X /= 255
```

```
In [20]: # categorize age classes
categorical_labels = to_categorical(age_classes, num_classes=5)
```

```
In [21]: categorical_labels[:5]
```

```
Out[21]: array([[1., 0., 0., 0., 0.],
                [0., 0., 0., 1., 0.],
                [0., 0., 1., 0., 0.],
                [0., 1., 0., 0., 0.],
                [0., 0., 1., 0., 0.]], dtype=float32)
```

```
In [22]: len(X)
```

```
Out[22]: 27123
```

Train and Test Split

```
In [23]: (x_train, y_train_age, y_train_true_age, y_train_gender), (x_test, y_test_age, y_test_true_age, y_test_gender) =
#(x_valid , y_valid) = (x_test[25000:], y_test[25000:])
#(x_test, y_test) = (x_test[:25000], y_test[:25000])

In [24]: len(x_train)+len(x_test) == len(X)

Out[24]: True

In [25]: len(x_train)

Out[25]: 25000

In [26]: len(x_test)

Out[26]: 2123
```

Eigenface Encoder

```
In [27]: from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix, classification_report

In [28]: cv2.cvtColor(x_train[0], cv2.COLOR_BGR2GRAY).shape

Out[28]: (32, 32)

In [29]: x_train_temp = np.array([cv2.cvtColor(x_t, cv2.COLOR_BGR2GRAY).flatten() for x_t in x_train])
x_train_temp.shape

Out[29]: (25000, 1024)

In [30]: x_train_df = pd.DataFrame(x_train_temp)
x_train_df[:5]

Out[30]:
```

	0	1	2	3	4	5	6	7	8	9	...	1014	1015	1016	1017
0	0.109482	0.115749	0.111871	0.118137	0.131051	0.120906	0.171439	0.182075	0.260718	0.239514	...	0.660267	0.758094	0.190835	0.861659
1	0.004024	0.026996	0.072494	0.230714	0.282604	0.338569	0.358220	0.337886	0.321027	0.311286	...	0.370902	0.358690	0.332133	0.351573
2	0.302384	0.399969	0.318231	0.287078	0.292451	0.272565	0.293024	0.282557	0.423286	0.310204	...	0.439980	0.428620	0.501224	0.624082
3	0.137255	0.466667	0.768627	0.839216	0.862745	0.866667	0.870588	0.901961	0.905882	0.901961	...	0.882353	0.792157	0.729412	0.533333
4	0.329412	0.282353	0.215686	0.149020	0.062745	0.062745	0.098039	0.223529	0.231373	0.258824	...	0.227451	0.278431	0.250980	0.239216

5 rows × 1024 columns

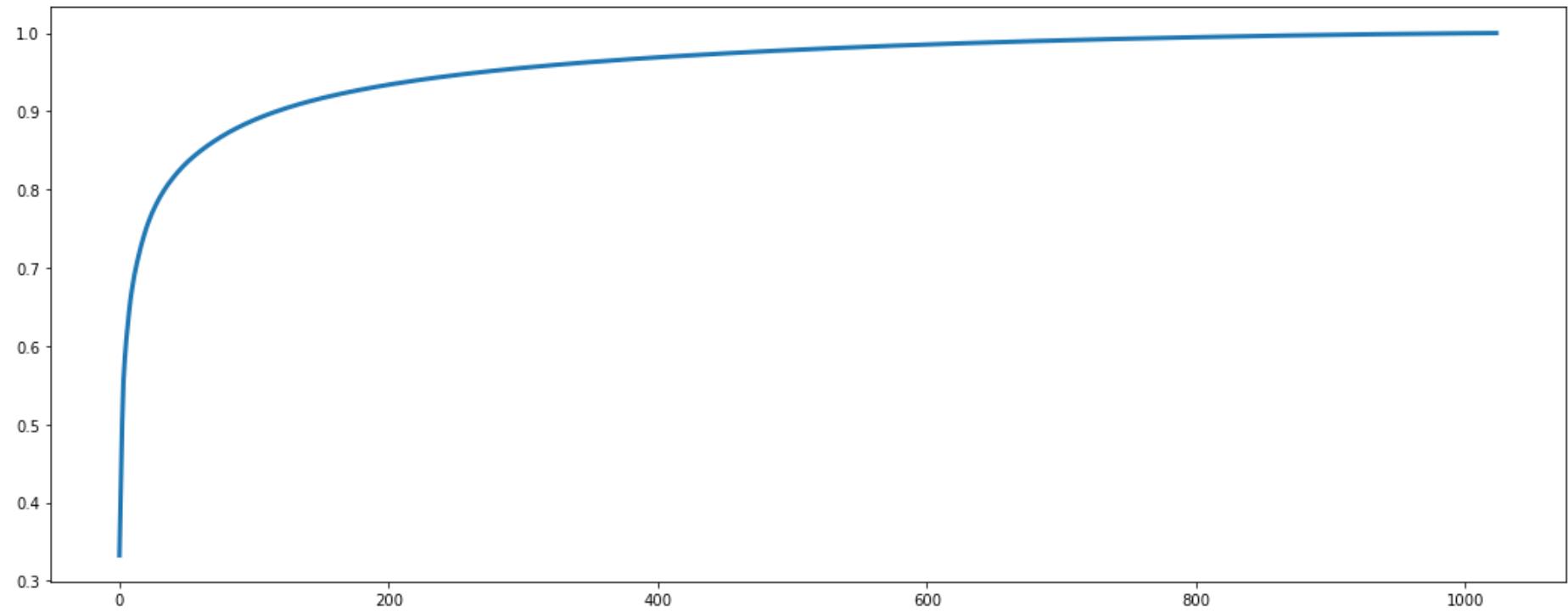
```
In [31]: def plot_faces(pixels):
    fig, axes = plt.subplots(5, 5, figsize=(6, 6))
    for i, ax in enumerate(axes.flat):
        ax.imshow(np.array(pixels)[i].reshape(32, 32), cmap='gray')
    plt.show()
# for x_t in x_train_temp:
#     plot_faces(x_t)
#     break
plot_faces(x_train_df)
```



PCA

```
In [32]: pca = PCA().fit(x_train_df)
plt.figure(figsize=(18, 7))
plt.plot(pca.explained_variance_ratio_.cumsum(), lw=3)
```

```
Out[32]: <matplotlib.lines.Line2D at 0x7fb753a22220>
```



```
In [33]: pca.explained_variance_ratio_.cumsum()[600]
```

```
Out[33]: 0.9853523
```

```
In [34]: representPercesntage = 0.9999 # This significantly affects accuracy
#representPercesntage = 0.9999
np.where(pca.explained_variance_ratio_.cumsum() > representPercesntage)[0][0:10]
```

```
Out[34]: array([1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023])
```

```
In [35]: n_com = np.where(pca.explained_variance_ratio_.cumsum() > representPercesntage)[0][0]
```

```
In [36]: pca = PCA(n_components=n_com).fit(x_train_df)
```

```
In [37]: x_train_pca = pca.transform(x_train_df)
```

```
In [38]: x_train_pca.shape
```

```
Out[38]: (25000, 1015)
```

```
In [39]: y_train_sklearn = np.array([np.where(yt == 1)[0][0] for yt in y_train_age])
```

```
In [40]: y_train_sklearn[:5]
```

```
Out[40]: array([0, 3, 2, 1, 2])
```

Age

```
In [41]: import time
```

```
def train_time(start_time, end_time):
    elapsed_time = end_time - start_time
    elapsed_mins = int(elapsed_time / 60)
    elapsed_secs = int(elapsed_time - (elapsed_mins * 60))
    return elapsed_mins, elapsed_secs
```

```
In [42]: start_time = time.time()
age_classifier = SVC().fit(x_train_pca, y_train_sklearn)
end_time = time.time()
train_mins, train_secs = train_time(start_time, end_time)
print(f'Train Time: {train_mins}m {train_secs}s')
```

```
Train Time: 9m 54s
```

```
In [43]: age_predictions = age_classifier.predict(x_train_pca)
```

```
In [44]: age_target_names = [str(l) for l in range(5)]
print(classification_report(y_train_sklearn, age_predictions, target_names=age_target_names))
```

	precision	recall	f1-score	support
0	1.00	0.00	0.00	1313
1	0.57	0.93	0.71	10014
2	0.95	0.14	0.24	4818
3	0.62	0.64	0.63	6228
4	0.81	0.45	0.58	2627
accuracy			0.61	25000
macro avg	0.79	0.43	0.43	25000
weighted avg	0.70	0.61	0.55	25000

```
In [45]: AgeTruePrediction = 0
for i in range(len(age_predictions)):
    if age_predictions[i] == y_train_sklearn[i]:
        AgeTruePrediction += 1
print(f"Train Accuracy: {AgeTruePrediction/len(age_predictions)}")
```

Train Accuracy: 0.60628

```
In [46]: # Preprocessing pipeline
x_test_temp = np.array([cv2.cvtColor(x_t, cv2.COLOR_BGR2GRAY).flatten() for x_t in x_test])
x_test_df = pd.DataFrame(x_test_temp)
x_test_pca = pca.transform(x_test_df)
```

```
In [47]: y_test_sklearn = np.array([np.where(yt == 1)[0][0] for yt in y_test_age])
```

```
In [48]: age_predictions = age_classifier.predict(x_test_pca)
print(classification_report(y_test_sklearn, age_predictions))
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	93
1	0.47	0.87	0.61	777
2	0.63	0.03	0.05	452
3	0.42	0.43	0.42	551
4	0.74	0.28	0.41	250
accuracy			0.47	2123
macro avg	0.45	0.32	0.30	2123
weighted avg	0.50	0.47	0.39	2123

```
/Users/sujingyi/opt/anaconda3/envs/d2l/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
/Users/sujingyi/opt/anaconda3/envs/d2l/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
/Users/sujingyi/opt/anaconda3/envs/d2l/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
```

```
In [49]: TruePrediction = 0
loss = 0
N = len(age_predictions)
for i in range(len(age_predictions)):
    if age_predictions[i] == y_test_sklearn[i]:
        TruePrediction += 1
    else:
        loss += (y_test_sklearn[i] - age_predictions[i])**2
print(f"Test Accuracy: {TruePrediction/len(age_predictions)}")
print(f"Test Loss: {(loss/N)**0.5}")
```

Test Accuracy: 0.4686764013188884
Test Loss: 1.1705685821274916

```
In [51]: labels = class_label

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    age_predict_index = age_predictions[index]
    age_true_index = np.argmax(y_test_age[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[age_predict_index],
                                  labels[age_true_index]),
                 color="green" if age_predict_index == age_true_index else "red"))
plt.show()
```



Gender

```
In [52]: start_time = time.time()
gender_classifier = SVC().fit(x_train_pca,y_train_gender)
end_time = time.time()
train_mins, train_secs = train_time(start_time, end_time)
print(f'Train Time: {train_mins}m {train_secs}s')
```

Train Time: 5m 19s

```
In [53]: gender_predictions = gender_classifier.predict(x_train_pca)
gender_target_names = [str(l) for l in range(2)]
print(classification_report(y_train_gender, gender_predictions, target_names=gender_target_names))
```

	precision	recall	f1-score	support
0	0.95	0.68	0.79	6946
1	0.89	0.99	0.93	18054
accuracy			0.90	25000
macro avg	0.92	0.83	0.86	25000
weighted avg	0.91	0.90	0.90	25000

```
In [54]: GenderTruePrediction = 0
for i in range(len(gender_predictions)):
    if gender_predictions[i] == y_train_gender[i]:
        GenderTruePrediction += 1
print(f'Train Accuracy: {GenderTruePrediction/len(gender_predictions)}')
```

Train Accuracy: 0.90072

```
In [55]: # Preprocessing pipeline
x_test_temp = np.array([cv2.cvtColor(x_t, cv2.COLOR_BGR2GRAY).flatten() for x_t in x_test])
x_test_df = pd.DataFrame(x_test_temp)
x_test_pca = pca.transform(x_test_df)

gender_predictions = gender_classifier.predict(x_test_pca)

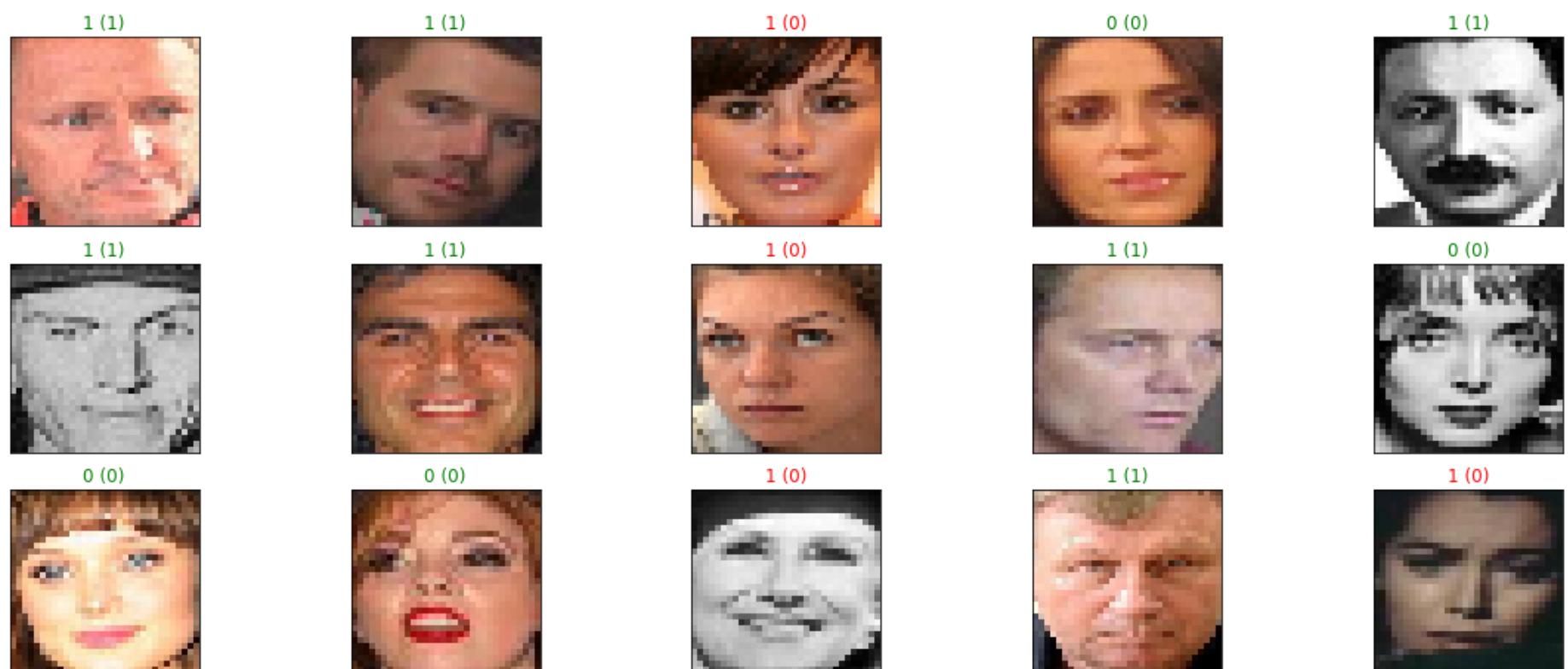
print(classification_report(y_test_gender, gender_predictions))
```

	precision	recall	f1-score	support
0	0.81	0.56	0.66	557
1	0.86	0.95	0.90	1566
accuracy			0.85	2123
macro avg	0.84	0.76	0.78	2123
weighted avg	0.85	0.85	0.84	2123

```
In [56]: TruePrediction = 0
loss = 0
N = len(gender_predictions)
for i in range(len(gender_predictions)):
    if gender_predictions[i] == y_test_gender[i]:
        TruePrediction += 1
    else:
        loss += (y_test_gender[i] - gender_predictions[i])**2
print(f"Test Accuracy: {TruePrediction/len(gender_predictions)}")
print(f"Test Loss: {(loss/N)**0.5}")
```

Test Accuracy: 0.8511540273198305
Test Loss: 0.385805615148574

```
In [57]: # Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    gender_predict_index = gender_predictions[index]
    gender_true_index = y_test_gender[index]
    # Set the title for each image
    ax.set_title("{} ({})".format(gender_predict_index,
                                  gender_true_index),
                  color=("green" if gender_predict_index == gender_true_index else "red"))
plt.show()
```



```
In [59]: # Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    gender_predict_index = gender_predictions[index]
    gender_true_index = y_test_gender[index]
    # Set the title for each image
    ax.set_title("Age: {} ({} {}) | Gender:{} ({} {})".format(labels[age_predict_index],
                                                               labels[age_true_index],
                                                               gender_predict_index,
                                                               gender_true_index),
                color="green" if age_predict_index == age_true_index and gender_predict_index == gender_true_index
                else "red"))
plt.show()
```



SVC

SVC Linear

Age

```
In [60]: #SVC linear
start_time = time.time()
classifier = SVC(kernel = 'linear').fit(x_train_pca,y_train_sklearn)
end_time = time.time()
train_mins, train_secs = train_time(start_time, end_time)
print(f'Train Time: {train_mins}m {train_secs}s')
```

Train Time: 17m 46s

```
In [62]: age_predictions = classifier.predict(x_train_pca)
```

```
In [63]: age_target_names = [str(l) for l in range(5)]
print(classification_report(y_train_sklearn, age_predictions, target_names=age_target_names))
```

	precision	recall	f1-score	support
0	0.61	0.01	0.02	1313
1	0.50	0.86	0.64	10014
2	0.58	0.03	0.05	4818
3	0.48	0.46	0.47	6228
4	0.63	0.41	0.49	2627
accuracy			0.51	25000
macro avg	0.56	0.35	0.33	25000
weighted avg	0.53	0.51	0.43	25000

```
In [64]: AgeTruePrediction = 0
for i in range(len(age_predictions)):
    if age_predictions[i] == y_train_sklearn[i]:
        AgeTruePrediction += 1
print(f"Train Accuracy: {AgeTruePrediction/len(age_predictions)}")
```

Train Accuracy: 0.50748

```
In [65]: # Preprocessing pipeline
x_test_temp = np.array([cv2.cvtColor(x_t, cv2.COLOR_BGR2GRAY).flatten() for x_t in x_test])
x_test_df = pd.DataFrame(x_test_temp)
x_test_pca = pca.transform(x_test_df)

y_test_sklearn = np.array([np.where(yt == 1)[0][0] for yt in y_test_age])
```

```
In [66]: age_predictions = classifier.predict(x_test_pca)
print(classification_report(y_test_sklearn, age_predictions))
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	93
1	0.44	0.84	0.58	777
2	0.38	0.01	0.03	452
3	0.38	0.35	0.37	551
4	0.56	0.27	0.37	250
accuracy			0.43	2123
macro avg	0.35	0.30	0.27	2123
weighted avg	0.41	0.43	0.36	2123

```
In [67]: labels = class_label
```

```
In [69]: TruePrediction = 0
loss = 0
N = len(age_predictions)
for i in range(N):
    if age_predictions[i] == y_test_sklearn[i]:
        TruePrediction += 1
    else:
        loss += (y_test_sklearn[i] - age_predictions[i])**2
print(f"Test Accuracy: {TruePrediction/len(age_predictions)}")
print(f"Test Loss: {(loss/N)**0.5}")
```

Test Accuracy: 0.4347621290626472
Test Loss: 1.2832483511246584

```
In [70]: # Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    age_predict_index = age_predictions[index]
    age_true_index = np.argmax(y_test_age[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[age_predict_index],
                                  labels[age_true_index]),
                 color="green" if age_predict_index == age_true_index else "red"))
plt.show()
```



Gender

```
In [71]: #SVC linear
start_time = time.time()
gender_classifier = SVC(kernel = 'linear').fit(x_train_pca,y_train_gender)
end_time = time.time()
train_mins, train_secs = train_time(start_time, end_time)
print(f'Train Time: {train_mins}m {train_secs}s')
```

Train Time: 14m 9s

```
In [72]: gender_predictions = gender_classifier.predict(x_train_pca)
gender_target_names = [str(l) for l in range(2)]
print(classification_report(y_train_gender, gender_predictions, target_names=gender_target_names))
```

	precision	recall	f1-score	support
0	0.77	0.51	0.62	6946
1	0.83	0.94	0.88	18054
accuracy			0.82	25000
macro avg	0.80	0.73	0.75	25000
weighted avg	0.82	0.82	0.81	25000

```
In [73]: GenderTruePrediction = 0
for i in range(len(gender_predictions)):
    if gender_predictions[i] == y_train_gender[i]:
        GenderTruePrediction += 1
print(f"Train Accuracy: {GenderTruePrediction/len(gender_predictions)}")
```

Train Accuracy: 0.82268

```
In [91]: # Preprocessing pipeline
x_test_temp = np.array([cv2.cvtColor(x_t, cv2.COLOR_BGR2GRAY).flatten() for x_t in x_test])
x_test_df = pd.DataFrame(x_test_temp)
x_test_pca = pca.transform(x_test_df)
```

```
In [92]: gender_predictions = gender_classifier.predict(x_test_pca)
print(classification_report(y_test_gender, gender_predictions))
```

	precision	recall	f1-score	support
0	0.66	0.48	0.56	557
1	0.83	0.91	0.87	1566
accuracy			0.80	2123
macro avg	0.75	0.70	0.71	2123
weighted avg	0.79	0.80	0.79	2123

```
In [93]: TruePrediction = 0
loss = 0
N = len(gender_predictions)
for i in range(N):
    if gender_predictions[i] == y_test_gender[i]:
        TruePrediction += 1
    else:
        loss += (y_test_gender[i] - gender_predictions[i])**2
print(f"Test Accuracy: {TruePrediction/len(gender_predictions)}")
print(f"Test Loss: {(loss/N)**0.5}")
```

Test Accuracy: 0.7993405558172397
Test Loss: 0.44795026976525

```
In [94]: # Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    gender_predict_index = gender_predictions[index]
    gender_true_index = y_test_gender[index]
    # Set the title for each image
    ax.set_title("{} ({})".format(gender_predict_index,
                                  gender_true_index),
                color="green" if gender_predict_index == gender_true_index else "red"))
plt.show()
```



```
In [95]: # Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    # Set the title for each image
    ax.set_title("Age: {} ({}) | Gender:{} ({})".format(labels[age_predict_index],
                                                       labels[age_true_index],
                                                       gender_predict_index,
                                                       gender_true_index),
                color="green"
                if age_predict_index == age_true_index and gender_predict_index == gender_true_index
                else "red"))
plt.show()
```



SVC Poly

Age

```
In [83]: #SVC poly
start_time = time.time()
classifier = SVC(kernel = 'poly', gamma='auto', class_weight='balanced').fit(x_train_pca,y_train_sklearn)
end_time = time.time()
train_mins, train_secs = train_time(start_time, end_time)
print(f'Train Time: {train_mins}m {train_secs}s')
```

Train Time: 12m 6s

```
In [96]: age_predictions = classifier.predict(x_train_pca)
```

```
In [97]: age_target_names = [str(l) for l in range(5)]
print(classification_report(y_train_sklearn, age_predictions, target_names=age_target_names))
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1313
1	0.00	0.00	0.00	10014
2	0.00	0.00	0.00	4818
3	0.25	1.00	0.40	6228
4	0.00	0.00	0.00	2627
accuracy			0.25	25000
macro avg	0.05	0.20	0.08	25000
weighted avg	0.06	0.25	0.10	25000

```
/Users/sujingyi/opt/anaconda3/envs/d21/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
/Users/sujingyi/opt/anaconda3/envs/d21/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
/Users/sujingyi/opt/anaconda3/envs/d21/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
```

```
In [98]: AgeTruePrediction = 0
for i in range(len(age_predictions)):
    if age_predictions[i] == y_train_sklearn[i]:
        AgeTruePrediction += 1
print(f"Train Accuracy: {AgeTruePrediction/len(age_predictions)}")
```

Train Accuracy: 0.24912

```
In [99]: # Preprocessing pipeline
x_test_temp = np.array([cv2.cvtColor(x_t, cv2.COLOR_BGR2GRAY).flatten() for x_t in x_test])
x_test_df = pd.DataFrame(x_test_temp)
x_test_pca = pca.transform(x_test_df)

y_test_sklearn = np.array([np.where(yt == 1)[0][0] for yt in y_test_age])
```

```
In [100]: age_predictions = classifier.predict(x_test_pca)
print(classification_report(y_test_sklearn, age_predictions))
```

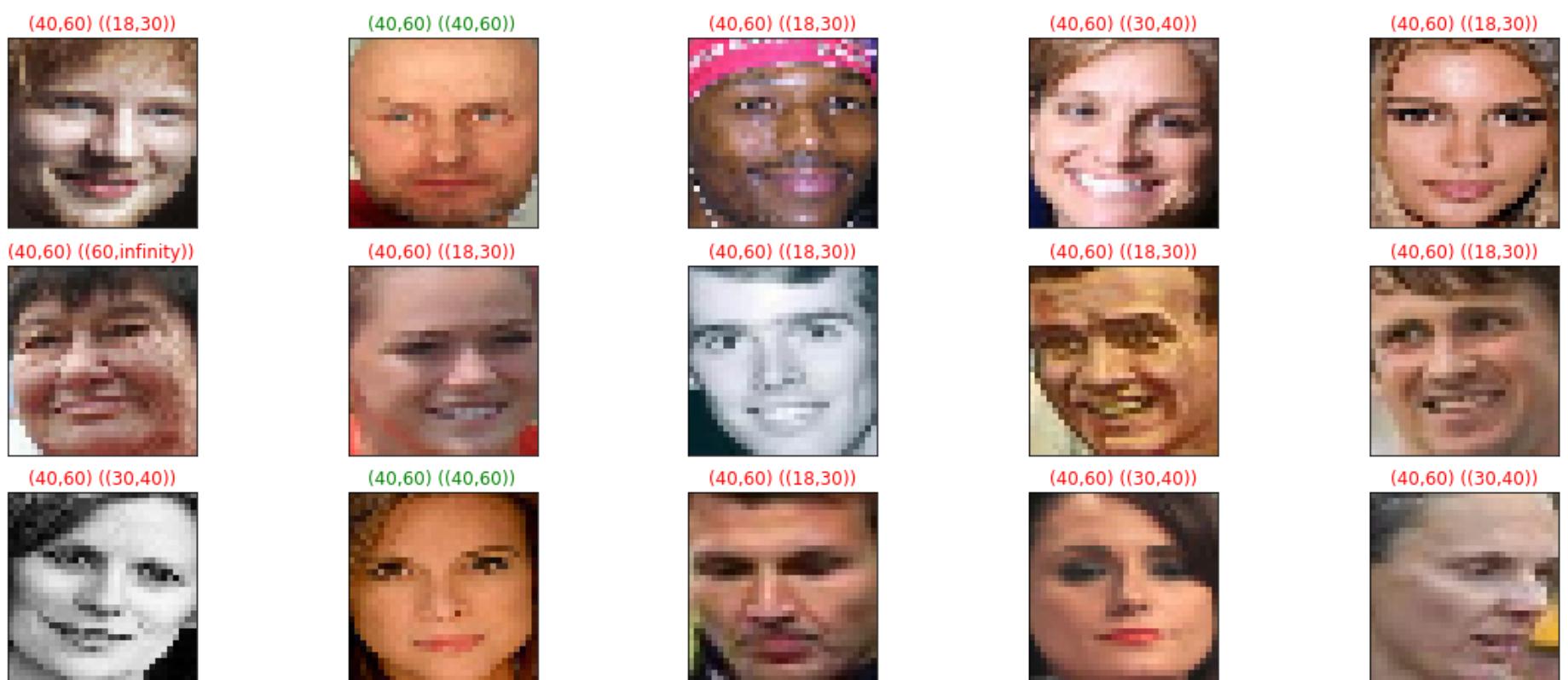
	precision	recall	f1-score	support
0	0.00	0.00	0.00	93
1	0.00	0.00	0.00	777
2	0.00	0.00	0.00	452
3	0.26	1.00	0.41	551
4	0.00	0.00	0.00	250
accuracy			0.26	2123
macro avg	0.05	0.20	0.08	2123
weighted avg	0.07	0.26	0.11	2123

```
/Users/sujingyi/opt/anaconda3/envs/d21/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
/Users/sujingyi/opt/anaconda3/envs/d21/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
/Users/sujingyi/opt/anaconda3/envs/d21/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
```

```
In [101]: TruePrediction = 0
loss = 0
N = len(age_predictions)
for i in range(N):
    if age_predictions[i] == y_test_sklearn[i]:
        TruePrediction += 1
    else:
        loss += (y_test_sklearn[i] - age_predictions[i])**2
print(f"Test Accuracy: {TruePrediction/len(age_predictions)}")
print(f"Test Loss: {(loss/N)**0.5}")
```

Test Accuracy: 0.25953838907206783
Test Loss: 1.47948763266372

```
In [102]: # Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    age_predict_index = age_predictions[index]
    age_true_index = np.argmax(y_test_age[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[age_predict_index],
                                  labels[age_true_index]),
                color=("green" if age_predict_index == age_true_index else "red"))
plt.show()
```



Gender

```
In [103]: #SVC poly
start_time = time.time()
gender_classifier = SVC(kernel = 'poly', gamma='auto', class_weight='balanced').fit(x_train_pca,y_train_gender)
end_time = time.time()
train_mins, train_secs = train_time(start_time, end_time)
print(f'Train Time: {train_mins}m {train_secs}s')
```

Train Time: 6m 23s

```
In [104]: gender_predictions = gender_classifier.predict(x_train_pca)
gender_target_names = [str(l) for l in range(2)]
print(classification_report(y_train_gender, gender_predictions, target_names=gender_target_names))
```

	precision	recall	f1-score	support
0	0.47	0.00	0.00	6946
1	0.72	1.00	0.84	18054
accuracy			0.72	25000
macro avg	0.60	0.50	0.42	25000
weighted avg	0.65	0.72	0.61	25000

```
In [105]: GenderTruePrediction = 0
for i in range(len(gender_predictions)):
    if gender_predictions[i] == y_train_gender[i]:
        GenderTruePrediction += 1
print(f"Train Accuracy: {GenderTruePrediction/len(gender_predictions)}")
```

Train Accuracy: 0.72212

```
In [106]: # Preprocessing pipeline
x_test_temp = np.array([cv2.cvtColor(x_t, cv2.COLOR_BGR2GRAY).flatten() for x_t in x_test])
x_test_df = pd.DataFrame(x_test_temp)
x_test_pca = pca.transform(x_test_df)
```

```
In [107]: gender_predictions = gender_classifier.predict(x_test_pca)
print(classification_report(y_test_gender, gender_predictions))
```

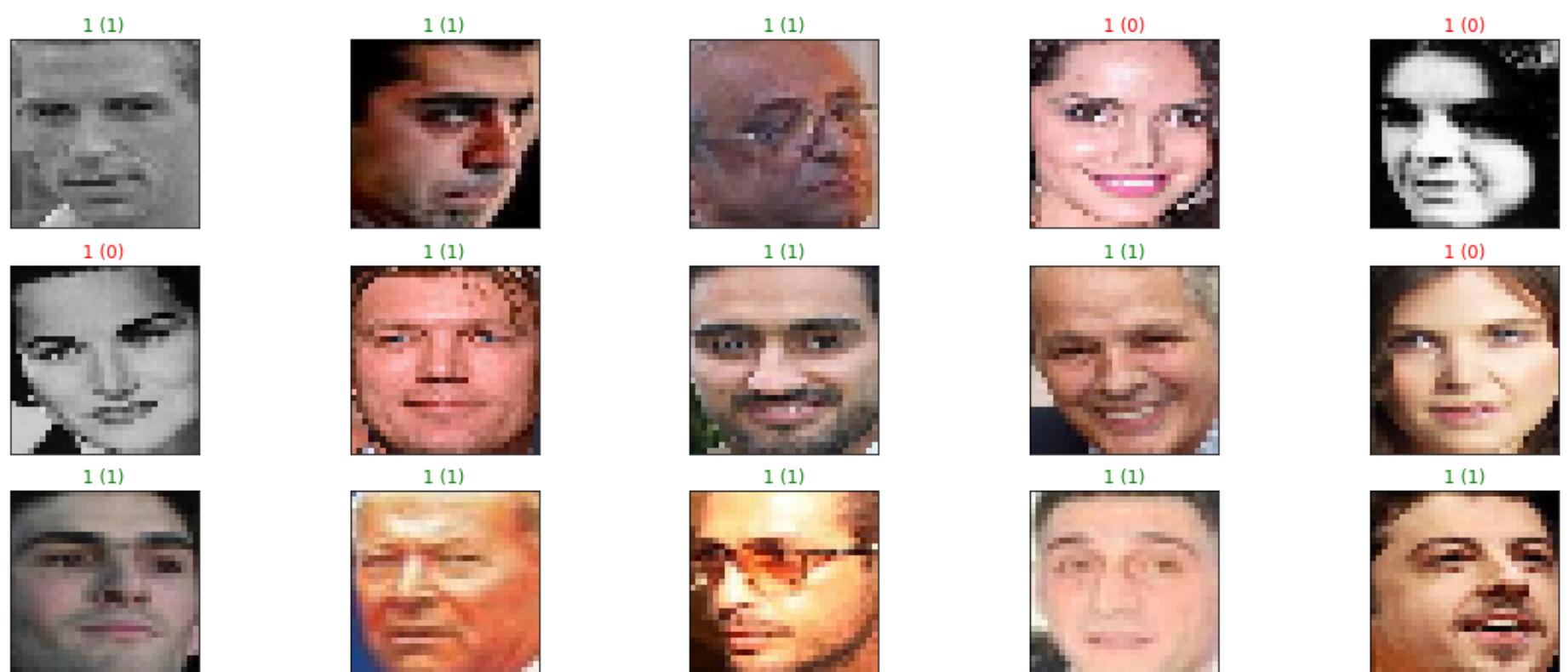
	precision	recall	f1-score	support
0	1.00	0.00	0.00	557
1	0.74	1.00	0.85	1566
accuracy			0.74	2123
macro avg	0.87	0.50	0.43	2123
weighted avg	0.81	0.74	0.63	2123

```
In [109]: TruePrediction = 0
loss = 0
N = len(gender_predictions)
for i in range(N):
    if gender_predictions[i] == y_test_gender[i]:
        TruePrediction += 1
    else:
        loss += (y_test_gender[i] - gender_predictions[i])**2
print(f"Test Accuracy: {TruePrediction/len(gender_predictions)}")
print(f"Test Loss: {(loss/N)**0.5}")
```

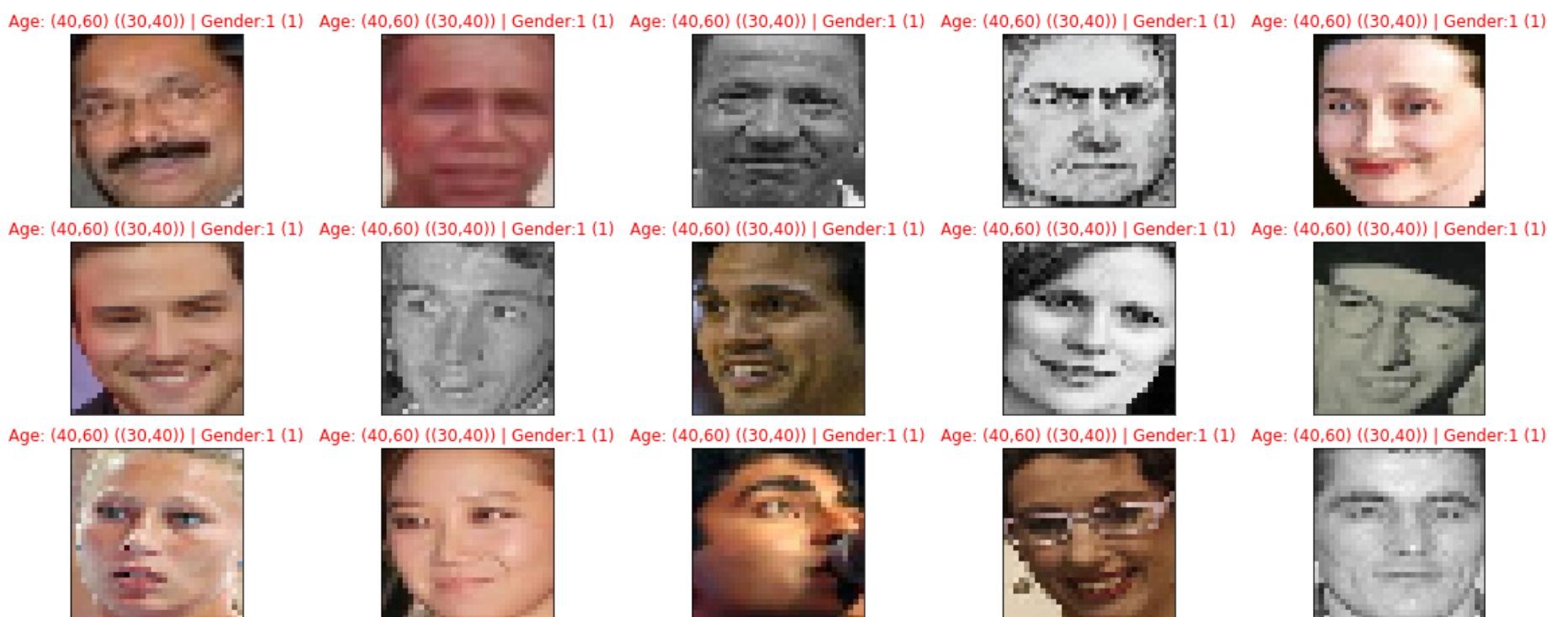
Test Accuracy: 0.7381064531323599

Test Loss: 0.511755358416148

```
In [111]: # Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    gender_predict_index = gender_predictions[index]
    gender_true_index = y_test_gender[index]
    # Set the title for each image
    ax.set_title("{} ({})".format(gender_predict_index,
                                  gender_true_index),
                color=("green" if gender_predict_index == gender_true_index else "red"))
plt.show()
```



```
In [112]: # Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    # Set the title for each image
    ax.set_title("Age: {} ({} {}) | Gender:{} ({} {})".format(labels[age_predict_index],
                                                               labels[age_true_index],
                                                               gender_predict_index,
                                                               gender_true_index),
                color="green"
                if age_predict_index == age_true_index and gender_predict_index == gender_true_index
                else "red"))
plt.show()
```



Random Forest

Age

```
In [113]: from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
```

```
In [114]: start_time = time.time()
clf = RandomForestClassifier(max_depth=100, random_state=100)
clf.fit(x_train_pca, y_train_sklearn)
end_time = time.time()
train_mins, train_secs = train_time(start_time, end_time)
print(f'Train Time: {train_mins}m {train_secs}s')
```

Train Time: 1m 18s

```
In [115]: age_predictions = clf.predict(x_train_pca)
print(classification_report(y_train_sklearn, age_predictions))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1313
1	1.00	1.00	1.00	10014
2	1.00	1.00	1.00	4818
3	1.00	1.00	1.00	6228
4	1.00	1.00	1.00	2627
accuracy			1.00	25000
macro avg	1.00	1.00	1.00	25000
weighted avg	1.00	1.00	1.00	25000

```
In [116]: TruePrediction = 0
loss = 0
N = len(age_predictions)
for i in range(N):
    if age_predictions[i] == y_train_sklearn[i]:
        TruePrediction += 1
    else:
        loss += (y_train_sklearn[i] - age_predictions[i])**2
print(f"Train Accuracy: {TruePrediction/len(age_predictions)}")
print(f"Train Loss: {(loss/N)**0.5}")
```

Train Accuracy: 0.99964
 Train Loss: 0.0322490309931942

```
In [117]: age_predictions = clf.predict(x_test_pca)
print(classification_report(y_test_sklearn, age_predictions))
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	93
1	0.39	0.90	0.54	777
2	0.27	0.03	0.06	452
3	0.32	0.15	0.20	551
4	0.75	0.01	0.02	250
accuracy			0.38	2123
macro avg	0.35	0.22	0.16	2123
weighted avg	0.37	0.38	0.26	2123

```
/Users/sujingyi/opt/anaconda3/envs/d2l/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
/Users/sujingyi/opt/anaconda3/envs/d2l/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
/Users/sujingyi/opt/anaconda3/envs/d2l/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
```

```
In [118]: TruePrediction = 0
loss = 0
N = len(age_predictions)
for i in range(N):
    if age_predictions[i] == y_test_sklearn[i]:
        TruePrediction += 1
    else:
        loss += (y_test_sklearn[i] - age_predictions[i])**2
print(f"Test Accuracy: {TruePrediction/len(age_predictions)}")
print(f"Test Loss: {(loss/N)**0.5}")
```

Test Accuracy: 0.3758831841733396
 Test Loss: 1.4466489612706839

```
In [119]: labels = class_label
# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    age_predict_index = age_predictions[index]
    age_true_index = np.argmax(y_test_age[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[age_predict_index],
                                  labels[age_true_index]),
                 color=("green" if age_predict_index == age_true_index else "red"))
plt.show()
```



Bootstrap Tuning Parameters

```
In [120]: start_time = time.time()
clf = RandomForestClassifier(max_depth=100, random_state=100, min_samples_leaf = 20, max_features = 600, warm_start=True)
clf.fit(x_train_pca, y_train_sklearn)
end_time = time.time()
train_mins, train_secs = train_time(start_time, end_time)
print(f'Train Time: {train_mins}m {train_secs}s')
```

Train Time: 18m 2s

```
In [122]: age_predictions = clf.predict(x_train_pca)
print(classification_report(y_train_sklearn, age_predictions))

TruePrediction = 0
loss = 0
N = len(age_predictions)
for i in range(N):
    if age_predictions[i] == y_train_sklearn[i]:
        TruePrediction += 1
    else:
        loss += (y_train_sklearn[i] - age_predictions[i])**2
print(f"Accuracy: {TruePrediction/len(age_predictions)}")
print(f"Loss: {((loss/N)**0.5}")
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1313
1	0.65	1.00	0.79	10014
2	1.00	0.52	0.69	4818
3	0.88	0.85	0.86	6228
4	0.98	0.37	0.53	2627
accuracy			0.75	25000
macro avg	0.70	0.55	0.57	25000
weighted avg	0.77	0.75	0.72	25000

Accuracy: 0.75084
Loss: 0.8949860334105779

```
/Users/sujingyi/opt/anaconda3/envs/d21/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
/Users/sujingyi/opt/anaconda3/envs/d21/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
/Users/sujingyi/opt/anaconda3/envs/d21/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
```

```
In [123]: age_predictions = clf.predict(x_test_pca)
print(classification_report(y_test_sklearn, age_predictions))

TruePrediction = 0
loss = 0
N = len(age_predictions)
for i in range(N):
    if age_predictions[i] == y_test_sklearn[i]:
        TruePrediction += 1
    else:
        loss += (y_test_sklearn[i] - age_predictions[i])**2
print(f"Accuracy: {TruePrediction/len(age_predictions)}")
print(f"Loss: {((loss/N)**0.5}")
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	93
1	0.41	0.94	0.57	777
2	0.00	0.00	0.00	452
3	0.38	0.23	0.29	551
4	0.75	0.07	0.13	250
accuracy			0.41	2123
macro avg	0.31	0.25	0.20	2123
weighted avg	0.34	0.41	0.30	2123

Accuracy: 0.4121526142251531
Loss: 1.3281615823700275

```
/Users/sujingyi/opt/anaconda3/envs/d21/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
/Users/sujingyi/opt/anaconda3/envs/d21/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
/Users/sujingyi/opt/anaconda3/envs/d21/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1245: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
```

Gender

```
In [124]: start_time = time.time()
clf = RandomForestClassifier(max_depth=100, min_samples_leaf = 10, random_state=100)
clf.fit(x_train_pca, y_train_gender)
end_time = time.time()
train_mins, train_secs = train_time(start_time, end_time)
print(f'Train Time: {train_mins}m {train_secs}s')
```

Train Time: 0m 58s

```
In [125]: gender_predictions = clf.predict(x_train_pca)
print(classification_report(y_train_gender, gender_predictions))
```

	precision	recall	f1-score	support
0	1.00	0.65	0.79	6946
1	0.88	1.00	0.94	18054
accuracy			0.90	25000
macro avg	0.94	0.82	0.86	25000
weighted avg	0.91	0.90	0.89	25000

```
In [127]: TruePrediction = 0
loss = 0
N = len(gender_predictions)
for i in range(N):
    if gender_predictions[i] == y_train_gender[i]:
        TruePrediction += 1
    else:
        loss += (y_train_gender[i] - gender_predictions[i])**2
print(f"Train Accuracy: {TruePrediction/len(gender_predictions)}")
print(f"Train Loss: {(loss/N)**0.5}")
```

Train Accuracy: 0.90208
Train Loss: 0.31292171544972713

```
In [128]: gender_predictions = clf.predict(x_test_pca)
print(classification_report(y_test_gender, gender_predictions))
```

	precision	recall	f1-score	support
0	1.00	0.02	0.04	557
1	0.74	1.00	0.85	1566
accuracy			0.74	2123
macro avg	0.87	0.51	0.45	2123
weighted avg	0.81	0.74	0.64	2123

```
In [130]: TruePrediction = 0
loss = 0
N = len(gender_predictions)
for i in range(N):
    if gender_predictions[i] == y_test_gender[i]:
        TruePrediction += 1
    else:
        loss += (y_test_gender[i] - gender_predictions[i])**2
print(f"Test Accuracy: {TruePrediction/len(gender_predictions)}")
print(f"Test Loss: {(loss/N)**0.5}")
```

Test Accuracy: 0.743287800282619
Test Loss: 0.506667740948031

```
In [131]: # Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    gender_predict_index = gender_predictions[index]
    gender_true_index = y_test_gender[index]
    # Set the title for each image
    ax.set_title("{} ({})".format(gender_predict_index,
                                  gender_true_index),
                color="green" if gender_predict_index == gender_true_index else "red"))
plt.show()
```



Bootstrap Tuning Parameters

```
In [132]: start_time = time.time()
clf = RandomForestClassifier(max_depth=100, random_state=100, min_samples_leaf = 100, max_features = 600, warm_
clf.fit(x_train_pca, y_train_gender)
end_time = time.time()
train_mins, train_secs = train_time(start_time, end_time)
print(f'Train Time: {train_mins}m {train_secs}s')
```

Train Time: 10m 25s

```
In [133]: gender_predictions = clf.predict(x_train_pca)
print(classification_report(y_train_gender, gender_predictions))
```

	precision	recall	f1-score	support
0	0.85	0.25	0.39	6946
1	0.77	0.98	0.87	18054
accuracy			0.78	25000
macro avg	0.81	0.62	0.63	25000
weighted avg	0.79	0.78	0.73	25000

```
In [134]: TruePrediction = 0
loss = 0
N = len(gender_predictions)
for i in range(N):
    if gender_predictions[i] == y_train_gender[i]:
        TruePrediction += 1
    else:
        loss += (y_train_gender[i] - gender_predictions[i])**2
print(f"Train Accuracy: {TruePrediction/len(gender_predictions)}")
print(f"Train Loss: {(loss/N)**0.5}")

Train Accuracy: 0.78024
Train Loss: 0.4687856653098514
```

```
In [135]: gender_predictions = clf.predict(x_test_pca)
print(classification_report(y_test_gender, gender_predictions))
```

	precision	recall	f1-score	support
0	0.74	0.21	0.33	557
1	0.78	0.97	0.86	1566
accuracy			0.77	2123
macro avg	0.76	0.59	0.60	2123
weighted avg	0.77	0.77	0.72	2123

```
In [136]: TruePrediction = 0
loss = 0
N = len(gender_predictions)
for i in range(N):
    if gender_predictions[i] == y_test_gender[i]:
        TruePrediction += 1
    else:
        loss += (y_test_gender[i] - gender_predictions[i])**2
print(f"Test Accuracy: {TruePrediction/len(gender_predictions)}")
print(f"Test Loss: {(loss/N)**0.5}")

Test Accuracy: 0.7743758831841734
Test Loss: 0.4749990703315394
```

```
In [137]: # Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    age_predict_index = age_predictions[index]
    age_true_index = np.argmax(y_test_age[index])
    gender_predict_index = gender_predictions[index]
    gender_true_index = y_test_gender[index]
    # Set the title for each image
    ax.set_title("Age: {} ({}) | Gender:{} ({})".format(labels[age_predict_index],
                                                       labels[age_true_index],
                                                       gender_predict_index,
                                                       gender_true_index),
               color=("green"
                      if age_predict_index == age_true_index and gender_predict_index == gender_true_index
                      else "red"))
plt.show()
```



```
In [ ]:
```


CNN- Predict Age

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import math
import cv2
import imageio
import matplotlib.pyplot as plt
import os
import seaborn as sns
!pip install umap
import umap
from PIL import Image
from scipy import misc
from os import listdir
from os.path import isfile, join
import numpy as np
from scipy import misc
from random import shuffle
from collections import Counter
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.utils import to_categorical

from tensorflow.keras.optimizers import RMSprop, Adam
from tensorflow.keras import layers
from tensorflow.keras import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import plot_model
from tensorflow.keras import backend as K
from keras.callbacks import LearningRateScheduler, EarlyStopping, Callback, ReduceLROnP
```

Requirement already satisfied: umap in c:\miniconda3\envs\d2l\lib\site-packages (0.1.1)

Set the path and import the photos

In [2]:

```
data_csv = pd.read_csv('wikifinal.csv')
```

In [3]:

```
len(data_csv)
```

Out[3]: 27123

In [4]:

```
data_csv = data_csv[data_csv['AGE'] <= 100].reset_index(drop=True)
data_csv.head()
```

Out[4]:

AGE	GENDER	NAME	PATH
-----	--------	------	------

	AGE	GENDER	NAME	PATH
0	27	1	Sami Jauhojärvi	1_crop.jpg
1	59	1	Marc Okrand	2_crop.jpg
2	50	0	Krista Tippett	3_crop.jpg
3	32	1	Bernie Whitebear	4_crop.jpg
4	41	1	Carl Greenberg	5_crop.jpg

In [5]: `len(data_csv)`

Out[5]: 27105

In [6]: `mypath = os.chdir('../crop')`
`onlyfiles = os.listdir()`

In [7]: `image_name = []`
`for name in data_csv["PATH"]:`
 `image_name.append(name)`

In [8]: `# This way is faster than the original one`
`def convertImage(filename):`
 `face = imageio.imread(filename)`
 `face = cv2.resize(face, (128, 128))`
 `return face`

`Image_data = list(map(convertImage, image_name))`

In [9]: `len(Image_data)`

Out[9]: 27105

#alternative to `Image_data` `image_data =[]` for `i in image_name:` `face = imageio.imread(os.path.join("../crop/" + i))`
`face = cv2.resize(face, (128, 128))` `image_data.append(face)`

In [10]: `len(Image_data)`

Out[10]: 27105

In [11]: `Image_data = np.squeeze(Image_data)`
`Image_data.shape`

Out[11]: (27105, 128, 128, 3)

In [12]: `# normalize data`
`Image_data = Image_data.astype('float32')`
`Image_data /= 255`

Get the age and gender data

In [13]:

```
age, gender = list(), list()
for i in range(len(data_csv)):
    age.append(data_csv["AGE"][i])
    gender.append(data_csv["GENDER"][i])
```

Due to the results come from PCA, we choose to use the age label:

- 0-18 years old
- 18-30 years old
- 30-40 years old
- 40-60 years old
- 60-100 years old

In [14]:

```
age_label = ['(0,18)', '(18,30)', '(30,40)', '(40,60)', '(60,100)']

age_classes = []

count = 0
for i in age:
    count += 1
    if i > 0 and i <= 18:
        age_classes.append(0)
    elif i <= 30:
        age_classes.append(1)
    elif i <= 40:
        age_classes.append(2)
    elif i <= 60 :
        age_classes.append(3)
    elif i <= 100:
        age_classes.append(4)
print(count)
```

27105

In [15]:

```
age_classes[:10]
```

Out[15]: [1, 3, 3, 2, 3, 2, 1, 3, 2, 1]

In [16]:

```
len(age_classes)
```

Out[16]: 27105

In [17]:

```
age_labels = to_categorical(age_classes, num_classes=5)
age_labels[:10]
```

Out[17]: array([[0., 1., 0., 0., 0.],
 [0., 0., 0., 1., 0.],
 [0., 0., 0., 1., 0.],

```
[0., 0., 1., 0., 0.],
[0., 0., 0., 1., 0.],
[0., 0., 1., 0., 0.],
[0., 1., 0., 0., 0.],
[0., 0., 0., 1., 0.],
[0., 0., 1., 0., 0.],
[0., 1., 0., 0., 0.]], dtype=float32)
```

```
In [18]: len(age_labels)
```

```
Out[18]: 27105
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train_age, y_test_age =
cross_validation.train_test_split(Image_data, age_labels, test_size=0.1, random_state=7011)
```

```
In [19]: print("Using 0- 80% data as training data: " + str(int(len(Image_data)*0.8)))
print("Using 80%-90% data as training data: " + str(int(len(Image_data)*0.9)))
print("Number of data in validation set: " + str(24394-21684))
```

```
Using 0- 80% data as training data: 21684
Using 80%-90% data as training data: 24394
Number of data in validation set: 2710
```

```
In [20]: (x_train, y_train_age), (x_test, y_test_age) = (Image_data[:21684], age_labels[:21684])
(x_valid, y_valid_age) = (x_test[:2710], y_test_age[:2710])
(x_test, y_test_age) = (x_test[2710:], y_test_age[2710:])
```

```
In [21]: len(x_train) + len(x_test) + len(x_valid) == len(Image_data)
```

```
Out[21]: True
```

```
In [22]: #earlystop
earlystop = EarlyStopping(monitor='accuracy',
                          min_delta=0,
                          patience=10,
                          verbose=2,
                          mode='max',
                          baseline=None,
                          restore_best_weights=True)

# Plateau
plateau = ReduceLROnPlateau(monitor='accuracy',
                            factor=0.1,
                            patience=5,
                            verbose=2,
                            mode='max',
                            min_delta=0.0001,
                            cooldown=0,
                            min_lr=0)
```

Build the multilayer CNN (change kernel size)

1-1 (2-1). Two-layer CNN

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- kernel size = 2
- Dropout = 0.3

In [23]:

```
model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(5, activation='sigmoid'))

# Take a Look at the model summary
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d (Conv2D)	(None, 128, 128, 64)	832
<hr/>		
max_pooling2d (MaxPooling2D)	(None, 64, 64, 64)	0
<hr/>		
dropout (Dropout)	(None, 64, 64, 64)	0
<hr/>		
conv2d_1 (Conv2D)	(None, 64, 64, 32)	8224
<hr/>		
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 32)	0
<hr/>		
dropout_1 (Dropout)	(None, 32, 32, 32)	0
<hr/>		
flatten (Flatten)	(None, 32768)	0
<hr/>		
dense (Dense)	(None, 256)	8388864
<hr/>		
dropout_2 (Dropout)	(None, 256)	0
<hr/>		
dense_1 (Dense)	(None, 5)	1285
<hr/>		
Total params: 8,399,205		
Trainable params: 8,399,205		
Non-trainable params: 0		

In [25]:

```
model.fit(x_train,
          y_train_age,
          batch_size=64,
          validation_data=(x_valid, y_valid_age),
          callbacks=[earlystop, plateau],)
```

Epoch 1/10

```
339/339 [=====] - 375s 1s/step - loss: 0.5573 - accuracy: 0.376
2 - val_loss: 0.5012 - val_accuracy: 0.4487
Epoch 2/10
339/339 [=====] - 366s 1s/step - loss: 0.4355 - accuracy: 0.440
6 - val_loss: 0.4304 - val_accuracy: 0.4926
Epoch 3/10
339/339 [=====] - 363s 1s/step - loss: 0.4232 - accuracy: 0.461
1 - val_loss: 0.4167 - val_accuracy: 0.5018
Epoch 4/10
339/339 [=====] - 369s 1s/step - loss: 0.4135 - accuracy: 0.482
1 - val_loss: 0.3990 - val_accuracy: 0.5192
Epoch 5/10
339/339 [=====] - 366s 1s/step - loss: 0.4075 - accuracy: 0.488
3 - val_loss: 0.4011 - val_accuracy: 0.5052
Epoch 6/10
339/339 [=====] - 365s 1s/step - loss: 0.4033 - accuracy: 0.497
4 - val_loss: 0.4019 - val_accuracy: 0.5018
Epoch 7/10
339/339 [=====] - 361s 1s/step - loss: 0.3969 - accuracy: 0.504
8 - val_loss: 0.3969 - val_accuracy: 0.5192
Epoch 8/10
339/339 [=====] - 363s 1s/step - loss: 0.3942 - accuracy: 0.516
1 - val_loss: 0.3868 - val_accuracy: 0.5255
Epoch 9/10
339/339 [=====] - 365s 1s/step - loss: 0.3852 - accuracy: 0.527
5 - val_loss: 0.3872 - val_accuracy: 0.5266
Epoch 10/10
339/339 [=====] - 223s 657ms/step - loss: 0.3777 - accuracy: 0.
5374 - val_loss: 0.3847 - val_accuracy: 0.5236
Out[25]: <tensorflow.python.keras.callbacks.History at 0x26f6e0b2880>
```

```
In [26]: localtime = time.asctime(time.localtime(time.time()))
print(localtime)
```

```
Sun Apr 25 03:26:19 2021
```

```
# Evaluate the model on test set
score = model.evaluate(x_test, y_test_age, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.3876202404499054, 0.5271117687225342]
```

Try to use the early stop function

```
In [24]: import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

```
Tue Apr 27 22:56:03 2021
```

```
In [25]: model.compile(loss='binary_crossentropy',
                    optimizer='adam',
                    metrics=['accuracy'])
```

```
In [26]: model.fit(x_train,y_train_age,
```

```

    validation_data=(x_valid , y_valid_age),
    batch_size=64,
    steps_per_epoch=20,
    epochs=100,
    validation_steps=2,
    verbose=2,
    workers=1,
    use_multiprocessing = False,
    callbacks=[earlystop, plateau])

```

```

Epoch 1/100
20/20 - 15s - loss: 0.7743 - accuracy: 0.3297 - val_loss: 0.5580 - val_accuracy: 0.4609
Epoch 2/100
20/20 - 14s - loss: 0.4889 - accuracy: 0.3609 - val_loss: 0.5206 - val_accuracy: 0.4609
Epoch 3/100
20/20 - 13s - loss: 0.4813 - accuracy: 0.3398 - val_loss: 0.4909 - val_accuracy: 0.4609
Epoch 4/100
20/20 - 14s - loss: 0.4733 - accuracy: 0.3719 - val_loss: 0.5038 - val_accuracy: 0.4609
Epoch 5/100
20/20 - 13s - loss: 0.4708 - accuracy: 0.3656 - val_loss: 0.5037 - val_accuracy: 0.4609
Epoch 6/100
20/20 - 13s - loss: 0.4701 - accuracy: 0.3852 - val_loss: 0.5189 - val_accuracy: 0.4688
Epoch 7/100
20/20 - 14s - loss: 0.4591 - accuracy: 0.4039 - val_loss: 0.4975 - val_accuracy: 0.5000
Epoch 8/100
20/20 - 14s - loss: 0.4562 - accuracy: 0.4086 - val_loss: 0.4680 - val_accuracy: 0.5078
Epoch 9/100
20/20 - 13s - loss: 0.4495 - accuracy: 0.4156 - val_loss: 0.4824 - val_accuracy: 0.4844
Epoch 10/100
20/20 - 14s - loss: 0.4579 - accuracy: 0.3922 - val_loss: 0.4670 - val_accuracy: 0.4844
Epoch 11/100
20/20 - 14s - loss: 0.4562 - accuracy: 0.3711 - val_loss: 0.4660 - val_accuracy: 0.5000
Epoch 12/100
20/20 - 14s - loss: 0.4555 - accuracy: 0.4172 - val_loss: 0.4884 - val_accuracy: 0.5078
Epoch 13/100
20/20 - 14s - loss: 0.4418 - accuracy: 0.4141 - val_loss: 0.4643 - val_accuracy: 0.4766
Epoch 14/100
20/20 - 14s - loss: 0.4484 - accuracy: 0.3992 - val_loss: 0.4745 - val_accuracy: 0.5156
Epoch 15/100
20/20 - 15s - loss: 0.4456 - accuracy: 0.4156 - val_loss: 0.4508 - val_accuracy: 0.4922
Epoch 16/100
20/20 - 14s - loss: 0.4322 - accuracy: 0.4297 - val_loss: 0.4390 - val_accuracy: 0.5312
Epoch 17/100
20/20 - 14s - loss: 0.4434 - accuracy: 0.4251 - val_loss: 0.4634 - val_accuracy: 0.5156
Epoch 18/100
20/20 - 14s - loss: 0.4441 - accuracy: 0.4031 - val_loss: 0.4734 - val_accuracy: 0.5000
Epoch 19/100
20/20 - 14s - loss: 0.4366 - accuracy: 0.4367 - val_loss: 0.4518 - val_accuracy: 0.5000
Epoch 20/100
20/20 - 14s - loss: 0.4312 - accuracy: 0.4469 - val_loss: 0.4377 - val_accuracy: 0.5312
Epoch 21/100
20/20 - 14s - loss: 0.4397 - accuracy: 0.4266 - val_loss: 0.4571 - val_accuracy: 0.5156
Epoch 22/100
20/20 - 14s - loss: 0.4352 - accuracy: 0.4352 - val_loss: 0.4297 - val_accuracy: 0.5234
Epoch 23/100
20/20 - 14s - loss: 0.4328 - accuracy: 0.4453 - val_loss: 0.4419 - val_accuracy: 0.5078
Epoch 24/100
20/20 - 22s - loss: 0.4438 - accuracy: 0.4383 - val_loss: 0.4461 - val_accuracy: 0.5156
Epoch 25/100
20/20 - 23s - loss: 0.4347 - accuracy: 0.4391 - val_loss: 0.4334 - val_accuracy: 0.5000

Epoch 00025: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.
Epoch 26/100
20/20 - 23s - loss: 0.4300 - accuracy: 0.4633 - val_loss: 0.4299 - val_accuracy: 0.5156

```

```

Epoch 27/100
20/20 - 21s - loss: 0.4263 - accuracy: 0.4586 - val_loss: 0.4275 - val_accuracy: 0.5000
Epoch 28/100
20/20 - 20s - loss: 0.4225 - accuracy: 0.4734 - val_loss: 0.4287 - val_accuracy: 0.5156
Epoch 29/100
20/20 - 20s - loss: 0.4273 - accuracy: 0.4375 - val_loss: 0.4255 - val_accuracy: 0.5078
Epoch 30/100
20/20 - 22s - loss: 0.4233 - accuracy: 0.4500 - val_loss: 0.4245 - val_accuracy: 0.5391
Epoch 31/100
20/20 - 21s - loss: 0.4286 - accuracy: 0.4500 - val_loss: 0.4239 - val_accuracy: 0.5547
Epoch 32/100
20/20 - 23s - loss: 0.4197 - accuracy: 0.4672 - val_loss: 0.4198 - val_accuracy: 0.5312
Epoch 33/100
20/20 - 22s - loss: 0.4141 - accuracy: 0.4734 - val_loss: 0.4157 - val_accuracy: 0.5234

Epoch 00033: ReduceLROnPlateau reducing learning rate to 1.000000474974514e-05.
Epoch 34/100
20/20 - 23s - loss: 0.4253 - accuracy: 0.4677 - val_loss: 0.4174 - val_accuracy: 0.5156
Epoch 35/100
20/20 - 22s - loss: 0.4238 - accuracy: 0.4734 - val_loss: 0.4198 - val_accuracy: 0.5234
Epoch 36/100
20/20 - 21s - loss: 0.4214 - accuracy: 0.4617 - val_loss: 0.4205 - val_accuracy: 0.5312
Epoch 37/100
20/20 - 21s - loss: 0.4209 - accuracy: 0.4672 - val_loss: 0.4208 - val_accuracy: 0.5312
Epoch 38/100
20/20 - 22s - loss: 0.4320 - accuracy: 0.4633 - val_loss: 0.4215 - val_accuracy: 0.5312
Restoring model weights from the end of the best epoch.

Epoch 00038: ReduceLROnPlateau reducing learning rate to 1.000000656873453e-06.
Epoch 00038: early stopping

```

Out[26]: <tensorflow.python.keras.callbacks.History at 0x2456562dc10>

In [31]:

```

# Evaluate the model on test set
score = model.evaluate(x_test, y_test_age, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)

```

Test accuracy: [0.44952255487442017, 0.4680929481983185]

In [32]:

```

labels = ["0-18 years old", # index 0
          "18-30 years old", # index 1
          "30-40 years old", # index 2
          "40-60 years old", # index 3
          "60-100 years old", # index 4
         ]

```

In [34]:

```

y_hat = model.predict(x_test)

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test_age[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],

```

```

        labels[true_index]),
        color=( "green" if predict_index == true_index else "r"
plt.show()

```



1-2. Two-layer CNN

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- Change kernel size: kernel size = 3
- Dropout = 0.3

In [69]:

```

model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(5, activation='sigmoid'))

# Take a Look at the model summary
model.summary()

```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 128, 128, 64)	1792
max_pooling2d_6 (MaxPooling2D)	(None, 64, 64, 64)	0
dropout_10 (Dropout)	(None, 64, 64, 64)	0
conv2d_7 (Conv2D)	(None, 64, 64, 32)	18464

max_pooling2d_7 (MaxPooling2	(None, 32, 32, 32)	0
dropout_11 (Dropout)	(None, 32, 32, 32)	0
flatten_3 (Flatten)	(None, 32768)	0
dense_7 (Dense)	(None, 256)	8388864
dropout_12 (Dropout)	(None, 256)	0
dense_8 (Dense)	(None, 5)	1285
<hr/>		
Total params:	8,410,405	
Trainable params:	8,410,405	
Non-trainable params:	0	

In [23]:

```
import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

Sun Apr 25 12:53:59 2021

In [24]:

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=[ 'accuracy'])
```

In [25]:

```
model.fit(x_train,
          y_train_age,
          batch_size=64,
          epochs=10,
          validation_data=(x_valid, y_valid_age),)
```

```
Epoch 1/10
339/339 [=====] - 275s 809ms/step - loss: 0.5176 - accuracy: 0.
3793 - val_loss: 0.4498 - val_accuracy: 0.4554
Epoch 2/10
339/339 [=====] - 254s 749ms/step - loss: 0.4354 - accuracy: 0.
4279 - val_loss: 0.4133 - val_accuracy: 0.4878
Epoch 3/10
339/339 [=====] - 253s 745ms/step - loss: 0.4178 - accuracy: 0.
4682 - val_loss: 0.4044 - val_accuracy: 0.5196
Epoch 4/10
339/339 [=====] - 252s 743ms/step - loss: 0.4054 - accuracy: 0.
4955 - val_loss: 0.4015 - val_accuracy: 0.5111
Epoch 5/10
339/339 [=====] - 252s 744ms/step - loss: 0.3962 - accuracy: 0.
5034 - val_loss: 0.3929 - val_accuracy: 0.5284
Epoch 6/10
339/339 [=====] - 252s 743ms/step - loss: 0.3884 - accuracy: 0.
5189 - val_loss: 0.3907 - val_accuracy: 0.5321
Epoch 7/10
339/339 [=====] - 251s 741ms/step - loss: 0.3791 - accuracy: 0.
5353 - val_loss: 0.3963 - val_accuracy: 0.5339
Epoch 8/10
339/339 [=====] - 251s 739ms/step - loss: 0.3727 - accuracy: 0.
5389 - val_loss: 0.3907 - val_accuracy: 0.5288
Epoch 9/10
339/339 [=====] - 251s 739ms/step - loss: 0.3625 - accuracy: 0.
5564 - val_loss: 0.3872 - val_accuracy: 0.5373
```

```
Epoch 10/10
339/339 [=====] - 250s 738ms/step - loss: 0.3535 - accuracy: 0.
5699 - val_loss: 0.3873 - val_accuracy: 0.5354
Out[25]: <tensorflow.python.keras.callbacks.History at 0x23aa4dd37f0>
```

```
In [27]: # Evaluate the model on test set
score = model.evaluate(x_test, y_test_age, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.3877262771129608, 0.5204721689224243]
```

Use Early stop

```
In [70]: import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

```
Tue Apr 27 16:50:01 2021
```

```
In [71]: model.compile(loss='binary_crossentropy',
                    optimizer='adam',
                    metrics=['accuracy'])
```

```
In [72]: model.fit(x_train,y_train_age,
                validation_data=(x_valid , y_valid_age),
                batch_size=64,
                steps_per_epoch=20,
                epochs=100,
                validation_steps=2,
                verbose=2,
                workers=1,
                use_multiprocessing = False,
                callbacks=[earlystop, plateau])
```

```
Epoch 1/100
20/20 - 17s - loss: 0.7022 - accuracy: 0.3102 - val_loss: 0.5283 - val_accuracy: 0.4609
Epoch 2/100
20/20 - 16s - loss: 0.4807 - accuracy: 0.3734 - val_loss: 0.5176 - val_accuracy: 0.4609
Epoch 3/100
20/20 - 17s - loss: 0.4728 - accuracy: 0.3477 - val_loss: 0.5622 - val_accuracy: 0.4609
Epoch 4/100
20/20 - 17s - loss: 0.4638 - accuracy: 0.3914 - val_loss: 0.4894 - val_accuracy: 0.4609
Epoch 5/100
20/20 - 16s - loss: 0.4728 - accuracy: 0.3648 - val_loss: 0.4919 - val_accuracy: 0.4609
Epoch 6/100
20/20 - 15s - loss: 0.4630 - accuracy: 0.3875 - val_loss: 0.5164 - val_accuracy: 0.4688
Epoch 7/100
20/20 - 16s - loss: 0.4546 - accuracy: 0.4102 - val_loss: 0.5238 - val_accuracy: 0.4766
Epoch 8/100
20/20 - 16s - loss: 0.4493 - accuracy: 0.4047 - val_loss: 0.4746 - val_accuracy: 0.4766
Epoch 9/100
20/20 - 17s - loss: 0.4434 - accuracy: 0.4180 - val_loss: 0.4718 - val_accuracy: 0.4922
Epoch 10/100
20/20 - 17s - loss: 0.4487 - accuracy: 0.4117 - val_loss: 0.4726 - val_accuracy: 0.4922
Epoch 11/100
20/20 - 16s - loss: 0.4463 - accuracy: 0.4180 - val_loss: 0.4601 - val_accuracy: 0.5078
```

Epoch 12/100
20/20 - 16s - loss: 0.4409 - accuracy: 0.4281 - val_loss: 0.4720 - val_accuracy: 0.5234
Epoch 13/100
20/20 - 16s - loss: 0.4452 - accuracy: 0.4016 - val_loss: 0.4504 - val_accuracy: 0.4844
Epoch 14/100
20/20 - 16s - loss: 0.4373 - accuracy: 0.4320 - val_loss: 0.4264 - val_accuracy: 0.5078
Epoch 15/100
20/20 - 17s - loss: 0.4447 - accuracy: 0.4141 - val_loss: 0.4609 - val_accuracy: 0.4922
Epoch 16/100
20/20 - 16s - loss: 0.4472 - accuracy: 0.4133 - val_loss: 0.4798 - val_accuracy: 0.5000
Epoch 17/100
20/20 - 16s - loss: 0.4431 - accuracy: 0.3967 - val_loss: 0.4361 - val_accuracy: 0.5312
Epoch 18/100
20/20 - 16s - loss: 0.4349 - accuracy: 0.4219 - val_loss: 0.4358 - val_accuracy: 0.5078
Epoch 19/100
20/20 - 16s - loss: 0.4345 - accuracy: 0.4516 - val_loss: 0.4546 - val_accuracy: 0.5000
Epoch 20/100
20/20 - 16s - loss: 0.4397 - accuracy: 0.4383 - val_loss: 0.4329 - val_accuracy: 0.5156
Epoch 21/100
20/20 - 16s - loss: 0.4302 - accuracy: 0.4500 - val_loss: 0.4414 - val_accuracy: 0.5234
Epoch 22/100
20/20 - 17s - loss: 0.4378 - accuracy: 0.4477 - val_loss: 0.4253 - val_accuracy: 0.5391
Epoch 23/100
20/20 - 16s - loss: 0.4346 - accuracy: 0.4406 - val_loss: 0.4178 - val_accuracy: 0.5234
Epoch 24/100
20/20 - 17s - loss: 0.4329 - accuracy: 0.4531 - val_loss: 0.4105 - val_accuracy: 0.5234
Epoch 25/100
20/20 - 17s - loss: 0.4397 - accuracy: 0.4297 - val_loss: 0.4349 - val_accuracy: 0.5312
Epoch 26/100
20/20 - 17s - loss: 0.4262 - accuracy: 0.4547 - val_loss: 0.4096 - val_accuracy: 0.5391
Epoch 27/100
20/20 - 17s - loss: 0.4342 - accuracy: 0.4383 - val_loss: 0.4179 - val_accuracy: 0.5469
Epoch 28/100
20/20 - 17s - loss: 0.4321 - accuracy: 0.4461 - val_loss: 0.4143 - val_accuracy: 0.5391
Epoch 29/100
20/20 - 17s - loss: 0.4329 - accuracy: 0.4437 - val_loss: 0.4284 - val_accuracy: 0.5703
Epoch 30/100
20/20 - 31s - loss: 0.4275 - accuracy: 0.4414 - val_loss: 0.4231 - val_accuracy: 0.5469
Epoch 31/100
20/20 - 30s - loss: 0.4279 - accuracy: 0.4695 - val_loss: 0.4171 - val_accuracy: 0.5312
Epoch 32/100
20/20 - 37s - loss: 0.4292 - accuracy: 0.4555 - val_loss: 0.3934 - val_accuracy: 0.5078
Epoch 33/100
20/20 - 59s - loss: 0.4243 - accuracy: 0.4508 - val_loss: 0.3893 - val_accuracy: 0.5703
Epoch 34/100
20/20 - 80s - loss: 0.4230 - accuracy: 0.4732 - val_loss: 0.4099 - val_accuracy: 0.5547
Epoch 35/100
20/20 - 86s - loss: 0.4338 - accuracy: 0.4281 - val_loss: 0.4032 - val_accuracy: 0.5703
Epoch 36/100
20/20 - 103s - loss: 0.4215 - accuracy: 0.4641 - val_loss: 0.3951 - val_accuracy: 0.5156
Epoch 37/100
20/20 - 109s - loss: 0.4205 - accuracy: 0.4898 - val_loss: 0.4103 - val_accuracy: 0.5703
Epoch 38/100
20/20 - 117s - loss: 0.4242 - accuracy: 0.4656 - val_loss: 0.4121 - val_accuracy: 0.5625
Epoch 39/100
20/20 - 184s - loss: 0.4311 - accuracy: 0.4492 - val_loss: 0.4096 - val_accuracy: 0.5469
Epoch 40/100
20/20 - 259s - loss: 0.4218 - accuracy: 0.4797 - val_loss: 0.4076 - val_accuracy: 0.5625
Epoch 41/100
20/20 - 239s - loss: 0.4154 - accuracy: 0.4727 - val_loss: 0.3846 - val_accuracy: 0.5703
Epoch 42/100
20/20 - 194s - loss: 0.4209 - accuracy: 0.4773 - val_loss: 0.4078 - val_accuracy: 0.5625

Epoch 00042: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.
Epoch 43/100

```
20/20 - 192s - loss: 0.4189 - accuracy: 0.4789 - val_loss: 0.3947 - val_accuracy: 0.5703
Epoch 44/100
20/20 - 197s - loss: 0.4125 - accuracy: 0.4945 - val_loss: 0.3957 - val_accuracy: 0.5625
Epoch 45/100
20/20 - 333s - loss: 0.4125 - accuracy: 0.4727 - val_loss: 0.3920 - val_accuracy: 0.5547
Epoch 46/100
20/20 - 342s - loss: 0.4193 - accuracy: 0.4594 - val_loss: 0.3932 - val_accuracy: 0.5625
Epoch 47/100
20/20 - 278s - loss: 0.4202 - accuracy: 0.4609 - val_loss: 0.3913 - val_accuracy: 0.5781
Epoch 48/100
20/20 - 131s - loss: 0.4248 - accuracy: 0.4414 - val_loss: 0.3945 - val_accuracy: 0.5703
Epoch 49/100
20/20 - 209s - loss: 0.4199 - accuracy: 0.4688 - val_loss: 0.3883 - val_accuracy: 0.5781

Epoch 00049: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 50/100
20/20 - 348s - loss: 0.4254 - accuracy: 0.4609 - val_loss: 0.3887 - val_accuracy: 0.5781
Epoch 51/100
20/20 - 422s - loss: 0.4154 - accuracy: 0.4677 - val_loss: 0.3888 - val_accuracy: 0.5781
Epoch 52/100
20/20 - 376s - loss: 0.4141 - accuracy: 0.4742 - val_loss: 0.3883 - val_accuracy: 0.5781
Epoch 53/100
20/20 - 401s - loss: 0.4171 - accuracy: 0.4672 - val_loss: 0.3884 - val_accuracy: 0.5781
Epoch 54/100
20/20 - 383s - loss: 0.4152 - accuracy: 0.4766 - val_loss: 0.3882 - val_accuracy: 0.5781
Restoring model weights from the end of the best epoch.

Epoch 00054: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
Epoch 00054: early stopping
Out[72]: <tensorflow.python.keras.callbacks.History at 0x188b5685b80>
```

In [73]:

```
# Evaluate the model on test set
score = model.evaluate(x_test, y_test_age, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.42105337977409363, 0.4928070902824402]
```

Second try

In [35]:

```
model = tf.keras.Sequential()

# Must define the input shape in the first Layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(5, activation='sigmoid'))

# Take a Look at the model summary
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_2 (Conv2D)	(None, 128, 128, 64)	1792
max_pooling2d_2 (MaxPooling2D)	(None, 64, 64, 64)	0
dropout_3 (Dropout)	(None, 64, 64, 64)	0
conv2d_3 (Conv2D)	(None, 64, 64, 32)	18464
max_pooling2d_3 (MaxPooling2D)	(None, 32, 32, 32)	0
dropout_4 (Dropout)	(None, 32, 32, 32)	0
flatten_1 (Flatten)	(None, 32768)	0
dense_2 (Dense)	(None, 256)	8388864
dropout_5 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 5)	1285
<hr/>		
Total params: 8,410,405		
Trainable params: 8,410,405		
Non-trainable params: 0		

In [36]:

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [37]:

```
model.fit(x_train,y_train_age,
           validation_data=(x_valid , y_valid_age),
           batch_size=64,
           steps_per_epoch=20,
           epochs=100,
           validation_steps=2,
           verbose=2,
           workers=1,
           use_multiprocessing = False,
           callbacks=[earlystop, plateau])
```

```
Epoch 1/100
20/20 - 29s - loss: 0.6539 - accuracy: 0.3555 - val_loss: 0.6331 - val_accuracy: 0.4609
Epoch 2/100
20/20 - 26s - loss: 0.4771 - accuracy: 0.3609 - val_loss: 0.4772 - val_accuracy: 0.4609
Epoch 3/100
20/20 - 26s - loss: 0.4666 - accuracy: 0.3750 - val_loss: 0.4865 - val_accuracy: 0.4609
Epoch 4/100
20/20 - 27s - loss: 0.4657 - accuracy: 0.3883 - val_loss: 0.5128 - val_accuracy: 0.4609
Epoch 5/100
20/20 - 27s - loss: 0.4665 - accuracy: 0.3711 - val_loss: 0.4988 - val_accuracy: 0.4609
Epoch 6/100
20/20 - 27s - loss: 0.4637 - accuracy: 0.3945 - val_loss: 0.5287 - val_accuracy: 0.4609
Epoch 7/100
20/20 - 25s - loss: 0.4656 - accuracy: 0.3938 - val_loss: 0.4904 - val_accuracy: 0.4609
Epoch 8/100
20/20 - 26s - loss: 0.4584 - accuracy: 0.4156 - val_loss: 0.5207 - val_accuracy: 0.4609
Epoch 9/100
20/20 - 25s - loss: 0.4659 - accuracy: 0.3727 - val_loss: 0.5220 - val_accuracy: 0.4609
```

Epoch 10/100
20/20 - 25s - loss: 0.4607 - accuracy: 0.3984 - val_loss: 0.5207 - val_accuracy: 0.4609
Epoch 11/100
20/20 - 24s - loss: 0.4622 - accuracy: 0.3938 - val_loss: 0.4768 - val_accuracy: 0.4609
Epoch 12/100
20/20 - 24s - loss: 0.4541 - accuracy: 0.4187 - val_loss: 0.4733 - val_accuracy: 0.4609
Epoch 13/100
20/20 - 25s - loss: 0.4527 - accuracy: 0.4008 - val_loss: 0.4713 - val_accuracy: 0.4922
Epoch 14/100
20/20 - 24s - loss: 0.4569 - accuracy: 0.4047 - val_loss: 0.4695 - val_accuracy: 0.4688
Epoch 15/100
20/20 - 25s - loss: 0.4532 - accuracy: 0.4016 - val_loss: 0.4576 - val_accuracy: 0.4609
Epoch 16/100
20/20 - 24s - loss: 0.4475 - accuracy: 0.4008 - val_loss: 0.4600 - val_accuracy: 0.5312
Epoch 17/100
20/20 - 26s - loss: 0.4478 - accuracy: 0.4196 - val_loss: 0.4665 - val_accuracy: 0.5234
Epoch 18/100
20/20 - 28s - loss: 0.4403 - accuracy: 0.4055 - val_loss: 0.4451 - val_accuracy: 0.5078
Epoch 19/100
20/20 - 26s - loss: 0.4388 - accuracy: 0.4148 - val_loss: 0.4292 - val_accuracy: 0.5469
Epoch 20/100
20/20 - 26s - loss: 0.4402 - accuracy: 0.4203 - val_loss: 0.4502 - val_accuracy: 0.4688
Epoch 21/100
20/20 - 25s - loss: 0.4352 - accuracy: 0.4437 - val_loss: 0.4346 - val_accuracy: 0.5312
Epoch 22/100
20/20 - 25s - loss: 0.4324 - accuracy: 0.4609 - val_loss: 0.4281 - val_accuracy: 0.5156
Epoch 23/100
20/20 - 25s - loss: 0.4349 - accuracy: 0.4461 - val_loss: 0.4442 - val_accuracy: 0.5312
Epoch 24/100
20/20 - 25s - loss: 0.4342 - accuracy: 0.4328 - val_loss: 0.4270 - val_accuracy: 0.4922
Epoch 25/100
20/20 - 25s - loss: 0.4372 - accuracy: 0.4258 - val_loss: 0.4258 - val_accuracy: 0.4922
Epoch 26/100
20/20 - 25s - loss: 0.4253 - accuracy: 0.4750 - val_loss: 0.4203 - val_accuracy: 0.5234
Epoch 27/100
20/20 - 25s - loss: 0.4279 - accuracy: 0.4586 - val_loss: 0.4202 - val_accuracy: 0.5391
Epoch 28/100
20/20 - 25s - loss: 0.4293 - accuracy: 0.4617 - val_loss: 0.4086 - val_accuracy: 0.5312
Epoch 29/100
20/20 - 25s - loss: 0.4451 - accuracy: 0.4375 - val_loss: 0.4144 - val_accuracy: 0.5312
Epoch 30/100
20/20 - 26s - loss: 0.4344 - accuracy: 0.4344 - val_loss: 0.4089 - val_accuracy: 0.5547
Epoch 31/100
20/20 - 26s - loss: 0.4336 - accuracy: 0.4461 - val_loss: 0.4081 - val_accuracy: 0.5312

Epoch 00031: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.
Epoch 32/100
20/20 - 26s - loss: 0.4213 - accuracy: 0.4555 - val_loss: 0.4058 - val_accuracy: 0.5312
Epoch 33/100
20/20 - 26s - loss: 0.4323 - accuracy: 0.4289 - val_loss: 0.4048 - val_accuracy: 0.5312
Epoch 34/100
20/20 - 25s - loss: 0.4211 - accuracy: 0.4440 - val_loss: 0.4011 - val_accuracy: 0.5391
Epoch 35/100
20/20 - 27s - loss: 0.4199 - accuracy: 0.4531 - val_loss: 0.3985 - val_accuracy: 0.5391
Epoch 36/100
20/20 - 27s - loss: 0.4182 - accuracy: 0.4961 - val_loss: 0.3985 - val_accuracy: 0.5391
Epoch 37/100
20/20 - 27s - loss: 0.4195 - accuracy: 0.4664 - val_loss: 0.3966 - val_accuracy: 0.5234
Epoch 38/100
20/20 - 27s - loss: 0.4125 - accuracy: 0.4797 - val_loss: 0.3951 - val_accuracy: 0.5391
Epoch 39/100
20/20 - 26s - loss: 0.4203 - accuracy: 0.4578 - val_loss: 0.4012 - val_accuracy: 0.5625
Epoch 40/100
20/20 - 26s - loss: 0.4250 - accuracy: 0.4547 - val_loss: 0.3930 - val_accuracy: 0.5469
Epoch 41/100

```
20/20 - 27s - loss: 0.4199 - accuracy: 0.4367 - val_loss: 0.3985 - val_accuracy: 0.5547
Epoch 00041: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 42/100
20/20 - 27s - loss: 0.4208 - accuracy: 0.4617 - val_loss: 0.3973 - val_accuracy: 0.5469
Epoch 43/100
20/20 - 27s - loss: 0.4262 - accuracy: 0.4477 - val_loss: 0.3970 - val_accuracy: 0.5469
Epoch 44/100
20/20 - 26s - loss: 0.4280 - accuracy: 0.4547 - val_loss: 0.3969 - val_accuracy: 0.5391
Epoch 45/100
20/20 - 27s - loss: 0.4167 - accuracy: 0.4711 - val_loss: 0.3962 - val_accuracy: 0.5312
Epoch 46/100
20/20 - 27s - loss: 0.4198 - accuracy: 0.4695 - val_loss: 0.3959 - val_accuracy: 0.5312
Restoring model weights from the end of the best epoch.
```

```
Epoch 00046: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
```

```
Epoch 00046: early stopping
```

```
Out[37]: <tensorflow.python.keras.callbacks.History at 0x245abfae490>
```

```
In [38]:
```

```
# Evaluate the model on test set
score = model.evaluate(x_test, y_test_age, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.4230794608592987, 0.47694578766822815]
```

```
In [ ]:
```

```
In [ ]:
```

1-3. Two-layer CNN

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- Change kernel size: kernel size = 5
- Dropout = 0.3

```
In [74]:
```

```
model = tf.keras.Sequential()

# Must define the input shape in the first Layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=5, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=5, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(5, activation='sigmoid'))
```

```
# Take a Look at the model summary  
model.summary()
```

Model: "sequential_4"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_8 (Conv2D)	(None, 128, 128, 64)	4864
<hr/>		
max_pooling2d_8 (MaxPooling2D)	(None, 64, 64, 64)	0
<hr/>		
dropout_13 (Dropout)	(None, 64, 64, 64)	0
<hr/>		
conv2d_9 (Conv2D)	(None, 64, 64, 32)	51232
<hr/>		
max_pooling2d_9 (MaxPooling2D)	(None, 32, 32, 32)	0
<hr/>		
dropout_14 (Dropout)	(None, 32, 32, 32)	0
<hr/>		
flatten_4 (Flatten)	(None, 32768)	0
<hr/>		
dense_9 (Dense)	(None, 256)	8388864
<hr/>		
dropout_15 (Dropout)	(None, 256)	0
<hr/>		
dense_10 (Dense)	(None, 5)	1285
<hr/>		
Total params: 8,446,245		
Trainable params: 8,446,245		
Non-trainable params: 0		

In [29]:

```
import time  
localtime = time.asctime( time.localtime(time.time()) )  
print(localtime)
```

Sun Apr 25 13:51:46 2021

In [30]:

```
model.compile(loss='binary_crossentropy',  
              optimizer='adam',  
              metrics=['accuracy'])
```

In [31]:

```
model.fit(x_train,  
          y_train_age,  
          batch_size=64,  
          epochs=10,  
          validation_data=(x_valid, y_valid_age),)
```

```
Epoch 1/10  
339/339 [=====] - 713s 2s/step - loss: 0.4966 - accuracy: 0.370  
1 - val_loss: 0.4315 - val_accuracy: 0.4469  
Epoch 2/10  
339/339 [=====] - 707s 2s/step - loss: 0.4336 - accuracy: 0.440  
7 - val_loss: 0.4240 - val_accuracy: 0.4786  
Epoch 3/10  
339/339 [=====] - 705s 2s/step - loss: 0.4191 - accuracy: 0.466  
9 - val_loss: 0.4035 - val_accuracy: 0.5199  
Epoch 4/10  
339/339 [=====] - 697s 2s/step - loss: 0.4082 - accuracy: 0.489  
6 - val_loss: 0.3998 - val_accuracy: 0.5170
```

```
Epoch 5/10
339/339 [=====] - 701s 2s/step - loss: 0.3991 - accuracy: 0.505
9 - val_loss: 0.3894 - val_accuracy: 0.5303
Epoch 6/10
339/339 [=====] - 706s 2s/step - loss: 0.3911 - accuracy: 0.511
1 - val_loss: 0.3849 - val_accuracy: 0.5321
Epoch 7/10
339/339 [=====] - 434s 1s/step - loss: 0.3837 - accuracy: 0.528
2 - val_loss: 0.3942 - val_accuracy: 0.5196
Epoch 8/10
339/339 [=====] - 719s 2s/step - loss: 0.3758 - accuracy: 0.535
1 - val_loss: 0.3807 - val_accuracy: 0.5373
Epoch 9/10
339/339 [=====] - 753s 2s/step - loss: 0.3650 - accuracy: 0.555
6 - val_loss: 0.3841 - val_accuracy: 0.5376
Epoch 10/10
339/339 [=====] - 1005s 3s/step - loss: 0.3531 - accuracy: 0.57
40 - val_loss: 0.3897 - val_accuracy: 0.5339
Out[31]: <tensorflow.python.keras.callbacks.History at 0x23afffc11c40>
```

In [33]:

```
# Evaluate the model on test set
score = model.evaluate(x_test, y_test_age, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.39115768671035767, 0.5167834758758545]
```

Early Stop

In [75]:

```
import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

```
Tue Apr 27 18:47:42 2021
```

In [76]:

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [77]:

```
model.fit(x_train,y_train_age,
           validation_data=(x_valid , y_valid_age),
           batch_size=64,
           steps_per_epoch=20,
           epochs=100,
           validation_steps=2,
           verbose=2,
           workers=1,
           use_multiprocessing = False,
           callbacks=[earlystop, plateau])
```

```
Epoch 1/100
20/20 - 402s - loss: 0.5998 - accuracy: 0.3195 - val_loss: 0.4908 - val_accuracy: 0.4688
Epoch 2/100
20/20 - 215s - loss: 0.4760 - accuracy: 0.3539 - val_loss: 0.5180 - val_accuracy: 0.4609
Epoch 3/100
20/20 - 222s - loss: 0.4755 - accuracy: 0.3586 - val_loss: 0.5615 - val_accuracy: 0.4609
Epoch 4/100
```

20/20 - 239s - loss: 0.4629 - accuracy: 0.3789 - val_loss: 0.5216 - val_accuracy: 0.4609
Epoch 5/100
20/20 - 244s - loss: 0.4729 - accuracy: 0.3805 - val_loss: 0.5426 - val_accuracy: 0.4609
Epoch 6/100
20/20 - 250s - loss: 0.4728 - accuracy: 0.3695 - val_loss: 0.5539 - val_accuracy: 0.4609
Epoch 7/100
20/20 - 263s - loss: 0.4568 - accuracy: 0.3727 - val_loss: 0.4592 - val_accuracy: 0.4609
Epoch 8/100
20/20 - 273s - loss: 0.4612 - accuracy: 0.4070 - val_loss: 0.4808 - val_accuracy: 0.4609
Epoch 9/100
20/20 - 278s - loss: 0.4517 - accuracy: 0.4164 - val_loss: 0.4814 - val_accuracy: 0.4688
Epoch 10/100
20/20 - 474s - loss: 0.4507 - accuracy: 0.3961 - val_loss: 0.4951 - val_accuracy: 0.5078
Epoch 11/100
20/20 - 585s - loss: 0.4534 - accuracy: 0.3977 - val_loss: 0.4755 - val_accuracy: 0.4844
Epoch 12/100
20/20 - 647s - loss: 0.4455 - accuracy: 0.4297 - val_loss: 0.4358 - val_accuracy: 0.5312
Epoch 13/100
20/20 - 676s - loss: 0.4537 - accuracy: 0.4070 - val_loss: 0.4266 - val_accuracy: 0.5156
Epoch 14/100
20/20 - 348s - loss: 0.4464 - accuracy: 0.4156 - val_loss: 0.4376 - val_accuracy: 0.5312
Epoch 15/100
20/20 - 322s - loss: 0.4375 - accuracy: 0.4266 - val_loss: 0.4189 - val_accuracy: 0.5234
Epoch 16/100
20/20 - 327s - loss: 0.4362 - accuracy: 0.4406 - val_loss: 0.4458 - val_accuracy: 0.5156
Epoch 17/100
20/20 - 234s - loss: 0.4453 - accuracy: 0.4125 - val_loss: 0.4279 - val_accuracy: 0.5312
Epoch 18/100
20/20 - 25s - loss: 0.4422 - accuracy: 0.4219 - val_loss: 0.4169 - val_accuracy: 0.5234
Epoch 19/100
20/20 - 24s - loss: 0.4358 - accuracy: 0.4352 - val_loss: 0.4215 - val_accuracy: 0.5391
Epoch 20/100
20/20 - 25s - loss: 0.4374 - accuracy: 0.4359 - val_loss: 0.4115 - val_accuracy: 0.5156
Epoch 21/100
20/20 - 24s - loss: 0.4376 - accuracy: 0.4461 - val_loss: 0.4193 - val_accuracy: 0.5156
Epoch 22/100
20/20 - 24s - loss: 0.4390 - accuracy: 0.4234 - val_loss: 0.4179 - val_accuracy: 0.5312
Epoch 23/100
20/20 - 24s - loss: 0.4254 - accuracy: 0.4656 - val_loss: 0.4125 - val_accuracy: 0.5391
Epoch 24/100
20/20 - 24s - loss: 0.4319 - accuracy: 0.4344 - val_loss: 0.4056 - val_accuracy: 0.5312
Epoch 25/100
20/20 - 24s - loss: 0.4241 - accuracy: 0.4789 - val_loss: 0.4053 - val_accuracy: 0.5234
Epoch 26/100
20/20 - 24s - loss: 0.4284 - accuracy: 0.4461 - val_loss: 0.4022 - val_accuracy: 0.5156
Epoch 27/100
20/20 - 23s - loss: 0.4258 - accuracy: 0.4320 - val_loss: 0.4057 - val_accuracy: 0.5312
Epoch 28/100
20/20 - 23s - loss: 0.4376 - accuracy: 0.4117 - val_loss: 0.4004 - val_accuracy: 0.5391
Epoch 29/100
20/20 - 23s - loss: 0.4317 - accuracy: 0.4516 - val_loss: 0.3946 - val_accuracy: 0.5234
Epoch 30/100
20/20 - 23s - loss: 0.4256 - accuracy: 0.4586 - val_loss: 0.4062 - val_accuracy: 0.5625

Epoch 00030: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.

Epoch 31/100
20/20 - 23s - loss: 0.4293 - accuracy: 0.4398 - val_loss: 0.4013 - val_accuracy: 0.5312
Epoch 32/100
20/20 - 23s - loss: 0.4314 - accuracy: 0.4359 - val_loss: 0.4007 - val_accuracy: 0.5391
Epoch 33/100
20/20 - 23s - loss: 0.4286 - accuracy: 0.4398 - val_loss: 0.3986 - val_accuracy: 0.5547
Epoch 34/100
20/20 - 23s - loss: 0.4214 - accuracy: 0.4629 - val_loss: 0.3977 - val_accuracy: 0.5469
Epoch 35/100
20/20 - 23s - loss: 0.4246 - accuracy: 0.4484 - val_loss: 0.3956 - val_accuracy: 0.5391

```
Restoring model weights from the end of the best epoch.  
Epoch 00035: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.  
Epoch 00035: early stopping  
Out[77]: <tensorflow.python.keras.callbacks.History at 0x188b676fd90>
```

```
In [78]:  
# Evaluate the model on test set  
score = model.evaluate(x_test, y_test_age, verbose=0)  
  
# Print test accuracy  
print('\n', 'Test accuracy:', score)
```

Test accuracy: [0.4309602975845337, 0.4588712751865387]

Build Multilayer CNN (change fully connected layer)

2-2. Two-layer CNN (3 fully connected layers)

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- kernel size = 2
- Dropout = 0.3

```
In [79]:  
model = tf.keras.Sequential()  
  
# Must define the input shape in the first layer of the neural network  
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu'))  
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))  
model.add(tf.keras.layers.Dropout(0.3))  
  
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))  
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))  
model.add(tf.keras.layers.Dropout(0.3))  
  
model.add(tf.keras.layers.Flatten())  
model.add(tf.keras.layers.Dense(256, activation='relu'))  
model.add(tf.keras.layers.Dropout(0.5))  
model.add(tf.keras.layers.Dense(128, activation='relu'))  
model.add(tf.keras.layers.Dropout(0.5))  
model.add(tf.keras.layers.Dense(5, activation='sigmoid'))  
  
# Take a look at the model summary  
model.summary()
```

Model: "sequential_5"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_10 (Conv2D)	(None, 128, 128, 64)	832
<hr/>		
max_pooling2d_10 (MaxPooling2D)	(None, 64, 64, 64)	0
<hr/>		
dropout_16 (Dropout)	(None, 64, 64, 64)	0
<hr/>		
conv2d_11 (Conv2D)	(None, 64, 64, 32)	8224
<hr/>		
max_pooling2d_11 (MaxPooling2D)	(None, 32, 32, 32)	0

dropout_17 (Dropout)	(None, 32, 32, 32)	0
flatten_5 (Flatten)	(None, 32768)	0
dense_11 (Dense)	(None, 256)	8388864
dropout_18 (Dropout)	(None, 256)	0
dense_12 (Dense)	(None, 128)	32896
dropout_19 (Dropout)	(None, 128)	0
dense_13 (Dense)	(None, 5)	645
<hr/>		
Total params: 8,431,461		
Trainable params: 8,431,461		
Non-trainable params: 0		

In [23]:

```
import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

Tue Apr 27 02:53:04 2021

In [24]:

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

First 10 epochs

In [25]:

```
model.fit(x_train,
          y_train_age,
          batch_size=64,
          epochs=10,
          validation_data=(x_valid, y_valid_age),)
```

```
Epoch 1/10
339/339 [=====] - 394s 1s/step - loss: 0.5370 - accuracy: 0.352
7 - val_loss: 0.4890 - val_accuracy: 0.4041
Epoch 2/10
339/339 [=====] - 378s 1s/step - loss: 0.4562 - accuracy: 0.397
6 - val_loss: 0.4400 - val_accuracy: 0.4269
Epoch 3/10
339/339 [=====] - 378s 1s/step - loss: 0.4398 - accuracy: 0.425
2 - val_loss: 0.4166 - val_accuracy: 0.4720
Epoch 4/10
339/339 [=====] - 382s 1s/step - loss: 0.4265 - accuracy: 0.453
7 - val_loss: 0.4096 - val_accuracy: 0.4867
Epoch 5/10
339/339 [=====] - 382s 1s/step - loss: 0.4210 - accuracy: 0.459
6 - val_loss: 0.4035 - val_accuracy: 0.4974
Epoch 6/10
339/339 [=====] - 388s 1s/step - loss: 0.4141 - accuracy: 0.474
3 - val_loss: 0.4001 - val_accuracy: 0.4978
Epoch 7/10
339/339 [=====] - 368s 1s/step - loss: 0.4104 - accuracy: 0.477
4 - val_loss: 0.4035 - val_accuracy: 0.4926
Epoch 8/10
339/339 [=====] - 366s 1s/step - loss: 0.4028 - accuracy: 0.492
```

```
4 - val_loss: 0.3962 - val_accuracy: 0.5048
Epoch 9/10
339/339 [=====] - 363s 1s/step - loss: 0.3972 - accuracy: 0.502
8 - val_loss: 0.3981 - val_accuracy: 0.5111
Epoch 10/10
339/339 [=====] - 367s 1s/step - loss: 0.3896 - accuracy: 0.513
4 - val_loss: 0.3903 - val_accuracy: 0.5192
Out[25]: <tensorflow.python.keras.callbacks.History at 0x18844ef24f0>
```

```
In [26]: # Evaluate the model on test set
score = model.evaluate(x_test, y_test_age, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.3917633891105652, 0.5112504363059998]
```

Second 10 epochs

```
In [27]: model.fit(x_train,
                 y_train_age,
                 batch_size=64,
                 epochs=10,
                 validation_data=(x_valid, y_valid_age),)

Epoch 1/10
339/339 [=====] - 365s 1s/step - loss: 0.3842 - accuracy: 0.521
6 - val_loss: 0.3896 - val_accuracy: 0.5214
Epoch 2/10
339/339 [=====] - 363s 1s/step - loss: 0.3782 - accuracy: 0.533
6 - val_loss: 0.3914 - val_accuracy: 0.5159
Epoch 3/10
339/339 [=====] - 362s 1s/step - loss: 0.3708 - accuracy: 0.545
8 - val_loss: 0.3902 - val_accuracy: 0.5203
Epoch 4/10
339/339 [=====] - 362s 1s/step - loss: 0.3617 - accuracy: 0.561
0 - val_loss: 0.3907 - val_accuracy: 0.5251
Epoch 5/10
339/339 [=====] - 363s 1s/step - loss: 0.3562 - accuracy: 0.565
5 - val_loss: 0.3910 - val_accuracy: 0.5229
Epoch 6/10
339/339 [=====] - 364s 1s/step - loss: 0.3482 - accuracy: 0.576
3 - val_loss: 0.4013 - val_accuracy: 0.5196
Epoch 7/10
339/339 [=====] - 360s 1s/step - loss: 0.3399 - accuracy: 0.592
8 - val_loss: 0.3940 - val_accuracy: 0.5269
Epoch 8/10
339/339 [=====] - 363s 1s/step - loss: 0.3353 - accuracy: 0.597
4 - val_loss: 0.3973 - val_accuracy: 0.5148
Epoch 9/10
339/339 [=====] - 360s 1s/step - loss: 0.3276 - accuracy: 0.611
5 - val_loss: 0.3969 - val_accuracy: 0.5181
Epoch 10/10
339/339 [=====] - 361s 1s/step - loss: 0.3216 - accuracy: 0.618
7 - val_loss: 0.3982 - val_accuracy: 0.5214
Out[27]: <tensorflow.python.keras.callbacks.History at 0x1887fe18d30>
```

```
In [28]: # Evaluate the model on test set
score = model.evaluate(x_test, y_test_age, verbose=0)
```

```
# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.40269163250923157, 0.5060862898826599]
```

Early Stop

In [80]:

```
import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

```
Tue Apr 27 20:34:53 2021
```

In [81]:

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [82]:

```
model.fit(x_train,y_train_age,
           validation_data=(x_valid , y_valid_age),
           batch_size=64,
           steps_per_epoch=20,
           epochs=100,
           validation_steps=2,
           verbose=2,
           workers=1,
           use_multiprocessing = False,
           callbacks=[earlystop, plateau])
```

```
Epoch 1/100
20/20 - 15s - loss: 0.7198 - accuracy: 0.3031 - val_loss: 0.6338 - val_accuracy: 0.4609
Epoch 2/100
20/20 - 13s - loss: 0.4997 - accuracy: 0.3547 - val_loss: 0.5733 - val_accuracy: 0.4609
Epoch 3/100
20/20 - 13s - loss: 0.4959 - accuracy: 0.3203 - val_loss: 0.6284 - val_accuracy: 0.4609
Epoch 4/100
20/20 - 13s - loss: 0.4789 - accuracy: 0.3867 - val_loss: 0.5857 - val_accuracy: 0.4609
Epoch 5/100
20/20 - 13s - loss: 0.4776 - accuracy: 0.3914 - val_loss: 0.5935 - val_accuracy: 0.4609
Epoch 6/100
20/20 - 13s - loss: 0.4760 - accuracy: 0.3773 - val_loss: 0.5667 - val_accuracy: 0.4609
Epoch 7/100
20/20 - 13s - loss: 0.4781 - accuracy: 0.3531 - val_loss: 0.5383 - val_accuracy: 0.4609
Epoch 8/100
20/20 - 13s - loss: 0.4710 - accuracy: 0.3844 - val_loss: 0.5471 - val_accuracy: 0.4609
Epoch 9/100
20/20 - 13s - loss: 0.4618 - accuracy: 0.3930 - val_loss: 0.5100 - val_accuracy: 0.4609
Epoch 10/100
20/20 - 13s - loss: 0.4704 - accuracy: 0.3750 - val_loss: 0.5101 - val_accuracy: 0.4609
Epoch 11/100
20/20 - 13s - loss: 0.4619 - accuracy: 0.3977 - val_loss: 0.4962 - val_accuracy: 0.4609
Epoch 12/100
20/20 - 12s - loss: 0.4636 - accuracy: 0.3898 - val_loss: 0.5010 - val_accuracy: 0.4609
Epoch 13/100
20/20 - 13s - loss: 0.4563 - accuracy: 0.4195 - val_loss: 0.4961 - val_accuracy: 0.4609
Epoch 14/100
20/20 - 13s - loss: 0.4581 - accuracy: 0.4055 - val_loss: 0.4718 - val_accuracy: 0.4609
Epoch 15/100
20/20 - 13s - loss: 0.4564 - accuracy: 0.3938 - val_loss: 0.4725 - val_accuracy: 0.4844
Epoch 16/100
```

```
20/20 - 13s - loss: 0.4561 - accuracy: 0.4187 - val_loss: 0.4771 - val_accuracy: 0.4922
Epoch 17/100
20/20 - 13s - loss: 0.4503 - accuracy: 0.4109 - val_loss: 0.4549 - val_accuracy: 0.4844
Epoch 18/100
20/20 - 13s - loss: 0.4587 - accuracy: 0.3797 - val_loss: 0.4486 - val_accuracy: 0.4766

Epoch 00018: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.
Epoch 19/100
20/20 - 13s - loss: 0.4589 - accuracy: 0.3984 - val_loss: 0.4538 - val_accuracy: 0.4766
Epoch 20/100
20/20 - 13s - loss: 0.4474 - accuracy: 0.4156 - val_loss: 0.4495 - val_accuracy: 0.4844
Epoch 21/100
20/20 - 13s - loss: 0.4457 - accuracy: 0.4328 - val_loss: 0.4446 - val_accuracy: 0.4844
Epoch 22/100
20/20 - 13s - loss: 0.4562 - accuracy: 0.3898 - val_loss: 0.4478 - val_accuracy: 0.4844
Epoch 23/100
20/20 - 13s - loss: 0.4458 - accuracy: 0.4258 - val_loss: 0.4434 - val_accuracy: 0.4844
Epoch 24/100
20/20 - 13s - loss: 0.4470 - accuracy: 0.4289 - val_loss: 0.4415 - val_accuracy: 0.4922
Epoch 25/100
20/20 - 13s - loss: 0.4460 - accuracy: 0.4141 - val_loss: 0.4379 - val_accuracy: 0.4922
Epoch 26/100
20/20 - 14s - loss: 0.4484 - accuracy: 0.4023 - val_loss: 0.4375 - val_accuracy: 0.4922

Epoch 00026: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 27/100
20/20 - 14s - loss: 0.4527 - accuracy: 0.4008 - val_loss: 0.4380 - val_accuracy: 0.4922
Epoch 28/100
20/20 - 14s - loss: 0.4472 - accuracy: 0.4273 - val_loss: 0.4381 - val_accuracy: 0.4922
Epoch 29/100
20/20 - 14s - loss: 0.4474 - accuracy: 0.4227 - val_loss: 0.4380 - val_accuracy: 0.4922
Epoch 30/100
20/20 - 14s - loss: 0.4453 - accuracy: 0.4141 - val_loss: 0.4380 - val_accuracy: 0.4922
Epoch 31/100
20/20 - 14s - loss: 0.4467 - accuracy: 0.4039 - val_loss: 0.4382 - val_accuracy: 0.4922
Restoring model weights from the end of the best epoch.
```

```
Epoch 00031: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
Epoch 00031: early stopping
```

```
Out[82]: <tensorflow.python.keras.callbacks.History at 0x188bcf1e100>
```

```
In [83]: # Evaluate the model on test set
score = model.evaluate(x_test, y_test_age, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.4609558582305908, 0.4142382740974426]
```

```
In [84]: labels =[ "0-18 years old", # index 0
              "18-30 years old", # index 1
              "30-40 years old", # index 2
              "40-60 years old", # index 3
              "60-100 years old", # index 4
            ]
```

```
In [85]: y_hat = model.predict(x_test)

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
```

```

for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test_age[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                 color=("green" if predict_index == true_index else "red"))
plt.show()

```



In []:

Build Multilayer CNN (change fully connected layer)

2-3. Two-layer CNN (2 fully connected layers)

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- kernel size = 2
- Dropout = 0.3 for convolutional layers

In [86]:

```

model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(64, activation='relu'))

```

```

model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(5, activation='sigmoid'))

# Take a look at the model summary
model.summary()

```

Model: "sequential_6"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_12 (Conv2D)	(None, 128, 128, 64)	832
<hr/>		
max_pooling2d_12 (MaxPooling)	(None, 64, 64, 64)	0
<hr/>		
dropout_20 (Dropout)	(None, 64, 64, 64)	0
<hr/>		
conv2d_13 (Conv2D)	(None, 64, 64, 32)	8224
<hr/>		
max_pooling2d_13 (MaxPooling)	(None, 32, 32, 32)	0
<hr/>		
dropout_21 (Dropout)	(None, 32, 32, 32)	0
<hr/>		
flatten_6 (Flatten)	(None, 32768)	0
<hr/>		
dense_14 (Dense)	(None, 128)	4194432
<hr/>		
dropout_22 (Dropout)	(None, 128)	0
<hr/>		
dense_15 (Dense)	(None, 64)	8256
<hr/>		
dropout_23 (Dropout)	(None, 64)	0
<hr/>		
dense_16 (Dense)	(None, 5)	325
<hr/>		
Total params: 4,212,069		
Trainable params: 4,212,069		
Non-trainable params: 0		

In [87]:

```

import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)

```

Tue Apr 27 20:41:53 2021

In [88]:

```

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

```

In [89]:

```

model.fit(x_train,y_train_age,
           validation_data=(x_valid , y_valid_age),
           batch_size=64,
           steps_per_epoch=20,
           epochs=100,
           validation_steps=2,
           verbose=2,
           workers=1,
           use_multiprocessing = False,
           callbacks=[earlystop, plateau])

```

Epoch 1/100
20/20 - 38s - loss: 0.6107 - accuracy: 0.2117 - val_loss: 0.5257 - val_accuracy: 0.2734
Epoch 2/100
20/20 - 93s - loss: 0.5214 - accuracy: 0.3117 - val_loss: 0.5168 - val_accuracy: 0.4609
Epoch 3/100
20/20 - 171s - loss: 0.5084 - accuracy: 0.3313 - val_loss: 0.5163 - val_accuracy: 0.4609
Epoch 4/100
20/20 - 232s - loss: 0.4937 - accuracy: 0.3414 - val_loss: 0.4983 - val_accuracy: 0.4609
Epoch 5/100
20/20 - 300s - loss: 0.4873 - accuracy: 0.3570 - val_loss: 0.5091 - val_accuracy: 0.4609
Epoch 6/100
20/20 - 365s - loss: 0.4889 - accuracy: 0.3414 - val_loss: 0.4900 - val_accuracy: 0.4609
Epoch 7/100
20/20 - 439s - loss: 0.4811 - accuracy: 0.3602 - val_loss: 0.5185 - val_accuracy: 0.4609
Epoch 8/100
20/20 - 484s - loss: 0.4813 - accuracy: 0.3625 - val_loss: 0.4767 - val_accuracy: 0.4609
Epoch 9/100
20/20 - 362s - loss: 0.4719 - accuracy: 0.3828 - val_loss: 0.4937 - val_accuracy: 0.4609
Epoch 10/100
20/20 - 307s - loss: 0.4651 - accuracy: 0.3977 - val_loss: 0.4776 - val_accuracy: 0.4609
Epoch 11/100
20/20 - 378s - loss: 0.4677 - accuracy: 0.4039 - val_loss: 0.4676 - val_accuracy: 0.4609
Epoch 12/100
20/20 - 399s - loss: 0.4619 - accuracy: 0.3938 - val_loss: 0.4559 - val_accuracy: 0.4609
Epoch 13/100
20/20 - 424s - loss: 0.4623 - accuracy: 0.3859 - val_loss: 0.4524 - val_accuracy: 0.4609
Epoch 14/100
20/20 - 469s - loss: 0.4628 - accuracy: 0.3914 - val_loss: 0.4600 - val_accuracy: 0.4688
Epoch 15/100
20/20 - 736s - loss: 0.4618 - accuracy: 0.4047 - val_loss: 0.4789 - val_accuracy: 0.4688
Epoch 16/100
20/20 - 634s - loss: 0.4628 - accuracy: 0.4109 - val_loss: 0.4432 - val_accuracy: 0.4766
Epoch 17/100
20/20 - 378s - loss: 0.4588 - accuracy: 0.4156 - val_loss: 0.4439 - val_accuracy: 0.4766
Epoch 18/100
20/20 - 30s - loss: 0.4563 - accuracy: 0.4148 - val_loss: 0.4526 - val_accuracy: 0.4688
Epoch 19/100
20/20 - 34s - loss: 0.4539 - accuracy: 0.4133 - val_loss: 0.4429 - val_accuracy: 0.4688
Epoch 20/100
20/20 - 28s - loss: 0.4530 - accuracy: 0.4125 - val_loss: 0.4295 - val_accuracy: 0.4766
Epoch 21/100
20/20 - 33s - loss: 0.4589 - accuracy: 0.4125 - val_loss: 0.4350 - val_accuracy: 0.5000
Epoch 22/100
20/20 - 30s - loss: 0.4469 - accuracy: 0.4320 - val_loss: 0.4258 - val_accuracy: 0.4688
Epoch 23/100
20/20 - 29s - loss: 0.4471 - accuracy: 0.4305 - val_loss: 0.4286 - val_accuracy: 0.5156
Epoch 24/100
20/20 - 36s - loss: 0.4506 - accuracy: 0.4008 - val_loss: 0.4298 - val_accuracy: 0.5156
Epoch 25/100
20/20 - 29s - loss: 0.4492 - accuracy: 0.4297 - val_loss: 0.4150 - val_accuracy: 0.5312
Epoch 26/100
20/20 - 29s - loss: 0.4482 - accuracy: 0.4117 - val_loss: 0.4246 - val_accuracy: 0.5000
Epoch 27/100
20/20 - 31s - loss: 0.4402 - accuracy: 0.4313 - val_loss: 0.4221 - val_accuracy: 0.5234

Epoch 00027: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.

Epoch 28/100
20/20 - 36s - loss: 0.4365 - accuracy: 0.4422 - val_loss: 0.4120 - val_accuracy: 0.5391
Epoch 29/100
20/20 - 29s - loss: 0.4373 - accuracy: 0.4313 - val_loss: 0.4090 - val_accuracy: 0.5391
Epoch 30/100
20/20 - 34s - loss: 0.4456 - accuracy: 0.4164 - val_loss: 0.4099 - val_accuracy: 0.5312
Epoch 31/100
20/20 - 24s - loss: 0.4387 - accuracy: 0.4484 - val_loss: 0.4073 - val_accuracy: 0.5234
Epoch 32/100

```
20/20 - 29s - loss: 0.4410 - accuracy: 0.4328 - val_loss: 0.4055 - val_accuracy: 0.5234
Epoch 33/100
20/20 - 31s - loss: 0.4488 - accuracy: 0.4297 - val_loss: 0.4046 - val_accuracy: 0.5234
Epoch 34/100
20/20 - 22s - loss: 0.4386 - accuracy: 0.4440 - val_loss: 0.4062 - val_accuracy: 0.5234
Epoch 35/100
20/20 - 13s - loss: 0.4303 - accuracy: 0.4656 - val_loss: 0.4066 - val_accuracy: 0.5156
Epoch 36/100
20/20 - 12s - loss: 0.4404 - accuracy: 0.4313 - val_loss: 0.4049 - val_accuracy: 0.5234
Epoch 37/100
20/20 - 12s - loss: 0.4462 - accuracy: 0.4164 - val_loss: 0.4073 - val_accuracy: 0.5234
Epoch 38/100
20/20 - 13s - loss: 0.4283 - accuracy: 0.4711 - val_loss: 0.4010 - val_accuracy: 0.5234
Epoch 39/100
20/20 - 12s - loss: 0.4360 - accuracy: 0.4281 - val_loss: 0.4027 - val_accuracy: 0.5234
Epoch 40/100
20/20 - 13s - loss: 0.4396 - accuracy: 0.4297 - val_loss: 0.4036 - val_accuracy: 0.5234
Epoch 41/100
20/20 - 13s - loss: 0.4344 - accuracy: 0.4523 - val_loss: 0.4019 - val_accuracy: 0.5234
Epoch 42/100
20/20 - 12s - loss: 0.4315 - accuracy: 0.4656 - val_loss: 0.4019 - val_accuracy: 0.5234
Epoch 43/100
20/20 - 12s - loss: 0.4314 - accuracy: 0.4445 - val_loss: 0.3995 - val_accuracy: 0.5234

Epoch 00043: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 44/100
20/20 - 13s - loss: 0.4411 - accuracy: 0.4391 - val_loss: 0.3996 - val_accuracy: 0.5234
Epoch 45/100
20/20 - 13s - loss: 0.4369 - accuracy: 0.4555 - val_loss: 0.3998 - val_accuracy: 0.5234
Epoch 46/100
20/20 - 12s - loss: 0.4365 - accuracy: 0.4398 - val_loss: 0.3998 - val_accuracy: 0.5234
Epoch 47/100
20/20 - 12s - loss: 0.4362 - accuracy: 0.4266 - val_loss: 0.3997 - val_accuracy: 0.5234
Epoch 48/100
20/20 - 12s - loss: 0.4383 - accuracy: 0.4406 - val_loss: 0.4000 - val_accuracy: 0.5234
Restoring model weights from the end of the best epoch.
```

```
Epoch 00048: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
Epoch 00048: early stopping
```

```
Out[89]: <tensorflow.python.keras.callbacks.History at 0x188bd221370>
```

```
In [37]:
```

```
model.fit(x_train,
           y_train_age,
           batch_size=64,
           epochs=10,
           validation_data=(x_valid, y_valid_age),)
```

```
Epoch 1/10
339/339 [=====] - 222s 647ms/step - loss: 0.5197 - accuracy: 0.
3618 - val_loss: 0.4619 - val_accuracy: 0.4435
Epoch 2/10
339/339 [=====] - 220s 648ms/step - loss: 0.4307 - accuracy: 0.
4425 - val_loss: 0.4238 - val_accuracy: 0.4989
Epoch 3/10
339/339 [=====] - 217s 639ms/step - loss: 0.4197 - accuracy: 0.
4723 - val_loss: 0.4065 - val_accuracy: 0.5063
Epoch 4/10
339/339 [=====] - 221s 653ms/step - loss: 0.4122 - accuracy: 0.
4841 - val_loss: 0.3995 - val_accuracy: 0.5210
Epoch 5/10
339/339 [=====] - 244s 721ms/step - loss: 0.4030 - accuracy: 0.
5027 - val_loss: 0.3978 - val_accuracy: 0.5207
Epoch 6/10
```

```
339/339 [=====] - 378s 1s/step - loss: 0.3964 - accuracy: 0.502
8 - val_loss: 0.3873 - val_accuracy: 0.5251
Epoch 7/10
339/339 [=====] - 305s 900ms/step - loss: 0.3894 - accuracy: 0.
5148 - val_loss: 0.3884 - val_accuracy: 0.5185
Epoch 8/10
339/339 [=====] - 235s 694ms/step - loss: 0.3802 - accuracy: 0.
5252 - val_loss: 0.3828 - val_accuracy: 0.5277
Epoch 9/10
339/339 [=====] - 220s 648ms/step - loss: 0.3683 - accuracy: 0.
5533 - val_loss: 0.3873 - val_accuracy: 0.5192
Epoch 10/10
339/339 [=====] - 216s 637ms/step - loss: 0.3583 - accuracy: 0.
5612 - val_loss: 0.3943 - val_accuracy: 0.5203
Out[37]: <tensorflow.python.keras.callbacks.History at 0x1887fe53700>
```

```
In [90]: # Evaluate the model on test set
score = model.evaluate(x_test, y_test_age, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

Test accuracy: [0.42414218187332153, 0.45370712876319885]

```
In [ ]:
```

Build the multilayer CNN (change dropout probabilities)

3-2. Two-layer CNN (2 fully connected layers)

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- kernel size = 2
- Dropout = 0.2 for covolutional layers

```
In [39]: model = tf.keras.Sequential()

# Must define the input shape in the first Layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.2))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.2))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(5, activation='sigmoid'))

# Take a Look at the model summary
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_4 (Conv2D)	(None, 128, 128, 64)	832
<hr/>		
max_pooling2d_4 (MaxPooling2D)	(None, 64, 64, 64)	0
<hr/>		
dropout_6 (Dropout)	(None, 64, 64, 64)	0
<hr/>		
conv2d_5 (Conv2D)	(None, 64, 64, 32)	8224
<hr/>		
max_pooling2d_5 (MaxPooling2D)	(None, 32, 32, 32)	0
<hr/>		
dropout_7 (Dropout)	(None, 32, 32, 32)	0
<hr/>		
flatten_2 (Flatten)	(None, 32768)	0
<hr/>		
dense_4 (Dense)	(None, 256)	8388864
<hr/>		
dropout_8 (Dropout)	(None, 256)	0
<hr/>		
dense_5 (Dense)	(None, 5)	1285
<hr/>		
Total params: 8,399,205		
Trainable params: 8,399,205		
Non-trainable params: 0		

In [40]:

```
import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

Tue Apr 27 23:34:25 2021

In [41]:

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [42]:

```
model.fit(x_train,y_train_age,
           validation_data=(x_valid , y_valid_age),
           batch_size=64,
           steps_per_epoch=20,
           epochs=100,
           validation_steps=2,
           verbose=2,
           workers=1,
           use_multiprocessing = False,
           callbacks=[earlystop, plateau])
```

```
Epoch 1/100
20/20 - 23s - loss: 0.7890 - accuracy: 0.3016 - val_loss: 0.5737 - val_accuracy: 0.4609
Epoch 2/100
20/20 - 21s - loss: 0.4840 - accuracy: 0.3742 - val_loss: 0.5139 - val_accuracy: 0.4609
Epoch 3/100
20/20 - 21s - loss: 0.4801 - accuracy: 0.3805 - val_loss: 0.5432 - val_accuracy: 0.4609
Epoch 4/100
20/20 - 21s - loss: 0.4688 - accuracy: 0.3617 - val_loss: 0.5086 - val_accuracy: 0.4609
Epoch 5/100
20/20 - 21s - loss: 0.4666 - accuracy: 0.3719 - val_loss: 0.4824 - val_accuracy: 0.4844
Epoch 6/100
20/20 - 21s - loss: 0.4582 - accuracy: 0.3875 - val_loss: 0.5050 - val_accuracy: 0.4766
```

Epoch 7/100
20/20 - 21s - loss: 0.4582 - accuracy: 0.3672 - val_loss: 0.4915 - val_accuracy: 0.4922
Epoch 8/100
20/20 - 21s - loss: 0.4613 - accuracy: 0.3891 - val_loss: 0.5321 - val_accuracy: 0.4688
Epoch 9/100
20/20 - 21s - loss: 0.4598 - accuracy: 0.4055 - val_loss: 0.4964 - val_accuracy: 0.4844
Epoch 10/100
20/20 - 20s - loss: 0.4502 - accuracy: 0.3836 - val_loss: 0.4989 - val_accuracy: 0.4766
Epoch 11/100
20/20 - 20s - loss: 0.4519 - accuracy: 0.3883 - val_loss: 0.4981 - val_accuracy: 0.5078
Epoch 12/100
20/20 - 20s - loss: 0.4532 - accuracy: 0.4133 - val_loss: 0.4961 - val_accuracy: 0.5000
Epoch 13/100
20/20 - 20s - loss: 0.4447 - accuracy: 0.4031 - val_loss: 0.4898 - val_accuracy: 0.4844
Epoch 14/100
20/20 - 20s - loss: 0.4373 - accuracy: 0.4406 - val_loss: 0.4701 - val_accuracy: 0.4922
Epoch 15/100
20/20 - 20s - loss: 0.4430 - accuracy: 0.4258 - val_loss: 0.4458 - val_accuracy: 0.4844
Epoch 16/100
20/20 - 20s - loss: 0.4435 - accuracy: 0.4398 - val_loss: 0.4549 - val_accuracy: 0.5000
Epoch 17/100
20/20 - 21s - loss: 0.4363 - accuracy: 0.4369 - val_loss: 0.4374 - val_accuracy: 0.5000
Epoch 18/100
20/20 - 21s - loss: 0.4387 - accuracy: 0.4375 - val_loss: 0.4386 - val_accuracy: 0.5156
Epoch 19/100
20/20 - 20s - loss: 0.4410 - accuracy: 0.4328 - val_loss: 0.4393 - val_accuracy: 0.5312

Epoch 00019: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.

Epoch 20/100
20/20 - 20s - loss: 0.4265 - accuracy: 0.4523 - val_loss: 0.4339 - val_accuracy: 0.5234
Epoch 21/100
20/20 - 20s - loss: 0.4277 - accuracy: 0.4500 - val_loss: 0.4290 - val_accuracy: 0.5078
Epoch 22/100
20/20 - 20s - loss: 0.4298 - accuracy: 0.4523 - val_loss: 0.4268 - val_accuracy: 0.5078
Epoch 23/100
20/20 - 20s - loss: 0.4263 - accuracy: 0.4484 - val_loss: 0.4248 - val_accuracy: 0.5234
Epoch 24/100
20/20 - 20s - loss: 0.4308 - accuracy: 0.4508 - val_loss: 0.4298 - val_accuracy: 0.5156
Epoch 25/100
20/20 - 20s - loss: 0.4249 - accuracy: 0.4570 - val_loss: 0.4224 - val_accuracy: 0.5078
Epoch 26/100
20/20 - 20s - loss: 0.4195 - accuracy: 0.4766 - val_loss: 0.4202 - val_accuracy: 0.5234
Epoch 27/100
20/20 - 19s - loss: 0.4310 - accuracy: 0.4453 - val_loss: 0.4253 - val_accuracy: 0.5078
Epoch 28/100
20/20 - 20s - loss: 0.4285 - accuracy: 0.4352 - val_loss: 0.4222 - val_accuracy: 0.5156
Epoch 29/100
20/20 - 20s - loss: 0.4280 - accuracy: 0.4461 - val_loss: 0.4253 - val_accuracy: 0.5078
Epoch 30/100
20/20 - 20s - loss: 0.4229 - accuracy: 0.4750 - val_loss: 0.4257 - val_accuracy: 0.5078
Epoch 31/100
20/20 - 21s - loss: 0.4256 - accuracy: 0.4750 - val_loss: 0.4220 - val_accuracy: 0.5078

Epoch 00031: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.

Epoch 32/100
20/20 - 20s - loss: 0.4256 - accuracy: 0.4594 - val_loss: 0.4219 - val_accuracy: 0.5000
Epoch 33/100
20/20 - 20s - loss: 0.4328 - accuracy: 0.4477 - val_loss: 0.4225 - val_accuracy: 0.5078
Epoch 34/100
20/20 - 20s - loss: 0.4320 - accuracy: 0.4503 - val_loss: 0.4231 - val_accuracy: 0.5078
Epoch 35/100
20/20 - 21s - loss: 0.4206 - accuracy: 0.4727 - val_loss: 0.4222 - val_accuracy: 0.5078
Epoch 36/100
20/20 - 20s - loss: 0.4269 - accuracy: 0.4656 - val_loss: 0.4214 - val_accuracy: 0.5078
Restoring model weights from the end of the best epoch.

```
Epoch 00036: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.  
Epoch 00036: early stopping  
Out[42]: <tensorflow.python.keras.callbacks.History at 0x245ac73cccd0>
```

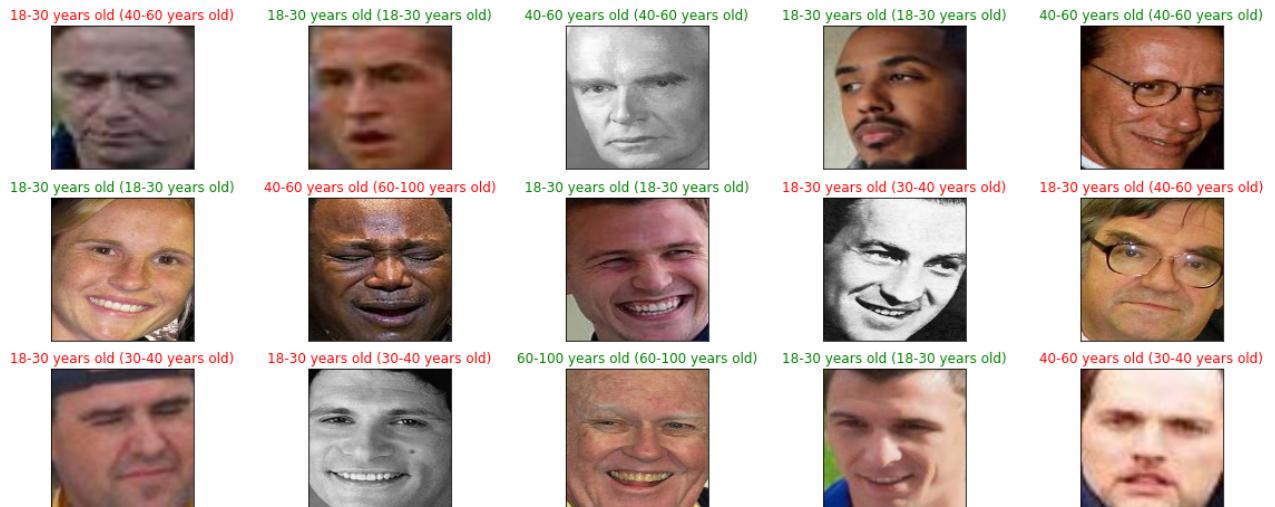
In [43]:

```
# Evaluate the model on test set  
score = model.evaluate(x_test, y_test_age, verbose=0)  
  
# Print test accuracy  
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.4478956460952759, 0.47362595796585083]
```

In [44]:

```
y_hat = model.predict(x_test)  
  
# Plot a random sample of 10 test images, their predicted labels and ground truth  
figure = plt.figure(figsize=(20, 8))  
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):  
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks=[])  
    # Display each image  
    ax.imshow(np.squeeze(x_test[index]))  
    predict_index = np.argmax(y_hat[index])  
    true_index = np.argmax(y_test_age[index])  
    # Set the title for each image  
    ax.set_title("{} ({})".format(labels[predict_index],  
                                  labels[true_index]),  
                color=("green" if predict_index == true_index else "red"))  
plt.show()
```



In []:

Build the multilayer CNN (change dropout probabilities)

3-3. Two-layer CNN (2 fully connected layers)

- Using Activation Function "ReLU"
- Using Maxing Pooling method

- kernel size = 2
- Dropout = 0.1 for convolutional layers

In [45]:

```
model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(5, activation='sigmoid'))

# Take a look at the model summary
model.summary()
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_6 (Conv2D)	(None, 128, 128, 64)	832
<hr/>		
max_pooling2d_6 (MaxPooling2D)	(None, 64, 64, 64)	0
<hr/>		
dropout_9 (Dropout)	(None, 64, 64, 64)	0
<hr/>		
conv2d_7 (Conv2D)	(None, 64, 64, 32)	8224
<hr/>		
max_pooling2d_7 (MaxPooling2D)	(None, 32, 32, 32)	0
<hr/>		
dropout_10 (Dropout)	(None, 32, 32, 32)	0
<hr/>		
flatten_3 (Flatten)	(None, 32768)	0
<hr/>		
dense_6 (Dense)	(None, 256)	8388864
<hr/>		
dropout_11 (Dropout)	(None, 256)	0
<hr/>		
dense_7 (Dense)	(None, 5)	1285
<hr/>		
Total params: 8,399,205		
Trainable params: 8,399,205		
Non-trainable params: 0		

In [46]:

```
import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

Tue Apr 27 23:46:57 2021

In [47]:

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [48]:

```
model.fit(x_train,y_train_age,
           validation_data=(x_valid , y_valid_age),
           batch_size=64,
           steps_per_epoch=20,
           epochs=100,
           validation_steps=2,
           verbose=2,
           workers=1,
           use_multiprocessing = False,
           callbacks=[earlystop, plateau])
```

```
Epoch 1/100
20/20 - 22s - loss: 0.6936 - accuracy: 0.3219 - val_loss: 0.5242 - val_accuracy: 0.4609
Epoch 2/100
20/20 - 20s - loss: 0.4820 - accuracy: 0.3805 - val_loss: 0.4884 - val_accuracy: 0.4609
Epoch 3/100
20/20 - 20s - loss: 0.4779 - accuracy: 0.3570 - val_loss: 0.4725 - val_accuracy: 0.4609
Epoch 4/100
20/20 - 20s - loss: 0.4688 - accuracy: 0.3734 - val_loss: 0.4654 - val_accuracy: 0.4609
Epoch 5/100
20/20 - 20s - loss: 0.4694 - accuracy: 0.3727 - val_loss: 0.4529 - val_accuracy: 0.4688
Epoch 6/100
20/20 - 21s - loss: 0.4619 - accuracy: 0.3930 - val_loss: 0.4515 - val_accuracy: 0.4609
Epoch 7/100
20/20 - 20s - loss: 0.4534 - accuracy: 0.3805 - val_loss: 0.4540 - val_accuracy: 0.4531
Epoch 8/100
20/20 - 20s - loss: 0.4551 - accuracy: 0.3883 - val_loss: 0.4549 - val_accuracy: 0.4844
Epoch 9/100
20/20 - 20s - loss: 0.4553 - accuracy: 0.3891 - val_loss: 0.4549 - val_accuracy: 0.4922
Epoch 10/100
20/20 - 20s - loss: 0.4451 - accuracy: 0.4219 - val_loss: 0.4608 - val_accuracy: 0.4766
Epoch 11/100
20/20 - 20s - loss: 0.4449 - accuracy: 0.4328 - val_loss: 0.4480 - val_accuracy: 0.4844
Epoch 12/100
20/20 - 19s - loss: 0.4309 - accuracy: 0.4469 - val_loss: 0.4301 - val_accuracy: 0.5000
Epoch 13/100
20/20 - 20s - loss: 0.4419 - accuracy: 0.4289 - val_loss: 0.4339 - val_accuracy: 0.5000
Epoch 14/100
20/20 - 20s - loss: 0.4393 - accuracy: 0.4359 - val_loss: 0.4231 - val_accuracy: 0.5234
Epoch 15/100
20/20 - 20s - loss: 0.4413 - accuracy: 0.4164 - val_loss: 0.4339 - val_accuracy: 0.5391
Epoch 16/100
20/20 - 20s - loss: 0.4332 - accuracy: 0.4516 - val_loss: 0.4251 - val_accuracy: 0.5078
Epoch 17/100
20/20 - 20s - loss: 0.4323 - accuracy: 0.4393 - val_loss: 0.4293 - val_accuracy: 0.5000
Epoch 18/100
20/20 - 20s - loss: 0.4219 - accuracy: 0.4773 - val_loss: 0.4093 - val_accuracy: 0.5000
Epoch 19/100
20/20 - 20s - loss: 0.4286 - accuracy: 0.4500 - val_loss: 0.4165 - val_accuracy: 0.5156
Epoch 20/100
20/20 - 21s - loss: 0.4301 - accuracy: 0.4586 - val_loss: 0.4089 - val_accuracy: 0.5156
Epoch 21/100
20/20 - 20s - loss: 0.4248 - accuracy: 0.4609 - val_loss: 0.4135 - val_accuracy: 0.5234
Epoch 22/100
20/20 - 20s - loss: 0.4298 - accuracy: 0.4523 - val_loss: 0.4088 - val_accuracy: 0.5312
Epoch 23/100
20/20 - 20s - loss: 0.4382 - accuracy: 0.4461 - val_loss: 0.4175 - val_accuracy: 0.5000

Epoch 00023: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.
Epoch 24/100
20/20 - 20s - loss: 0.4334 - accuracy: 0.4453 - val_loss: 0.4188 - val_accuracy: 0.5000
Epoch 25/100
```

```

20/20 - 19s - loss: 0.4136 - accuracy: 0.4734 - val_loss: 0.4086 - val_accuracy: 0.5078
Epoch 26/100
20/20 - 19s - loss: 0.4255 - accuracy: 0.4531 - val_loss: 0.4083 - val_accuracy: 0.5156
Epoch 27/100
20/20 - 20s - loss: 0.4159 - accuracy: 0.4711 - val_loss: 0.4051 - val_accuracy: 0.5234
Epoch 28/100
20/20 - 20s - loss: 0.4253 - accuracy: 0.4570 - val_loss: 0.4039 - val_accuracy: 0.5391
Restoring model weights from the end of the best epoch.

```

```

Epoch 00028: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 00028: early stopping

```

Out[48]: <tensorflow.python.keras.callbacks.History at 0x245ab311dc0>

In [49]:

```

# Evaluate the model on test set
score = model.evaluate(x_test, y_test_age, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)

```

```
Test accuracy: [0.4327627122402191, 0.44079676270484924]
```

In [50]:

```

y_hat = model.predict(x_test)

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test_age[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                 color=("green" if predict_index == true_index else "red"))
plt.show()

```



Build the multilayer CNN (change dropout probabilities)

3-3. Two-layer CNN (2 fully connected layers)

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- kernel size = 2
- Dropout = 0.5 for covolutional layers

In [52]:

```
model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.5))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.5))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(5, activation='sigmoid'))

# Take a Look at the model summary
model.summary()
```

Model: "sequential_5"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_10 (Conv2D)	(None, 128, 128, 64)	832
<hr/>		
max_pooling2d_10 (MaxPooling)	(None, 64, 64, 64)	0
<hr/>		
dropout_15 (Dropout)	(None, 64, 64, 64)	0
<hr/>		
conv2d_11 (Conv2D)	(None, 64, 64, 32)	8224
<hr/>		
max_pooling2d_11 (MaxPooling)	(None, 32, 32, 32)	0
<hr/>		
dropout_16 (Dropout)	(None, 32, 32, 32)	0
<hr/>		
flatten_5 (Flatten)	(None, 32768)	0
<hr/>		
dense_10 (Dense)	(None, 256)	8388864
<hr/>		
dropout_17 (Dropout)	(None, 256)	0
<hr/>		
dense_11 (Dense)	(None, 5)	1285
<hr/>		
Total params: 8,399,205		
Trainable params: 8,399,205		
Non-trainable params: 0		

In [53]:

```
import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

Tue Apr 27 23:56:37 2021

In [54]:

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

```
In [55]: model.fit(x_train,y_train_age,
                  validation_data=(x_valid , y_valid_age),
                  batch_size=64,
                  steps_per_epoch=20,
                  epochs=100,
                  validation_steps=2,
                  verbose=2,
                  workers=1,
                  use_multiprocessing = False,
                  callbacks=[earlieststop, plateau])
```

```
Epoch 1/100
20/20 - 22s - loss: 0.8818 - accuracy: 0.2742 - val_loss: 0.6762 - val_accuracy: 0.2891
Epoch 2/100
20/20 - 20s - loss: 0.4846 - accuracy: 0.3461 - val_loss: 0.6477 - val_accuracy: 0.4609
Epoch 3/100
20/20 - 21s - loss: 0.4768 - accuracy: 0.3703 - val_loss: 0.6586 - val_accuracy: 0.4609
Epoch 4/100
20/20 - 21s - loss: 0.4671 - accuracy: 0.3984 - val_loss: 0.6471 - val_accuracy: 0.4609
Epoch 5/100
20/20 - 24s - loss: 0.4576 - accuracy: 0.3805 - val_loss: 0.6401 - val_accuracy: 0.4609
Epoch 6/100
20/20 - 23s - loss: 0.4722 - accuracy: 0.3516 - val_loss: 0.6414 - val_accuracy: 0.4609
Epoch 7/100
20/20 - 21s - loss: 0.4714 - accuracy: 0.3758 - val_loss: 0.6223 - val_accuracy: 0.4609
Epoch 8/100
20/20 - 20s - loss: 0.4674 - accuracy: 0.3672 - val_loss: 0.6255 - val_accuracy: 0.4609
Epoch 9/100
20/20 - 21s - loss: 0.4649 - accuracy: 0.4008 - val_loss: 0.6149 - val_accuracy: 0.4609
Epoch 10/100
20/20 - 20s - loss: 0.4645 - accuracy: 0.3766 - val_loss: 0.6193 - val_accuracy: 0.4609
Epoch 11/100
20/20 - 21s - loss: 0.4693 - accuracy: 0.3891 - val_loss: 0.6098 - val_accuracy: 0.4688
Epoch 12/100
20/20 - 20s - loss: 0.4639 - accuracy: 0.3898 - val_loss: 0.5979 - val_accuracy: 0.4609
Epoch 13/100
20/20 - 21s - loss: 0.4592 - accuracy: 0.4227 - val_loss: 0.6024 - val_accuracy: 0.4688
Epoch 14/100
20/20 - 21s - loss: 0.4574 - accuracy: 0.4164 - val_loss: 0.5861 - val_accuracy: 0.4766
Epoch 15/100
20/20 - 21s - loss: 0.4578 - accuracy: 0.3984 - val_loss: 0.5835 - val_accuracy: 0.5000
Epoch 16/100
20/20 - 21s - loss: 0.4525 - accuracy: 0.4031 - val_loss: 0.5669 - val_accuracy: 0.4844
Epoch 17/100
20/20 - 20s - loss: 0.4537 - accuracy: 0.4054 - val_loss: 0.5412 - val_accuracy: 0.4844
Epoch 18/100
20/20 - 20s - loss: 0.4490 - accuracy: 0.4102 - val_loss: 0.5510 - val_accuracy: 0.5000

Epoch 00018: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.
Epoch 19/100
20/20 - 20s - loss: 0.4556 - accuracy: 0.3945 - val_loss: 0.5480 - val_accuracy: 0.4844
Epoch 20/100
20/20 - 20s - loss: 0.4453 - accuracy: 0.4219 - val_loss: 0.5413 - val_accuracy: 0.4922
Epoch 21/100
20/20 - 21s - loss: 0.4506 - accuracy: 0.4141 - val_loss: 0.5444 - val_accuracy: 0.4766
Epoch 22/100
20/20 - 21s - loss: 0.4514 - accuracy: 0.4187 - val_loss: 0.5412 - val_accuracy: 0.4844
Epoch 23/100
```

```
20/20 - 21s - loss: 0.4483 - accuracy: 0.4062 - val_loss: 0.5382 - val_accuracy: 0.4844  
Restoring model weights from the end of the best epoch.
```

```
Epoch 00023: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.  
Epoch 00023: early stopping
```

```
Out[55]: <tensorflow.python.keras.callbacks.History at 0x245b35faca0>
```

```
In [56]:
```

```
# Evaluate the model on test set  
score = model.evaluate(x_test, y_test_age, verbose=0)  
  
# Print test accuracy  
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.6159905791282654, 0.39431944489479065]
```

```
In [57]:
```

```
y_hat = model.predict(x_test)  
  
# Plot a random sample of 10 test images, their predicted labels and ground truth  
figure = plt.figure(figsize=(20, 8))  
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):  
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks=[])  
    # Display each image  
    ax.imshow(np.squeeze(x_test[index]))  
    predict_index = np.argmax(y_hat[index])  
    true_index = np.argmax(y_test_age[index])  
    # Set the title for each image  
    ax.set_title("{} ({})".format(labels[predict_index],  
                                   labels[true_index]),  
                color=("green" if predict_index == true_index else "red"))  
plt.show()
```



Conclusion for the previous models

- Kernel size = 3
- Dropout rate = 0.2
- #Fully connected layers do not affect much, so still use two layers: 256, 5.
- For other fixed info:
 - Activation function: ReLU,

- Pooling Methods: Max Pooling,
- #Convolution layers = 2

In [67]:

```
model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.2))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.2))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(5, activation='sigmoid'))

# Take a look at the model summary
model.summary()
```

Model: "sequential_7"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_14 (Conv2D)	(None, 128, 128, 64)	1792
<hr/>		
max_pooling2d_14 (MaxPooling)	(None, 64, 64, 64)	0
<hr/>		
dropout_21 (Dropout)	(None, 64, 64, 64)	0
<hr/>		
conv2d_15 (Conv2D)	(None, 64, 64, 32)	18464
<hr/>		
max_pooling2d_15 (MaxPooling)	(None, 32, 32, 32)	0
<hr/>		
dropout_22 (Dropout)	(None, 32, 32, 32)	0
<hr/>		
flatten_7 (Flatten)	(None, 32768)	0
<hr/>		
dense_14 (Dense)	(None, 256)	8388864
<hr/>		
dropout_23 (Dropout)	(None, 256)	0
<hr/>		
dense_15 (Dense)	(None, 5)	1285
<hr/>		
Total params: 8,410,405		
Trainable params: 8,410,405		
Non-trainable params: 0		

In [68]:

```
import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

Wed Apr 28 10:08:55 2021

In [69]:

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

```
In [ ]: model.fit(x_train,
                 y_train_age,
                 batch_size=64,
                 epochs=20,
                 validation_data=(x_valid, y_valid_age),)

Epoch 1/20
339/339 [=====] - 619s 2s/step - loss: 0.4946 - accuracy: 0.374
7 - val_loss: 0.4381 - val_accuracy: 0.4469
Epoch 2/20
339/339 [=====] - 612s 2s/step - loss: 0.4348 - accuracy: 0.435
0 - val_loss: 0.4067 - val_accuracy: 0.4963
Epoch 3/20
44/339 [==>.....] - ETA: 5:53 - loss: 0.4231 - accuracy: 0.4575
```

```
In [ ]: # Evaluate the model on test set
score = model.evaluate(x_test, y_test_age, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
In [ ]: y_hat = model.predict(x_test)

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test_age[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                color=("green" if predict_index == true_index else "red"))
plt.show()
```

```
In [ ]:
```

```
In [61]: model.fit(x_train,y_train_age,
                 validation_data=(x_valid , y_valid_age),
                 batch_size=64,
                 steps_per_epoch=20,
                 epochs=100,
                 validation_steps=2,
                 verbose=2,
                 workers=1,
                 use_multiprocessing = False,
                 callbacks=[earlystop, plateau])
```

```
Epoch 1/100
20/20 - 28s - loss: 0.6542 - accuracy: 0.2594 - val_loss: 0.4918 - val_accuracy: 0.4609
Epoch 2/100
20/20 - 26s - loss: 0.4789 - accuracy: 0.3711 - val_loss: 0.5039 - val_accuracy: 0.4609
Epoch 3/100
```

20/20 - 26s - loss: 0.4723 - accuracy: 0.3859 - val_loss: 0.4973 - val_accuracy: 0.4609
Epoch 4/100
20/20 - 28s - loss: 0.4701 - accuracy: 0.3664 - val_loss: 0.4937 - val_accuracy: 0.4531
Epoch 5/100
20/20 - 25s - loss: 0.4611 - accuracy: 0.3945 - val_loss: 0.5239 - val_accuracy: 0.4609
Epoch 6/100
20/20 - 28s - loss: 0.4658 - accuracy: 0.3789 - val_loss: 0.5070 - val_accuracy: 0.4531
Epoch 7/100
20/20 - 27s - loss: 0.4564 - accuracy: 0.4039 - val_loss: 0.5262 - val_accuracy: 0.5000
Epoch 8/100
20/20 - 26s - loss: 0.4566 - accuracy: 0.3961 - val_loss: 0.4680 - val_accuracy: 0.4531
Epoch 9/100
20/20 - 27s - loss: 0.4556 - accuracy: 0.3867 - val_loss: 0.4781 - val_accuracy: 0.4922
Epoch 10/100
20/20 - 25s - loss: 0.4486 - accuracy: 0.3984 - val_loss: 0.4626 - val_accuracy: 0.4922
Epoch 11/100
20/20 - 26s - loss: 0.4516 - accuracy: 0.4047 - val_loss: 0.4692 - val_accuracy: 0.4922
Epoch 12/100
20/20 - 19s - loss: 0.4531 - accuracy: 0.3969 - val_loss: 0.4719 - val_accuracy: 0.4844
Epoch 13/100
20/20 - 22s - loss: 0.4487 - accuracy: 0.4219 - val_loss: 0.4552 - val_accuracy: 0.4844
Epoch 14/100
20/20 - 26s - loss: 0.4478 - accuracy: 0.4117 - val_loss: 0.4599 - val_accuracy: 0.5000
Epoch 15/100
20/20 - 26s - loss: 0.4451 - accuracy: 0.4219 - val_loss: 0.4510 - val_accuracy: 0.4922
Epoch 16/100
20/20 - 27s - loss: 0.4483 - accuracy: 0.3969 - val_loss: 0.4360 - val_accuracy: 0.4844
Epoch 17/100
20/20 - 27s - loss: 0.4371 - accuracy: 0.4314 - val_loss: 0.4341 - val_accuracy: 0.4922
Epoch 18/100
20/20 - 26s - loss: 0.4359 - accuracy: 0.4461 - val_loss: 0.4232 - val_accuracy: 0.4531
Epoch 19/100
20/20 - 27s - loss: 0.4331 - accuracy: 0.4445 - val_loss: 0.4198 - val_accuracy: 0.4844
Epoch 20/100
20/20 - 27s - loss: 0.4326 - accuracy: 0.4352 - val_loss: 0.4177 - val_accuracy: 0.5000
Epoch 21/100
20/20 - 27s - loss: 0.4282 - accuracy: 0.4437 - val_loss: 0.4276 - val_accuracy: 0.5234
Epoch 22/100
20/20 - 27s - loss: 0.4282 - accuracy: 0.4523 - val_loss: 0.4041 - val_accuracy: 0.5156
Epoch 23/100
20/20 - 26s - loss: 0.4403 - accuracy: 0.4289 - val_loss: 0.4127 - val_accuracy: 0.5156
Epoch 24/100
20/20 - 25s - loss: 0.4358 - accuracy: 0.4266 - val_loss: 0.4075 - val_accuracy: 0.5078
Epoch 25/100
20/20 - 25s - loss: 0.4357 - accuracy: 0.4305 - val_loss: 0.4094 - val_accuracy: 0.5078
Epoch 26/100
20/20 - 26s - loss: 0.4318 - accuracy: 0.4539 - val_loss: 0.4348 - val_accuracy: 0.5078
Epoch 27/100
20/20 - 27s - loss: 0.4224 - accuracy: 0.4547 - val_loss: 0.3985 - val_accuracy: 0.5312
Epoch 28/100
20/20 - 26s - loss: 0.4355 - accuracy: 0.4578 - val_loss: 0.4103 - val_accuracy: 0.5391
Epoch 29/100
20/20 - 26s - loss: 0.4297 - accuracy: 0.4461 - val_loss: 0.3947 - val_accuracy: 0.5156
Epoch 30/100
20/20 - 27s - loss: 0.4252 - accuracy: 0.4430 - val_loss: 0.3997 - val_accuracy: 0.5312
Epoch 31/100
20/20 - 28s - loss: 0.4240 - accuracy: 0.4711 - val_loss: 0.3916 - val_accuracy: 0.5547
Epoch 32/100
20/20 - 27s - loss: 0.4280 - accuracy: 0.4469 - val_loss: 0.4014 - val_accuracy: 0.5391
Epoch 33/100
20/20 - 26s - loss: 0.4210 - accuracy: 0.4633 - val_loss: 0.3790 - val_accuracy: 0.5703
Epoch 34/100
20/20 - 27s - loss: 0.4147 - accuracy: 0.4787 - val_loss: 0.3868 - val_accuracy: 0.5469
Epoch 35/100
20/20 - 26s - loss: 0.4202 - accuracy: 0.4664 - val_loss: 0.3794 - val_accuracy: 0.5625

```
Epoch 36/100
20/20 - 26s - loss: 0.4257 - accuracy: 0.4414 - val_loss: 0.3806 - val_accuracy: 0.5625
Epoch 37/100
20/20 - 26s - loss: 0.4199 - accuracy: 0.4594 - val_loss: 0.3737 - val_accuracy: 0.5469
Epoch 38/100
20/20 - 25s - loss: 0.4194 - accuracy: 0.4734 - val_loss: 0.3637 - val_accuracy: 0.6016
Epoch 39/100
20/20 - 26s - loss: 0.4190 - accuracy: 0.4688 - val_loss: 0.3762 - val_accuracy: 0.5703

Epoch 00039: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.
Epoch 40/100
20/20 - 26s - loss: 0.4228 - accuracy: 0.4586 - val_loss: 0.3774 - val_accuracy: 0.5547
Epoch 41/100
20/20 - 27s - loss: 0.4126 - accuracy: 0.4875 - val_loss: 0.3761 - val_accuracy: 0.5781
Epoch 42/100
20/20 - 27s - loss: 0.4131 - accuracy: 0.4953 - val_loss: 0.3734 - val_accuracy: 0.5625
Epoch 43/100
20/20 - 27s - loss: 0.4138 - accuracy: 0.4695 - val_loss: 0.3718 - val_accuracy: 0.6016
Epoch 44/100
20/20 - 27s - loss: 0.4113 - accuracy: 0.4859 - val_loss: 0.3701 - val_accuracy: 0.5781
Epoch 45/100
20/20 - 28s - loss: 0.4197 - accuracy: 0.4664 - val_loss: 0.3732 - val_accuracy: 0.5859
Epoch 46/100
20/20 - 28s - loss: 0.4069 - accuracy: 0.4930 - val_loss: 0.3675 - val_accuracy: 0.5703
Epoch 47/100
20/20 - 28s - loss: 0.4155 - accuracy: 0.4844 - val_loss: 0.3722 - val_accuracy: 0.6016

Epoch 00047: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 48/100
20/20 - 28s - loss: 0.4107 - accuracy: 0.4789 - val_loss: 0.3716 - val_accuracy: 0.5938
Epoch 49/100
20/20 - 31s - loss: 0.4185 - accuracy: 0.4781 - val_loss: 0.3706 - val_accuracy: 0.5938
Epoch 50/100
20/20 - 39s - loss: 0.4012 - accuracy: 0.4930 - val_loss: 0.3700 - val_accuracy: 0.5938
Epoch 51/100
20/20 - 64s - loss: 0.4163 - accuracy: 0.4440 - val_loss: 0.3697 - val_accuracy: 0.6094
Epoch 52/100
20/20 - 78s - loss: 0.4130 - accuracy: 0.4852 - val_loss: 0.3697 - val_accuracy: 0.6094
Restoring model weights from the end of the best epoch.

Epoch 00052: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
Epoch 00052: early stopping
```

Out[61]: <tensorflow.python.keras.callbacks.History at 0x245aba66af0>

```
In [62]: # Evaluate the model on test set
score = model.evaluate(x_test, y_test_age, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

Test accuracy: [0.40654438734054565, 0.4957580268383026]

```
In [65]: y_hat = model.predict(x_test)

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
```

```
        true_index = np.argmax(y_test_age[index])
        # Set the title for each image
        ax.set_title("{} ({})".format(labels[predict_index],
                                      labels[true_index]),
                                      color=("green" if predict_index == true_index else "r"))
plt.show()
```



In []:

CNN- Predict Gender

In [1]:

```
import numpy as np # Linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import math
import cv2
import matplotlib.pyplot as plt
import os
import seaborn as sns
!pip install umap
import umap
from PIL import Image
from scipy import misc
from os import listdir
from os.path import isfile, join
import numpy as np
from scipy import misc
from random import shuffle
from collections import Counter
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.utils import to_categorical
```

Requirement already satisfied: umap in c:\miniconda3\envs\d2l\lib\site-packages (0.1.1)

In [2]:

```
data_csv = pd.read_csv('wikifinal.csv')
data_csv = data_csv[data_csv['AGE'] <= 100].reset_index(drop=True)
```

In [3]:

```
mypath = os.chdir('../crop')
onlyfiles = os.listdir()
```

In [4]:

```
import imageio
```

In [5]:

```
image_name = []
for name in data_csv["PATH"]:
    image_name.append(name)
```

In [6]:

```
# This way is faster than the original one
def convertImage(filename):
    face = imageio.imread(filename)
    face = cv2.resize(face, (128, 128))
    return face

Image_data = list(map(convertImage, image_name))
```

In [7]:

```
len(Image_data)
```

```
Out[7]: 27105
```

```
image_data =[] for i in image_name: face = imageio.imread(os.path.join("./crop/" + i)) face = cv2.resize(face, (128, 128)) image_data.append(face)
```

```
In [8]:  
Image_data = np.squeeze(Image_data)  
Image_data.shape
```

```
Out[8]: (27105, 128, 128, 3)
```

```
In [9]:  
# normalize data  
Image_data = Image_data.astype('float32')  
Image_data /= 255
```

```
In [10]:  
age, gender = list(), list()  
for i in range(len(data_csv)):  
    age.append(data_csv["AGE"][i])  
    gender.append(data_csv["GENDER"][i])
```

```
In [11]: gender_labels = to_categorical(gender, num_classes=2)
```

```
In [12]: gender_labels[:10]
```

```
Out[12]: array([[0., 1.],  
                 [0., 1.],  
                 [1., 0.],  
                 [0., 1.],  
                 [0., 1.],  
                 [0., 1.],  
                 [1., 0.],  
                 [0., 1.],  
                 [0., 1.],  
                 [1., 0.]], dtype=float32)
```

```
In [13]:  
print("Using 0- 80% data as training data: " + str(int(len(Image_data)*0.8)))  
print("Using 80%-90% data as training data: " + str(int(len(Image_data)*0.9)))  
print("Number of data in validation set: " + str(24394-21684))
```

```
Using 0- 80% data as training data: 21684  
Using 80%-90% data as training data: 24394  
Number of data in validation set: 2710
```

```
In [14]:  
(x_train, y_train_gender), (x_test, y_test_gender) = (Image_data[:21684], gender_labels)  
(x_valid, y_valid_gender) = (x_test[:2710], y_test_gender[:2710])  
(x_test, y_test_gender) = (x_test[2710:], y_test_gender[2710:])
```

```
In [15]: len(x_train) + len(x_test) + len(x_valid) == len(Image_data)
```

```
Out[15]: True
```

Define functions

```
In [16]: from keras.callbacks import LearningRateScheduler, EarlyStopping, Callback, ReduceLROnPlateau

earlystop = EarlyStopping(monitor='accuracy',
                         min_delta=0,
                         patience=10,
                         verbose=2,
                         mode='max',
                         baseline=None,
                         restore_best_weights=True)

# Plateau
plateau = ReduceLROnPlateau(monitor='accuracy',
                            factor=0.1,
                            patience=5,
                            verbose=2,
                            mode='max',
                            min_delta=0.0001,
                            cooldown=0,
                            min_lr=0)
```

Build the multilayer CNN (change Kernel Size)

1-1(2-1). Two-layer CNN

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- kernel size = 2
- Dropout = 0.3

```
In [17]: model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(2, activation='sigmoid'))

# Take a Look at the model summary
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d (Conv2D)	(None, 128, 128, 64)	832

max_pooling2d	(MaxPooling2D)	(None, 64, 64, 64)	0
dropout	(Dropout)	(None, 64, 64, 64)	0
conv2d_1	(Conv2D)	(None, 64, 64, 32)	8224
max_pooling2d_1	(MaxPooling2D)	(None, 32, 32, 32)	0
dropout_1	(Dropout)	(None, 32, 32, 32)	0
flatten	(Flatten)	(None, 32768)	0
dense	(Dense)	(None, 256)	8388864
dropout_2	(Dropout)	(None, 256)	0
dense_1	(Dense)	(None, 2)	514
<hr/>			
Total params: 8,398,434			
Trainable params: 8,398,434			
Non-trainable params: 0			

In [18]:

```
import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

Tue Apr 27 17:21:28 2021

In [19]:

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [20]:

```
model.fit(x_train,y_train_gender,
           validation_data=(x_valid, y_valid_gender),
           batch_size=64,
           steps_per_epoch=20,
           epochs=100,
           validation_steps=2,
           verbose=2,
           workers=1,
           use_multiprocessing = False,
           callbacks=[earlystop, plateau])
```

```
Epoch 1/100
20/20 - 24s - loss: 1.1781 - accuracy: 0.6687 - val_loss: 0.6695 - val_accuracy: 0.8047
Epoch 2/100
20/20 - 21s - loss: 0.6229 - accuracy: 0.6891 - val_loss: 0.6699 - val_accuracy: 0.8047
Epoch 3/100
20/20 - 21s - loss: 0.5911 - accuracy: 0.7125 - val_loss: 0.6188 - val_accuracy: 0.8047
Epoch 4/100
20/20 - 22s - loss: 0.5927 - accuracy: 0.7063 - val_loss: 0.5657 - val_accuracy: 0.8047
Epoch 5/100
20/20 - 21s - loss: 0.5456 - accuracy: 0.7430 - val_loss: 0.5400 - val_accuracy: 0.8047
Epoch 6/100
20/20 - 21s - loss: 0.5395 - accuracy: 0.7570 - val_loss: 0.5458 - val_accuracy: 0.8203
Epoch 7/100
20/20 - 22s - loss: 0.5198 - accuracy: 0.7602 - val_loss: 0.5407 - val_accuracy: 0.8125
Epoch 8/100
20/20 - 21s - loss: 0.5325 - accuracy: 0.7422 - val_loss: 0.5100 - val_accuracy: 0.8047
```

Epoch 9/100
20/20 - 20s - loss: 0.5386 - accuracy: 0.7398 - val_loss: 0.5552 - val_accuracy: 0.8203
Epoch 10/100
20/20 - 20s - loss: 0.4961 - accuracy: 0.7797 - val_loss: 0.4993 - val_accuracy: 0.8047
Epoch 11/100
20/20 - 20s - loss: 0.5005 - accuracy: 0.7680 - val_loss: 0.5232 - val_accuracy: 0.7891
Epoch 12/100
20/20 - 20s - loss: 0.4587 - accuracy: 0.7930 - val_loss: 0.4597 - val_accuracy: 0.8281
Epoch 13/100
20/20 - 20s - loss: 0.4826 - accuracy: 0.7852 - val_loss: 0.4674 - val_accuracy: 0.8281
Epoch 14/100
20/20 - 20s - loss: 0.4500 - accuracy: 0.8000 - val_loss: 0.4663 - val_accuracy: 0.8047
Epoch 15/100
20/20 - 21s - loss: 0.4510 - accuracy: 0.7906 - val_loss: 0.4731 - val_accuracy: 0.8125
Epoch 16/100
20/20 - 21s - loss: 0.4511 - accuracy: 0.7937 - val_loss: 0.4635 - val_accuracy: 0.8203
Epoch 17/100
20/20 - 20s - loss: 0.4287 - accuracy: 0.8218 - val_loss: 0.4176 - val_accuracy: 0.7891
Epoch 18/100
20/20 - 20s - loss: 0.4044 - accuracy: 0.8211 - val_loss: 0.4371 - val_accuracy: 0.8203
Epoch 19/100
20/20 - 20s - loss: 0.4154 - accuracy: 0.8094 - val_loss: 0.4275 - val_accuracy: 0.8281
Epoch 20/100
20/20 - 20s - loss: 0.4218 - accuracy: 0.8266 - val_loss: 0.4402 - val_accuracy: 0.8281
Epoch 21/100
20/20 - 19s - loss: 0.3898 - accuracy: 0.8352 - val_loss: 0.4263 - val_accuracy: 0.8281
Epoch 22/100
20/20 - 20s - loss: 0.3742 - accuracy: 0.8406 - val_loss: 0.4082 - val_accuracy: 0.8281
Epoch 23/100
20/20 - 20s - loss: 0.4195 - accuracy: 0.8039 - val_loss: 0.4340 - val_accuracy: 0.8359
Epoch 24/100
20/20 - 20s - loss: 0.3622 - accuracy: 0.8477 - val_loss: 0.4079 - val_accuracy: 0.8125
Epoch 25/100
20/20 - 20s - loss: 0.4160 - accuracy: 0.8211 - val_loss: 0.3890 - val_accuracy: 0.8203
Epoch 26/100
20/20 - 20s - loss: 0.3654 - accuracy: 0.8570 - val_loss: 0.4082 - val_accuracy: 0.8516
Epoch 27/100
20/20 - 20s - loss: 0.3774 - accuracy: 0.8359 - val_loss: 0.4033 - val_accuracy: 0.8516
Epoch 28/100
20/20 - 20s - loss: 0.4047 - accuracy: 0.8250 - val_loss: 0.4103 - val_accuracy: 0.8281
Epoch 29/100
20/20 - 20s - loss: 0.3631 - accuracy: 0.8586 - val_loss: 0.4135 - val_accuracy: 0.8359
Epoch 30/100
20/20 - 20s - loss: 0.3827 - accuracy: 0.8422 - val_loss: 0.3941 - val_accuracy: 0.8516
Epoch 31/100
20/20 - 20s - loss: 0.3556 - accuracy: 0.8453 - val_loss: 0.3952 - val_accuracy: 0.8516
Epoch 32/100
20/20 - 20s - loss: 0.3576 - accuracy: 0.8477 - val_loss: 0.3764 - val_accuracy: 0.8594
Epoch 33/100
20/20 - 20s - loss: 0.3928 - accuracy: 0.8383 - val_loss: 0.3994 - val_accuracy: 0.8359
Epoch 34/100
20/20 - 20s - loss: 0.3821 - accuracy: 0.8423 - val_loss: 0.3770 - val_accuracy: 0.8438

Epoch 00034: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.

Epoch 35/100
20/20 - 20s - loss: 0.3596 - accuracy: 0.8438 - val_loss: 0.3700 - val_accuracy: 0.8516
Epoch 36/100
20/20 - 20s - loss: 0.3664 - accuracy: 0.8469 - val_loss: 0.3763 - val_accuracy: 0.8359
Epoch 37/100
20/20 - 20s - loss: 0.3381 - accuracy: 0.8617 - val_loss: 0.3715 - val_accuracy: 0.8203
Epoch 38/100
20/20 - 20s - loss: 0.3651 - accuracy: 0.8469 - val_loss: 0.3730 - val_accuracy: 0.8359
Epoch 39/100
20/20 - 20s - loss: 0.3396 - accuracy: 0.8672 - val_loss: 0.3726 - val_accuracy: 0.8438
Epoch 40/100

```
20/20 - 20s - loss: 0.3376 - accuracy: 0.8656 - val_loss: 0.3718 - val_accuracy: 0.8359
Epoch 41/100
20/20 - 21s - loss: 0.3312 - accuracy: 0.8594 - val_loss: 0.3644 - val_accuracy: 0.8359
Epoch 42/100
20/20 - 20s - loss: 0.3186 - accuracy: 0.8680 - val_loss: 0.3610 - val_accuracy: 0.8359
Epoch 43/100
20/20 - 21s - loss: 0.3014 - accuracy: 0.8664 - val_loss: 0.3616 - val_accuracy: 0.8359
Epoch 44/100
20/20 - 20s - loss: 0.3146 - accuracy: 0.8648 - val_loss: 0.3658 - val_accuracy: 0.8516
Epoch 45/100
20/20 - 22s - loss: 0.3140 - accuracy: 0.8734 - val_loss: 0.3635 - val_accuracy: 0.8438
Epoch 46/100
20/20 - 22s - loss: 0.3208 - accuracy: 0.8766 - val_loss: 0.3637 - val_accuracy: 0.8359
Epoch 47/100
20/20 - 21s - loss: 0.3237 - accuracy: 0.8680 - val_loss: 0.3707 - val_accuracy: 0.8516
Epoch 48/100
20/20 - 21s - loss: 0.3189 - accuracy: 0.8648 - val_loss: 0.3689 - val_accuracy: 0.8438
Epoch 49/100
20/20 - 21s - loss: 0.3328 - accuracy: 0.8547 - val_loss: 0.3682 - val_accuracy: 0.8359
Epoch 50/100
20/20 - 20s - loss: 0.3203 - accuracy: 0.8648 - val_loss: 0.3707 - val_accuracy: 0.8438
Epoch 51/100
20/20 - 22s - loss: 0.3271 - accuracy: 0.8667 - val_loss: 0.3699 - val_accuracy: 0.8359

Epoch 00051: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 52/100
20/20 - 23s - loss: 0.3348 - accuracy: 0.8695 - val_loss: 0.3698 - val_accuracy: 0.8359
Epoch 53/100
20/20 - 22s - loss: 0.3348 - accuracy: 0.8578 - val_loss: 0.3703 - val_accuracy: 0.8359
Epoch 54/100
20/20 - 22s - loss: 0.3275 - accuracy: 0.8633 - val_loss: 0.3705 - val_accuracy: 0.8359
Epoch 55/100
20/20 - 22s - loss: 0.3301 - accuracy: 0.8656 - val_loss: 0.3708 - val_accuracy: 0.8359
Epoch 56/100
20/20 - 23s - loss: 0.3175 - accuracy: 0.8711 - val_loss: 0.3708 - val_accuracy: 0.8359
Restoring model weights from the end of the best epoch.
```

```
Epoch 00056: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
Epoch 00056: early stopping
```

```
Out[20]: <tensorflow.python.keras.callbacks.History at 0x2342698ad60>
```

```
In [21]: # Evaluate the model on test set
score = model.evaluate(x_test, y_test_gender, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.3422989845275879, 0.8561416268348694]
```

Another try

```
In [18]: import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

```
Tue Apr 27 20:42:09 2021
```

```
In [19]: model.compile(loss='binary_crossentropy',
                    optimizer='adam',
                    metrics=['accuracy'])
```

In [20]:

```
model.fit(x_train,y_train_gender,
           validation_data=(x_valid, y_valid_gender),
           batch_size=64,
           steps_per_epoch=20,
           epochs=100,
           validation_steps=2,
           verbose=2,
           workers=1,
           use_multiprocessing = False,
           callbacks=[earlystop, plateau])
```

```
Epoch 1/100
20/20 - 22s - loss: 1.1405 - accuracy: 0.6641 - val_loss: 0.6884 - val_accuracy: 0.8047
Epoch 2/100
20/20 - 20s - loss: 0.6214 - accuracy: 0.7094 - val_loss: 0.6668 - val_accuracy: 0.8047
Epoch 3/100
20/20 - 20s - loss: 0.6103 - accuracy: 0.7023 - val_loss: 0.6244 - val_accuracy: 0.8047
Epoch 4/100
20/20 - 20s - loss: 0.5940 - accuracy: 0.7047 - val_loss: 0.5861 - val_accuracy: 0.8047
Epoch 5/100
20/20 - 19s - loss: 0.5665 - accuracy: 0.7234 - val_loss: 0.5493 - val_accuracy: 0.8047
Epoch 6/100
20/20 - 20s - loss: 0.5644 - accuracy: 0.7156 - val_loss: 0.5760 - val_accuracy: 0.7656
Epoch 7/100
20/20 - 20s - loss: 0.5645 - accuracy: 0.7133 - val_loss: 0.4975 - val_accuracy: 0.8047
Epoch 8/100
20/20 - 20s - loss: 0.5587 - accuracy: 0.7156 - val_loss: 0.5092 - val_accuracy: 0.8047
Epoch 9/100
20/20 - 20s - loss: 0.5303 - accuracy: 0.7430 - val_loss: 0.4740 - val_accuracy: 0.7969
Epoch 10/100
20/20 - 20s - loss: 0.5180 - accuracy: 0.7547 - val_loss: 0.4875 - val_accuracy: 0.8125
Epoch 11/100
20/20 - 20s - loss: 0.4952 - accuracy: 0.7547 - val_loss: 0.4486 - val_accuracy: 0.8125
Epoch 12/100
20/20 - 21s - loss: 0.4965 - accuracy: 0.7609 - val_loss: 0.4604 - val_accuracy: 0.8125
Epoch 13/100
20/20 - 21s - loss: 0.4968 - accuracy: 0.7602 - val_loss: 0.4632 - val_accuracy: 0.8125
Epoch 14/100
20/20 - 21s - loss: 0.4668 - accuracy: 0.7930 - val_loss: 0.4138 - val_accuracy: 0.8125
Epoch 15/100
20/20 - 20s - loss: 0.4386 - accuracy: 0.8031 - val_loss: 0.4302 - val_accuracy: 0.8047
Epoch 16/100
20/20 - 20s - loss: 0.4685 - accuracy: 0.7977 - val_loss: 0.4235 - val_accuracy: 0.8125
Epoch 17/100
20/20 - 21s - loss: 0.4666 - accuracy: 0.8005 - val_loss: 0.4448 - val_accuracy: 0.8125
Epoch 18/100
20/20 - 21s - loss: 0.4512 - accuracy: 0.7937 - val_loss: 0.4293 - val_accuracy: 0.8203
Epoch 19/100
20/20 - 21s - loss: 0.4231 - accuracy: 0.8117 - val_loss: 0.4190 - val_accuracy: 0.8203
Epoch 20/100
20/20 - 20s - loss: 0.4404 - accuracy: 0.8023 - val_loss: 0.4267 - val_accuracy: 0.8125
Epoch 21/100
20/20 - 20s - loss: 0.4368 - accuracy: 0.8117 - val_loss: 0.3975 - val_accuracy: 0.7734
Epoch 22/100
20/20 - 20s - loss: 0.4048 - accuracy: 0.8273 - val_loss: 0.4488 - val_accuracy: 0.8047
Epoch 23/100
20/20 - 20s - loss: 0.4142 - accuracy: 0.8242 - val_loss: 0.4068 - val_accuracy: 0.8203
Epoch 24/100
20/20 - 20s - loss: 0.4025 - accuracy: 0.8289 - val_loss: 0.3976 - val_accuracy: 0.8281
Epoch 25/100
20/20 - 20s - loss: 0.4096 - accuracy: 0.8242 - val_loss: 0.3865 - val_accuracy: 0.8516
Epoch 26/100
20/20 - 20s - loss: 0.4090 - accuracy: 0.8305 - val_loss: 0.3921 - val_accuracy: 0.8281
```

Epoch 27/100
20/20 - 20s - loss: 0.3947 - accuracy: 0.8281 - val_loss: 0.3793 - val_accuracy: 0.8438
Epoch 28/100
20/20 - 20s - loss: 0.3974 - accuracy: 0.8250 - val_loss: 0.3958 - val_accuracy: 0.8203
Epoch 29/100
20/20 - 20s - loss: 0.4000 - accuracy: 0.8313 - val_loss: 0.4145 - val_accuracy: 0.8203
Epoch 30/100
20/20 - 19s - loss: 0.4119 - accuracy: 0.8195 - val_loss: 0.4078 - val_accuracy: 0.8203
Epoch 31/100
20/20 - 20s - loss: 0.3741 - accuracy: 0.8367 - val_loss: 0.4097 - val_accuracy: 0.8203
Epoch 32/100
20/20 - 19s - loss: 0.3753 - accuracy: 0.8414 - val_loss: 0.4031 - val_accuracy: 0.8281
Epoch 33/100
20/20 - 20s - loss: 0.3963 - accuracy: 0.8281 - val_loss: 0.3811 - val_accuracy: 0.8281
Epoch 34/100
20/20 - 19s - loss: 0.4087 - accuracy: 0.8257 - val_loss: 0.3986 - val_accuracy: 0.8281
Epoch 35/100
20/20 - 20s - loss: 0.3814 - accuracy: 0.8336 - val_loss: 0.3705 - val_accuracy: 0.8438
Epoch 36/100
20/20 - 20s - loss: 0.3579 - accuracy: 0.8523 - val_loss: 0.4014 - val_accuracy: 0.8516
Epoch 37/100
20/20 - 20s - loss: 0.3834 - accuracy: 0.8430 - val_loss: 0.4169 - val_accuracy: 0.8047
Epoch 38/100
20/20 - 20s - loss: 0.3621 - accuracy: 0.8516 - val_loss: 0.3799 - val_accuracy: 0.8359
Epoch 39/100
20/20 - 20s - loss: 0.3893 - accuracy: 0.8297 - val_loss: 0.4399 - val_accuracy: 0.8359
Epoch 40/100
20/20 - 20s - loss: 0.3558 - accuracy: 0.8406 - val_loss: 0.3831 - val_accuracy: 0.8359
Epoch 41/100
20/20 - 20s - loss: 0.3618 - accuracy: 0.8453 - val_loss: 0.3836 - val_accuracy: 0.8438

Epoch 00041: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.

Epoch 42/100
20/20 - 20s - loss: 0.3605 - accuracy: 0.8383 - val_loss: 0.3791 - val_accuracy: 0.8281
Epoch 43/100
20/20 - 20s - loss: 0.3587 - accuracy: 0.8508 - val_loss: 0.3829 - val_accuracy: 0.8203
Epoch 44/100
20/20 - 20s - loss: 0.3320 - accuracy: 0.8633 - val_loss: 0.3779 - val_accuracy: 0.8516
Epoch 45/100
20/20 - 20s - loss: 0.3597 - accuracy: 0.8570 - val_loss: 0.3804 - val_accuracy: 0.8359
Epoch 46/100
20/20 - 20s - loss: 0.3372 - accuracy: 0.8641 - val_loss: 0.3823 - val_accuracy: 0.8516
Epoch 47/100
20/20 - 20s - loss: 0.3408 - accuracy: 0.8687 - val_loss: 0.3814 - val_accuracy: 0.8438
Epoch 48/100
20/20 - 20s - loss: 0.3407 - accuracy: 0.8594 - val_loss: 0.3803 - val_accuracy: 0.8438
Epoch 49/100
20/20 - 20s - loss: 0.3589 - accuracy: 0.8562 - val_loss: 0.3821 - val_accuracy: 0.8438
Epoch 50/100
20/20 - 20s - loss: 0.3207 - accuracy: 0.8703 - val_loss: 0.3795 - val_accuracy: 0.8359
Epoch 51/100
20/20 - 20s - loss: 0.3115 - accuracy: 0.8778 - val_loss: 0.3804 - val_accuracy: 0.8438
Epoch 52/100
20/20 - 20s - loss: 0.3104 - accuracy: 0.8797 - val_loss: 0.3822 - val_accuracy: 0.8438
Epoch 53/100
20/20 - 20s - loss: 0.3149 - accuracy: 0.8633 - val_loss: 0.3861 - val_accuracy: 0.8438
Epoch 54/100
20/20 - 20s - loss: 0.3327 - accuracy: 0.8445 - val_loss: 0.3872 - val_accuracy: 0.8438
Epoch 55/100
20/20 - 20s - loss: 0.3221 - accuracy: 0.8586 - val_loss: 0.3836 - val_accuracy: 0.8359
Epoch 56/100
20/20 - 19s - loss: 0.3566 - accuracy: 0.8500 - val_loss: 0.3817 - val_accuracy: 0.8438
Epoch 57/100
20/20 - 20s - loss: 0.3260 - accuracy: 0.8648 - val_loss: 0.3861 - val_accuracy: 0.8359

```
Epoch 00057: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 58/100
20/20 - 20s - loss: 0.3236 - accuracy: 0.8695 - val_loss: 0.3860 - val_accuracy: 0.8438
Epoch 59/100
20/20 - 20s - loss: 0.3334 - accuracy: 0.8617 - val_loss: 0.3852 - val_accuracy: 0.8438
Epoch 60/100
20/20 - 20s - loss: 0.3427 - accuracy: 0.8602 - val_loss: 0.3848 - val_accuracy: 0.8438
Epoch 61/100
20/20 - 21s - loss: 0.3254 - accuracy: 0.8734 - val_loss: 0.3839 - val_accuracy: 0.8359
Epoch 62/100
20/20 - 20s - loss: 0.3506 - accuracy: 0.8648 - val_loss: 0.3838 - val_accuracy: 0.8359
Restoring model weights from the end of the best epoch.

Epoch 00062: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
Epoch 00062: early stopping
Out[20]: <tensorflow.python.keras.callbacks.History at 0x2c2277c7040>
```

```
In [21]: # Evaluate the model on test set
score = model.evaluate(x_test, y_test_gender, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.33726251125335693, 0.8605680465698242]
```

```
In [22]: labels =["Female", # index 0
           "Male",      # index 1
           ]
```

```
In [23]: y_hat = model.predict(x_test)

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test_gender[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                color=("green" if predict_index == true_index else "red"))
plt.show()
```



In []:

Old version without early stop

In [18]:

```
model.fit(x_train,
           y_train_gender,
           batch_size=64,
           epochs=10,
           validation_data=(x_valid, y_valid_gender),)
```

```
Epoch 1/10
339/339 [=====] - 219s 645ms/step - loss: 0.6475 - accuracy: 0.
7043 - val_loss: 0.4487 - val_accuracy: 0.8269
Epoch 2/10
339/339 [=====] - 221s 652ms/step - loss: 0.4231 - accuracy: 0.
8176 - val_loss: 0.3863 - val_accuracy: 0.8472
Epoch 3/10
339/339 [=====] - 236s 696ms/step - loss: 0.3776 - accuracy: 0.
8443 - val_loss: 0.3744 - val_accuracy: 0.8428
Epoch 4/10
339/339 [=====] - 312s 921ms/step - loss: 0.3532 - accuracy: 0.
8590 - val_loss: 0.3830 - val_accuracy: 0.8413
Epoch 5/10
339/339 [=====] - 381s 1s/step - loss: 0.3444 - accuracy: 0.858
7 - val_loss: 0.3689 - val_accuracy: 0.8520
Epoch 6/10
339/339 [=====] - 381s 1s/step - loss: 0.3275 - accuracy: 0.873
7 - val_loss: 0.3547 - val_accuracy: 0.8613
Epoch 7/10
339/339 [=====] - 381s 1s/step - loss: 0.3190 - accuracy: 0.872
6 - val_loss: 0.3470 - val_accuracy: 0.8587
Epoch 8/10
339/339 [=====] - 383s 1s/step - loss: 0.3093 - accuracy: 0.878
0 - val_loss: 0.3453 - val_accuracy: 0.8554
Epoch 9/10
339/339 [=====] - 390s 1s/step - loss: 0.3031 - accuracy: 0.880
6 - val_loss: 0.3416 - val_accuracy: 0.8590
Epoch 10/10
339/339 [=====] - 372s 1s/step - loss: 0.2860 - accuracy: 0.892
0 - val_loss: 0.3461 - val_accuracy: 0.8605
```

Out[18]: <tensorflow.python.keras.callbacks.History at 0x1aeb803be50>

In [24]:

```
# Evaluate the model on test set
score = model.evaluate(x_test, y_test_gender, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

Test accuracy: [0.36228927969932556, 0.8531907200813293]

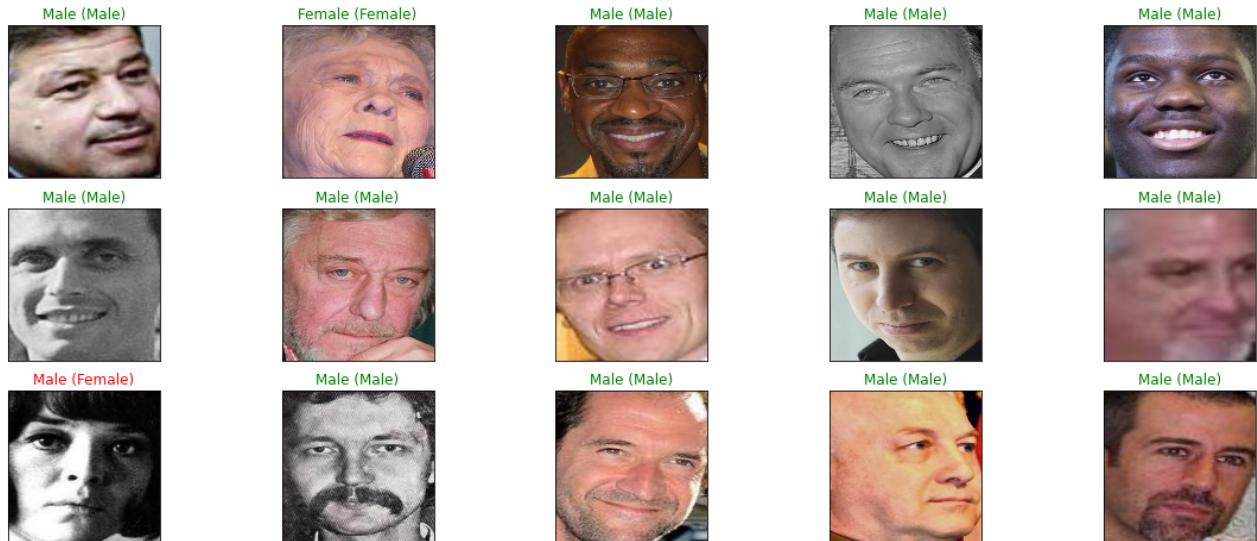
In [25]:

```
labels = ["Female", # index 0
          "Male",      # index 1
          ]
```

In [26]:

```
y_hat = model.predict(x_test)

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test_gender[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                 color="green" if predict_index == true_index else "red")
plt.show()
```



1-2. Two-layer CNN

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- kernel size = 3
- Dropout = 0.3

In []:

```
model = tf.keras.Sequential()
```

```

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(2, activation='sigmoid'))

# Take a look at the model summary
model.summary()

```

In [23]:

```

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

```

In [24]:

```

model.fit(x_train,y_train_gender,
           validation_data=(x_valid, y_valid_gender),
           batch_size=64,
           steps_per_epoch=20,
           epochs=100,
           validation_steps=2,
           verbose=2,
           workers=1,
           use_multiprocessing = False,
           callbacks=[earlystop, plateau])

```

```

Epoch 1/100
20/20 - 29s - loss: 1.0760 - accuracy: 0.6773 - val_loss: 0.6171 - val_accuracy: 0.8047
Epoch 2/100
20/20 - 25s - loss: 0.5981 - accuracy: 0.7305 - val_loss: 0.6750 - val_accuracy: 0.8047
Epoch 3/100
20/20 - 24s - loss: 0.6028 - accuracy: 0.7156 - val_loss: 0.5556 - val_accuracy: 0.8047
Epoch 4/100
20/20 - 24s - loss: 0.5764 - accuracy: 0.7078 - val_loss: 0.5153 - val_accuracy: 0.8047
Epoch 5/100
20/20 - 24s - loss: 0.5823 - accuracy: 0.7148 - val_loss: 0.5111 - val_accuracy: 0.7891
Epoch 6/100
20/20 - 24s - loss: 0.5956 - accuracy: 0.7164 - val_loss: 0.4713 - val_accuracy: 0.8047
Epoch 7/100
20/20 - 23s - loss: 0.5791 - accuracy: 0.7156 - val_loss: 0.4883 - val_accuracy: 0.7891

Epoch 00007: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.
Epoch 8/100
20/20 - 24s - loss: 0.5582 - accuracy: 0.7336 - val_loss: 0.4905 - val_accuracy: 0.7969
Epoch 9/100
20/20 - 24s - loss: 0.5382 - accuracy: 0.7422 - val_loss: 0.4862 - val_accuracy: 0.7969
Epoch 10/100
20/20 - 24s - loss: 0.5435 - accuracy: 0.7539 - val_loss: 0.4784 - val_accuracy: 0.7969
Epoch 11/100
20/20 - 24s - loss: 0.5475 - accuracy: 0.7484 - val_loss: 0.4895 - val_accuracy: 0.7969
Epoch 12/100
20/20 - 24s - loss: 0.5440 - accuracy: 0.7500 - val_loss: 0.4833 - val_accuracy: 0.7969
Epoch 13/100
20/20 - 24s - loss: 0.5214 - accuracy: 0.7695 - val_loss: 0.4701 - val_accuracy: 0.7891

```

```

Epoch 14/100
20/20 - 24s - loss: 0.5150 - accuracy: 0.7586 - val_loss: 0.4834 - val_accuracy: 0.7812
Epoch 15/100
20/20 - 24s - loss: 0.5456 - accuracy: 0.7414 - val_loss: 0.4765 - val_accuracy: 0.7969
Epoch 16/100
20/20 - 24s - loss: 0.5322 - accuracy: 0.7563 - val_loss: 0.4646 - val_accuracy: 0.7891
Epoch 17/100
20/20 - 24s - loss: 0.5215 - accuracy: 0.7658 - val_loss: 0.4648 - val_accuracy: 0.7891
Epoch 18/100
20/20 - 24s - loss: 0.5180 - accuracy: 0.7609 - val_loss: 0.4687 - val_accuracy: 0.8047

Epoch 00018: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 19/100
20/20 - 24s - loss: 0.5126 - accuracy: 0.7648 - val_loss: 0.4669 - val_accuracy: 0.8047
Epoch 20/100
20/20 - 24s - loss: 0.5242 - accuracy: 0.7664 - val_loss: 0.4670 - val_accuracy: 0.8047
Epoch 21/100
20/20 - 24s - loss: 0.5353 - accuracy: 0.7547 - val_loss: 0.4747 - val_accuracy: 0.8047
Epoch 22/100
20/20 - 24s - loss: 0.5162 - accuracy: 0.7539 - val_loss: 0.4753 - val_accuracy: 0.8047
Epoch 23/100
20/20 - 24s - loss: 0.4970 - accuracy: 0.7625 - val_loss: 0.4685 - val_accuracy: 0.8047
Restoring model weights from the end of the best epoch.

```

```

Epoch 00023: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
Epoch 00023: early stopping

```

Out[24]: <tensorflow.python.keras.callbacks.History at 0x2342ad4f940>

In [25]:

```

# Evaluate the model on test set
score = model.evaluate(x_test, y_test_gender, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)

```

Test accuracy: [0.4978107511997223, 0.8118775486946106]

Second Try

In [47]:

```

model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(2, activation='sigmoid'))

# Take a Look at the model summary
model.summary()

```

Model: "sequential_4"

Layer (type)	Output Shape	Param #
--------------	--------------	---------

```
=====
conv2d_8 (Conv2D)           (None, 128, 128, 64)    1792
max_pooling2d_8 (MaxPooling2D) (None, 64, 64, 64)    0
dropout_12 (Dropout)        (None, 64, 64, 64)    0
conv2d_9 (Conv2D)           (None, 64, 64, 32)     18464
max_pooling2d_9 (MaxPooling2D) (None, 32, 32, 32)    0
dropout_13 (Dropout)        (None, 32, 32, 32)    0
flatten_4 (Flatten)         (None, 32768)          0
dense_8 (Dense)             (None, 256)            8388864
dropout_14 (Dropout)        (None, 256)            0
dense_9 (Dense)             (None, 2)              514
=====
Total params: 8,409,634
Trainable params: 8,409,634
Non-trainable params: 0
```

In [48]:

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [49]:

```
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

Wed Apr 28 01:01:07 2021

In [50]:

```
model.fit(x_train,y_train_gender,
           validation_data=(x_valid, y_valid_gender),
           batch_size=64,
           steps_per_epoch=20,
           epochs=100,
           validation_steps=2,
           verbose=2,
           workers=1,
           use_multiprocessing = False,
           callbacks=[earlystop, plateau])
```

```
Epoch 1/100
20/20 - 192s - loss: 1.1628 - accuracy: 0.6406 - val_loss: 0.6356 - val_accuracy: 0.8047
Epoch 2/100
20/20 - 191s - loss: 0.6403 - accuracy: 0.6883 - val_loss: 0.6098 - val_accuracy: 0.8047
Epoch 3/100
20/20 - 160s - loss: 0.6073 - accuracy: 0.6922 - val_loss: 0.5906 - val_accuracy: 0.8047
Epoch 4/100
20/20 - 205s - loss: 0.5958 - accuracy: 0.7078 - val_loss: 0.5563 - val_accuracy: 0.8047
Epoch 5/100
20/20 - 163s - loss: 0.6029 - accuracy: 0.7039 - val_loss: 0.6043 - val_accuracy: 0.7969
Epoch 6/100
20/20 - 67s - loss: 0.5918 - accuracy: 0.6977 - val_loss: 0.4969 - val_accuracy: 0.7891
Epoch 7/100
20/20 - 64s - loss: 0.5484 - accuracy: 0.7398 - val_loss: 0.4701 - val_accuracy: 0.8047
Epoch 8/100
```

20/20 - 72s - loss: 0.5707 - accuracy: 0.7180 - val_loss: 0.5046 - val_accuracy: 0.7812
Epoch 9/100
20/20 - 104s - loss: 0.5367 - accuracy: 0.7500 - val_loss: 0.4810 - val_accuracy: 0.7891
Epoch 10/100
20/20 - 161s - loss: 0.5375 - accuracy: 0.7344 - val_loss: 0.4566 - val_accuracy: 0.8125
Epoch 11/100
20/20 - 159s - loss: 0.5409 - accuracy: 0.7320 - val_loss: 0.4704 - val_accuracy: 0.8047
Epoch 12/100
20/20 - 180s - loss: 0.5141 - accuracy: 0.7664 - val_loss: 0.4425 - val_accuracy: 0.8047
Epoch 13/100
20/20 - 184s - loss: 0.4943 - accuracy: 0.7820 - val_loss: 0.4278 - val_accuracy: 0.8203
Epoch 14/100
20/20 - 174s - loss: 0.5261 - accuracy: 0.7508 - val_loss: 0.4252 - val_accuracy: 0.8203
Epoch 15/100
20/20 - 217s - loss: 0.4931 - accuracy: 0.7703 - val_loss: 0.4172 - val_accuracy: 0.8281
Epoch 16/100
20/20 - 178s - loss: 0.4710 - accuracy: 0.7844 - val_loss: 0.4243 - val_accuracy: 0.8359
Epoch 17/100
20/20 - 171s - loss: 0.4593 - accuracy: 0.7863 - val_loss: 0.4240 - val_accuracy: 0.8125
Epoch 18/100
20/20 - 85s - loss: 0.4830 - accuracy: 0.7742 - val_loss: 0.4104 - val_accuracy: 0.8516
Epoch 19/100
20/20 - 89s - loss: 0.4310 - accuracy: 0.8062 - val_loss: 0.4106 - val_accuracy: 0.8125
Epoch 20/100
20/20 - 102s - loss: 0.4312 - accuracy: 0.8047 - val_loss: 0.3890 - val_accuracy: 0.8281
Epoch 21/100
20/20 - 91s - loss: 0.4052 - accuracy: 0.8211 - val_loss: 0.4056 - val_accuracy: 0.8203
Epoch 22/100
20/20 - 70s - loss: 0.4294 - accuracy: 0.8078 - val_loss: 0.4053 - val_accuracy: 0.8516
Epoch 23/100
20/20 - 102s - loss: 0.4068 - accuracy: 0.8359 - val_loss: 0.4110 - val_accuracy: 0.8281
Epoch 24/100
20/20 - 115s - loss: 0.4093 - accuracy: 0.8234 - val_loss: 0.4146 - val_accuracy: 0.8438
Epoch 25/100
20/20 - 122s - loss: 0.3982 - accuracy: 0.8383 - val_loss: 0.4340 - val_accuracy: 0.7969
Epoch 26/100
20/20 - 118s - loss: 0.4189 - accuracy: 0.8164 - val_loss: 0.4011 - val_accuracy: 0.8359
Epoch 27/100
20/20 - 107s - loss: 0.3944 - accuracy: 0.8328 - val_loss: 0.3760 - val_accuracy: 0.8281
Epoch 28/100
20/20 - 81s - loss: 0.3917 - accuracy: 0.8250 - val_loss: 0.3726 - val_accuracy: 0.8203
Epoch 29/100
20/20 - 82s - loss: 0.3772 - accuracy: 0.8391 - val_loss: 0.3875 - val_accuracy: 0.8438
Epoch 30/100
20/20 - 83s - loss: 0.3828 - accuracy: 0.8313 - val_loss: 0.3997 - val_accuracy: 0.8438
Epoch 31/100
20/20 - 86s - loss: 0.3809 - accuracy: 0.8328 - val_loss: 0.3859 - val_accuracy: 0.8281
Epoch 32/100
20/20 - 84s - loss: 0.3695 - accuracy: 0.8523 - val_loss: 0.3922 - val_accuracy: 0.8438
Epoch 33/100
20/20 - 94s - loss: 0.3702 - accuracy: 0.8383 - val_loss: 0.3853 - val_accuracy: 0.8516
Epoch 34/100
20/20 - 128s - loss: 0.3483 - accuracy: 0.8580 - val_loss: 0.3819 - val_accuracy: 0.8594
Epoch 35/100
20/20 - 33s - loss: 0.3313 - accuracy: 0.8750 - val_loss: 0.3789 - val_accuracy: 0.8438
Epoch 36/100
20/20 - 32s - loss: 0.3738 - accuracy: 0.8438 - val_loss: 0.3926 - val_accuracy: 0.8281
Epoch 37/100
20/20 - 34s - loss: 0.3773 - accuracy: 0.8398 - val_loss: 0.3857 - val_accuracy: 0.8516
Epoch 38/100
20/20 - 35s - loss: 0.3871 - accuracy: 0.8313 - val_loss: 0.3874 - val_accuracy: 0.8281
Epoch 39/100
20/20 - 38s - loss: 0.3550 - accuracy: 0.8586 - val_loss: 0.3847 - val_accuracy: 0.8672
Epoch 40/100
20/20 - 57s - loss: 0.3572 - accuracy: 0.8313 - val_loss: 0.3592 - val_accuracy: 0.8672

```
Epoch 00040: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.
Epoch 41/100
20/20 - 95s - loss: 0.3545 - accuracy: 0.8469 - val_loss: 0.3617 - val_accuracy: 0.8594
Epoch 42/100
20/20 - 125s - loss: 0.3180 - accuracy: 0.8789 - val_loss: 0.3587 - val_accuracy: 0.8516
Epoch 43/100
20/20 - 139s - loss: 0.3373 - accuracy: 0.8742 - val_loss: 0.3572 - val_accuracy: 0.8516
Epoch 44/100
20/20 - 111s - loss: 0.3365 - accuracy: 0.8438 - val_loss: 0.3580 - val_accuracy: 0.8516
Epoch 45/100
20/20 - 162s - loss: 0.3483 - accuracy: 0.8555 - val_loss: 0.3543 - val_accuracy: 0.8516
Epoch 46/100
20/20 - 110s - loss: 0.3118 - accuracy: 0.8703 - val_loss: 0.3503 - val_accuracy: 0.8594
Epoch 47/100
20/20 - 100s - loss: 0.3443 - accuracy: 0.8742 - val_loss: 0.3520 - val_accuracy: 0.8594

Epoch 00047: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 48/100
20/20 - 109s - loss: 0.3242 - accuracy: 0.8633 - val_loss: 0.3520 - val_accuracy: 0.8594
Epoch 49/100
20/20 - 126s - loss: 0.3467 - accuracy: 0.8633 - val_loss: 0.3525 - val_accuracy: 0.8672
Epoch 50/100
20/20 - 101s - loss: 0.3697 - accuracy: 0.8406 - val_loss: 0.3531 - val_accuracy: 0.8594
Epoch 51/100
20/20 - 92s - loss: 0.3522 - accuracy: 0.8462 - val_loss: 0.3537 - val_accuracy: 0.8594
Epoch 52/100
20/20 - 109s - loss: 0.3280 - accuracy: 0.8703 - val_loss: 0.3540 - val_accuracy: 0.8594
Restoring model weights from the end of the best epoch.

Epoch 00052: ReduceLROnPlateau reducing learning rate to 1.000000656873453e-06.
Epoch 00052: early stopping
```

Out[50]: <tensorflow.python.keras.callbacks.History at 0x2c22a3712e0>

In [51]:

```
# Evaluate the model on test set
score = model.evaluate(x_test, y_test_gender, verbose=0)

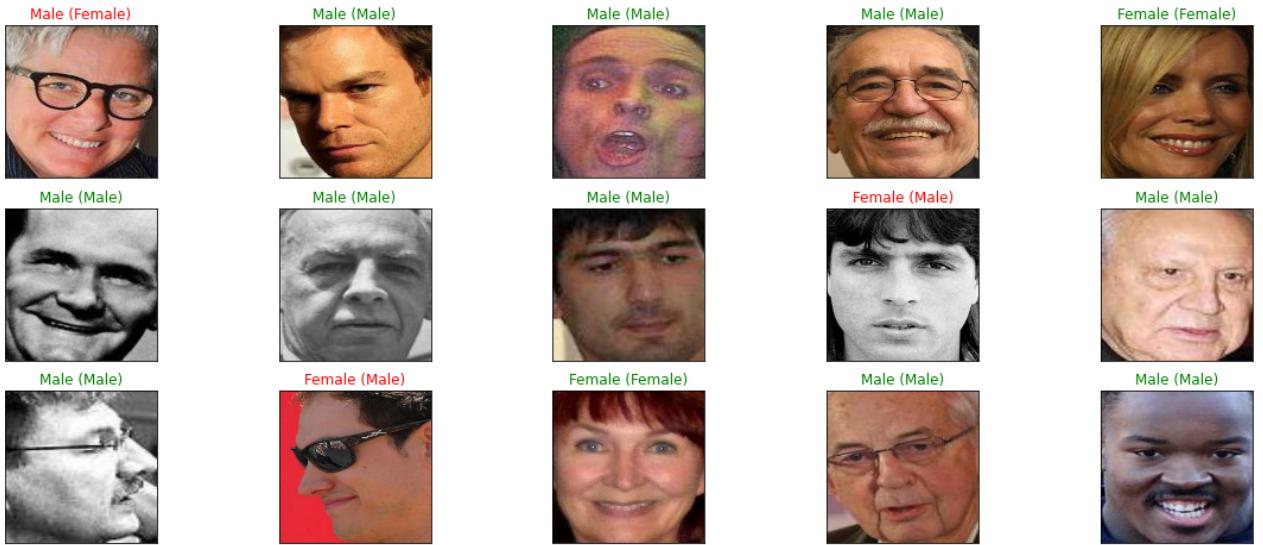
# Print test accuracy
print('\n', 'Test accuracy:', score)
```

Test accuracy: [0.34186938405036926, 0.8587237000465393]

In [52]:

```
y_hat = model.predict(x_test)

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test_gender[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                color=("green" if predict_index == true_index else "red"))
plt.show()
```



In []:

Old version without early stop

In [20]:

```
model.fit(x_train,
           y_train_gender,
           batch_size=64,
           epochs=10,
           validation_data=(x_valid, y_valid_gender),)
```

```
Epoch 1/10
339/339 [=====] - 366s 1s/step - loss: 0.7375 - accuracy: 0.704
6 - val_loss: 0.4378 - val_accuracy: 0.8044
Epoch 2/10
339/339 [=====] - 448s 1s/step - loss: 0.4390 - accuracy: 0.803
9 - val_loss: 0.4135 - val_accuracy: 0.8199
Epoch 3/10
339/339 [=====] - 449s 1s/step - loss: 0.3961 - accuracy: 0.828
9 - val_loss: 0.3708 - val_accuracy: 0.8483
Epoch 4/10
339/339 [=====] - 445s 1s/step - loss: 0.3522 - accuracy: 0.855
6 - val_loss: 0.3542 - val_accuracy: 0.8509
Epoch 5/10
339/339 [=====] - 445s 1s/step - loss: 0.3273 - accuracy: 0.866
1 - val_loss: 0.3689 - val_accuracy: 0.8528
Epoch 6/10
339/339 [=====] - 444s 1s/step - loss: 0.3079 - accuracy: 0.874
4 - val_loss: 0.3412 - val_accuracy: 0.8594
Epoch 7/10
339/339 [=====] - 440s 1s/step - loss: 0.2939 - accuracy: 0.878
8 - val_loss: 0.3299 - val_accuracy: 0.8716
Epoch 8/10
339/339 [=====] - 443s 1s/step - loss: 0.2696 - accuracy: 0.890
8 - val_loss: 0.3298 - val_accuracy: 0.8672
Epoch 9/10
339/339 [=====] - 446s 1s/step - loss: 0.2572 - accuracy: 0.894
9 - val_loss: 0.3128 - val_accuracy: 0.8801
Epoch 10/10
339/339 [=====] - 443s 1s/step - loss: 0.2262 - accuracy: 0.910
4 - val_loss: 0.3002 - val_accuracy: 0.8852
```

Out[20]: <tensorflow.python.keras.callbacks.History at 0x223d4a4e790>

In [21]:

```
# Evaluate the model on test set
score = model.evaluate(x_test, y_test_gender, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

Test accuracy: [0.28964030742645264, 0.8845444321632385]

1-3. Two-layer CNN

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- Change kernel size: kernel size = 5
- Dropout = 0.3

In [26]:

```
model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=5, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=5, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(2, activation='sigmoid'))

# Take a look at the model summary
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 128, 128, 64)	4864
max_pooling2d_4 (MaxPooling2D)	(None, 64, 64, 64)	0
dropout_6 (Dropout)	(None, 64, 64, 64)	0
conv2d_5 (Conv2D)	(None, 64, 64, 32)	51232
max_pooling2d_5 (MaxPooling2D)	(None, 32, 32, 32)	0
dropout_7 (Dropout)	(None, 32, 32, 32)	0
flatten_2 (Flatten)	(None, 32768)	0
dense_4 (Dense)	(None, 256)	8388864
dropout_8 (Dropout)	(None, 256)	0
dense_5 (Dense)	(None, 2)	514
Total params: 8,445,474		

```
Trainable params: 8,445,474  
Non-trainable params: 0
```

In [27]:

```
import time  
localtime = time.asctime( time.localtime(time.time()) )  
print(localtime)
```

Tue Apr 27 17:50:19 2021

In [28]:

```
model.compile(loss='binary_crossentropy',  
              optimizer='adam',  
              metrics=['accuracy'])
```

In [29]:

```
model.fit(x_train,y_train_gender,  
          validation_data=(x_valid, y_valid_gender),  
          batch_size=64,  
          steps_per_epoch=20,  
          epochs=100,  
          validation_steps=2,  
          verbose=2,  
          workers=1,  
          use_multiprocessing = False,  
          callbacks=[earlystop, plateau])
```

```
Epoch 1/100  
20/20 - 39s - loss: 0.8577 - accuracy: 0.6938 - val_loss: 0.5530 - val_accuracy: 0.8047  
Epoch 2/100  
20/20 - 37s - loss: 0.5915 - accuracy: 0.7289 - val_loss: 0.5985 - val_accuracy: 0.8047  
Epoch 3/100  
20/20 - 37s - loss: 0.5545 - accuracy: 0.7352 - val_loss: 0.4833 - val_accuracy: 0.7969  
Epoch 4/100  
20/20 - 37s - loss: 0.6114 - accuracy: 0.7016 - val_loss: 0.5263 - val_accuracy: 0.8047  
Epoch 5/100  
20/20 - 37s - loss: 0.5915 - accuracy: 0.6953 - val_loss: 0.4890 - val_accuracy: 0.7969  
Epoch 6/100  
20/20 - 37s - loss: 0.5681 - accuracy: 0.7188 - val_loss: 0.4667 - val_accuracy: 0.8125  
Epoch 7/100  
20/20 - 37s - loss: 0.5352 - accuracy: 0.7484 - val_loss: 0.4510 - val_accuracy: 0.8047  
Epoch 8/100  
20/20 - 37s - loss: 0.5464 - accuracy: 0.7523 - val_loss: 0.4765 - val_accuracy: 0.7969  
Epoch 9/100  
20/20 - 37s - loss: 0.5433 - accuracy: 0.7391 - val_loss: 0.5183 - val_accuracy: 0.8281  
Epoch 10/100  
20/20 - 37s - loss: 0.5685 - accuracy: 0.7359 - val_loss: 0.4703 - val_accuracy: 0.7969  
Epoch 11/100  
20/20 - 37s - loss: 0.5281 - accuracy: 0.7711 - val_loss: 0.4649 - val_accuracy: 0.8359  
Epoch 12/100  
20/20 - 37s - loss: 0.5008 - accuracy: 0.7758 - val_loss: 0.4438 - val_accuracy: 0.7969  
Epoch 13/100  
20/20 - 37s - loss: 0.5210 - accuracy: 0.7547 - val_loss: 0.4261 - val_accuracy: 0.8047  
Epoch 14/100  
20/20 - 37s - loss: 0.5022 - accuracy: 0.7750 - val_loss: 0.4334 - val_accuracy: 0.8516  
Epoch 15/100  
20/20 - 37s - loss: 0.4936 - accuracy: 0.7844 - val_loss: 0.4110 - val_accuracy: 0.8281  
Epoch 16/100  
20/20 - 37s - loss: 0.4744 - accuracy: 0.7953 - val_loss: 0.4293 - val_accuracy: 0.8281  
Epoch 17/100  
20/20 - 37s - loss: 0.4900 - accuracy: 0.7752 - val_loss: 0.4102 - val_accuracy: 0.8359  
Epoch 18/100  
20/20 - 38s - loss: 0.4193 - accuracy: 0.8172 - val_loss: 0.3874 - val_accuracy: 0.8359
```

Epoch 19/100
20/20 - 37s - loss: 0.4748 - accuracy: 0.7859 - val_loss: 0.3971 - val_accuracy: 0.8281
Epoch 20/100
20/20 - 37s - loss: 0.4395 - accuracy: 0.8133 - val_loss: 0.4085 - val_accuracy: 0.8125
Epoch 21/100
20/20 - 37s - loss: 0.4591 - accuracy: 0.7875 - val_loss: 0.4016 - val_accuracy: 0.8359
Epoch 22/100
20/20 - 37s - loss: 0.4250 - accuracy: 0.8094 - val_loss: 0.4070 - val_accuracy: 0.8281
Epoch 23/100
20/20 - 37s - loss: 0.4613 - accuracy: 0.7953 - val_loss: 0.4607 - val_accuracy: 0.8047

Epoch 00023: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.

Epoch 24/100
20/20 - 37s - loss: 0.4319 - accuracy: 0.8133 - val_loss: 0.3856 - val_accuracy: 0.8438
Epoch 25/100
20/20 - 37s - loss: 0.4139 - accuracy: 0.8180 - val_loss: 0.3841 - val_accuracy: 0.8438
Epoch 26/100
20/20 - 37s - loss: 0.3825 - accuracy: 0.8367 - val_loss: 0.3847 - val_accuracy: 0.8359
Epoch 27/100
20/20 - 39s - loss: 0.3854 - accuracy: 0.8367 - val_loss: 0.3827 - val_accuracy: 0.8438
Epoch 28/100
20/20 - 39s - loss: 0.3938 - accuracy: 0.8273 - val_loss: 0.3840 - val_accuracy: 0.8359
Epoch 29/100
20/20 - 38s - loss: 0.3757 - accuracy: 0.8438 - val_loss: 0.3813 - val_accuracy: 0.8438
Epoch 30/100
20/20 - 39s - loss: 0.3879 - accuracy: 0.8438 - val_loss: 0.3842 - val_accuracy: 0.8359
Epoch 31/100
20/20 - 38s - loss: 0.3854 - accuracy: 0.8375 - val_loss: 0.3828 - val_accuracy: 0.8438
Epoch 32/100
20/20 - 37s - loss: 0.4112 - accuracy: 0.8320 - val_loss: 0.3826 - val_accuracy: 0.8359
Epoch 33/100
20/20 - 37s - loss: 0.3821 - accuracy: 0.8328 - val_loss: 0.3792 - val_accuracy: 0.8516
Epoch 34/100
20/20 - 37s - loss: 0.3906 - accuracy: 0.8391 - val_loss: 0.3805 - val_accuracy: 0.8281

Epoch 00034: ReduceLROnPlateau reducing learning rate to 1.000000474974514e-05.

Epoch 35/100
20/20 - 37s - loss: 0.3531 - accuracy: 0.8562 - val_loss: 0.3794 - val_accuracy: 0.8281
Epoch 36/100
20/20 - 37s - loss: 0.3932 - accuracy: 0.8266 - val_loss: 0.3789 - val_accuracy: 0.8359
Epoch 37/100
20/20 - 38s - loss: 0.3442 - accuracy: 0.8641 - val_loss: 0.3784 - val_accuracy: 0.8359
Epoch 38/100
20/20 - 37s - loss: 0.3752 - accuracy: 0.8359 - val_loss: 0.3783 - val_accuracy: 0.8359
Epoch 39/100
20/20 - 37s - loss: 0.3758 - accuracy: 0.8336 - val_loss: 0.3786 - val_accuracy: 0.8359
Epoch 40/100
20/20 - 37s - loss: 0.3751 - accuracy: 0.8414 - val_loss: 0.3785 - val_accuracy: 0.8359
Epoch 41/100
20/20 - 37s - loss: 0.3678 - accuracy: 0.8453 - val_loss: 0.3785 - val_accuracy: 0.8359
Epoch 42/100
20/20 - 37s - loss: 0.3579 - accuracy: 0.8516 - val_loss: 0.3781 - val_accuracy: 0.8438

Epoch 00042: ReduceLROnPlateau reducing learning rate to 1.000000656873453e-06.

Epoch 43/100
20/20 - 38s - loss: 0.3548 - accuracy: 0.8461 - val_loss: 0.3781 - val_accuracy: 0.8438
Epoch 44/100
20/20 - 37s - loss: 0.3805 - accuracy: 0.8367 - val_loss: 0.3781 - val_accuracy: 0.8359
Epoch 45/100
20/20 - 37s - loss: 0.3427 - accuracy: 0.8547 - val_loss: 0.3781 - val_accuracy: 0.8438
Epoch 46/100
20/20 - 37s - loss: 0.4066 - accuracy: 0.8328 - val_loss: 0.3782 - val_accuracy: 0.8359
Epoch 47/100
20/20 - 37s - loss: 0.3673 - accuracy: 0.8453 - val_loss: 0.3782 - val_accuracy: 0.8359
Restoring model weights from the end of the best epoch.

```
Epoch 00047: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.  
Epoch 00047: early stopping  
Out[29]: <tensorflow.python.keras.callbacks.History at 0x2342b8f36d0>
```

```
In [30]: # Evaluate the model on test set  
score = model.evaluate(x_test, y_test_gender, verbose=0)  
  
# Print test accuracy  
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.3879869878292084, 0.83253413438797]
```

Old version

```
In [25]: model.fit(x_train,  
                 y_train_gender,  
                 batch_size=64,  
                 epochs=10,  
                 validation_data=(x_valid, y_valid_gender),)  
  
Epoch 1/10  
339/339 [=====] - 723s 2s/step - loss: 0.6981 - accuracy: 0.706  
8 - val_loss: 0.4767 - val_accuracy: 0.7738  
Epoch 2/10  
339/339 [=====] - 761s 2s/step - loss: 0.4588 - accuracy: 0.799  
5 - val_loss: 0.3885 - val_accuracy: 0.8328  
Epoch 3/10  
339/339 [=====] - 1001s 3s/step - loss: 0.3805 - accuracy: 0.83  
84 - val_loss: 0.3527 - val_accuracy: 0.8469  
Epoch 4/10  
339/339 [=====] - 703s 2s/step - loss: 0.3306 - accuracy: 0.860  
7 - val_loss: 0.3390 - val_accuracy: 0.8587  
Epoch 5/10  
339/339 [=====] - 675s 2s/step - loss: 0.2974 - accuracy: 0.875  
6 - val_loss: 0.3121 - val_accuracy: 0.8679  
Epoch 6/10  
339/339 [=====] - 676s 2s/step - loss: 0.2742 - accuracy: 0.886  
2 - val_loss: 0.3065 - val_accuracy: 0.8694  
Epoch 7/10  
339/339 [=====] - 674s 2s/step - loss: 0.2598 - accuracy: 0.891  
3 - val_loss: 0.3074 - val_accuracy: 0.8708  
Epoch 8/10  
339/339 [=====] - 513s 2s/step - loss: 0.2505 - accuracy: 0.898  
3 - val_loss: 0.3057 - val_accuracy: 0.8760  
Epoch 9/10  
339/339 [=====] - 381s 1s/step - loss: 0.2342 - accuracy: 0.905  
7 - val_loss: 0.3058 - val_accuracy: 0.8727  
Epoch 10/10  
339/339 [=====] - 382s 1s/step - loss: 0.2149 - accuracy: 0.912  
1 - val_loss: 0.3058 - val_accuracy: 0.8768  
Out[25]: <tensorflow.python.keras.callbacks.History at 0x2242af90b20>
```

```
In [26]: import time  
localtime = time.asctime( time.localtime(time.time()) )  
print(localtime)
```

```
Sun Apr 25 17:18:03 2021
```

```
In [27]: # Evaluate the model on test set
score = model.evaluate(x_test, y_test_gender, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

Test accuracy: [0.3115690052509308, 0.8856510519981384]

Build the multilayer CNN (change fully connected layer)

2-2. Two-layer CNN (3 fully connected layers)

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- kernel size = 5
- Dropout = 0.3

```
In [31]: model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=5, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=5, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(2, activation='sigmoid'))

# Take a look at the model summary
model.summary()
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 128, 128, 64)	4864
max_pooling2d_6 (MaxPooling2D)	(None, 64, 64, 64)	0
dropout_9 (Dropout)	(None, 64, 64, 64)	0
conv2d_7 (Conv2D)	(None, 64, 64, 32)	51232
max_pooling2d_7 (MaxPooling2D)	(None, 32, 32, 32)	0
dropout_10 (Dropout)	(None, 32, 32, 32)	0
flatten_3 (Flatten)	(None, 32768)	0
dense_6 (Dense)	(None, 256)	8388864

dropout_11 (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 128)	32896
dropout_12 (Dropout)	(None, 128)	0
dense_8 (Dense)	(None, 2)	258

Total params: 8,478,114
Trainable params: 8,478,114
Non-trainable params: 0

```
In [32]: model.compile(loss='binary_crossentropy',
                    optimizer='adam',
                    metrics=[ 'accuracy'])
```

```
In [33]: model.fit(x_train,y_train_gender,
                 validation_data=(x_valid, y_valid_gender),
                 batch_size=64,
                 steps_per_epoch=20,
                 epochs=100,
                 validation_steps=2,
                 verbose=2,
                 workers=1,
                 use_multiprocessing = False,
                 callbacks=[earlystop, plateau])
```

```
Epoch 1/100
20/20 - 38s - loss: 0.6926 - accuracy: 0.6672 - val_loss: 0.6715 - val_accuracy: 0.8047
Epoch 2/100
20/20 - 37s - loss: 0.6429 - accuracy: 0.6992 - val_loss: 0.5861 - val_accuracy: 0.8047
Epoch 3/100
20/20 - 37s - loss: 0.5934 - accuracy: 0.7234 - val_loss: 0.5243 - val_accuracy: 0.8047
Epoch 4/100
20/20 - 37s - loss: 0.5941 - accuracy: 0.7133 - val_loss: 0.5101 - val_accuracy: 0.8047
Epoch 5/100
20/20 - 37s - loss: 0.5862 - accuracy: 0.7258 - val_loss: 0.4972 - val_accuracy: 0.7969
Epoch 6/100
20/20 - 37s - loss: 0.5693 - accuracy: 0.7242 - val_loss: 0.5230 - val_accuracy: 0.8047
Epoch 7/100
20/20 - 35s - loss: 0.5843 - accuracy: 0.7164 - val_loss: 0.5248 - val_accuracy: 0.7734
Epoch 8/100
20/20 - 37s - loss: 0.5431 - accuracy: 0.7484 - val_loss: 0.4675 - val_accuracy: 0.7656
Epoch 9/100
20/20 - 37s - loss: 0.5365 - accuracy: 0.7391 - val_loss: 0.4694 - val_accuracy: 0.7891
Epoch 10/100
20/20 - 37s - loss: 0.5455 - accuracy: 0.7422 - val_loss: 0.4580 - val_accuracy: 0.7812
Epoch 11/100
20/20 - 37s - loss: 0.5375 - accuracy: 0.7414 - val_loss: 0.4090 - val_accuracy: 0.8125
Epoch 12/100
20/20 - 37s - loss: 0.5245 - accuracy: 0.7617 - val_loss: 0.4335 - val_accuracy: 0.8125
Epoch 13/100
20/20 - 37s - loss: 0.4993 - accuracy: 0.7727 - val_loss: 0.4375 - val_accuracy: 0.7969
Epoch 14/100
20/20 - 37s - loss: 0.4591 - accuracy: 0.8039 - val_loss: 0.4051 - val_accuracy: 0.8203
Epoch 15/100
20/20 - 37s - loss: 0.4538 - accuracy: 0.7906 - val_loss: 0.4095 - val_accuracy: 0.8203
Epoch 16/100
20/20 - 37s - loss: 0.4516 - accuracy: 0.8070 - val_loss: 0.4259 - val_accuracy: 0.8047
Epoch 17/100
20/20 - 36s - loss: 0.4981 - accuracy: 0.7697 - val_loss: 0.4079 - val_accuracy: 0.8125
```

Epoch 18/100
20/20 - 37s - loss: 0.4079 - accuracy: 0.8313 - val_loss: 0.3731 - val_accuracy: 0.8125
Epoch 19/100
20/20 - 37s - loss: 0.3966 - accuracy: 0.8391 - val_loss: 0.3620 - val_accuracy: 0.8594
Epoch 20/100
20/20 - 37s - loss: 0.4137 - accuracy: 0.8203 - val_loss: 0.3588 - val_accuracy: 0.8438
Epoch 21/100
20/20 - 37s - loss: 0.4084 - accuracy: 0.8219 - val_loss: 0.3666 - val_accuracy: 0.8359
Epoch 22/100
20/20 - 37s - loss: 0.4036 - accuracy: 0.8211 - val_loss: 0.3643 - val_accuracy: 0.8359
Epoch 23/100
20/20 - 37s - loss: 0.4201 - accuracy: 0.8117 - val_loss: 0.3555 - val_accuracy: 0.8359
Epoch 24/100
20/20 - 36s - loss: 0.4043 - accuracy: 0.8305 - val_loss: 0.3416 - val_accuracy: 0.8438

Epoch 00024: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.
Epoch 25/100
20/20 - 37s - loss: 0.3946 - accuracy: 0.8273 - val_loss: 0.3617 - val_accuracy: 0.8281
Epoch 26/100
20/20 - 37s - loss: 0.3847 - accuracy: 0.8508 - val_loss: 0.3376 - val_accuracy: 0.8203
Epoch 27/100
20/20 - 37s - loss: 0.3685 - accuracy: 0.8391 - val_loss: 0.3347 - val_accuracy: 0.8281
Epoch 28/100
20/20 - 37s - loss: 0.3831 - accuracy: 0.8359 - val_loss: 0.3394 - val_accuracy: 0.8281
Epoch 29/100
20/20 - 37s - loss: 0.3815 - accuracy: 0.8359 - val_loss: 0.3403 - val_accuracy: 0.8438
Epoch 30/100
20/20 - 37s - loss: 0.4023 - accuracy: 0.8414 - val_loss: 0.3350 - val_accuracy: 0.8281
Epoch 31/100
20/20 - 37s - loss: 0.3682 - accuracy: 0.8422 - val_loss: 0.3371 - val_accuracy: 0.8281

Epoch 00031: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 32/100
20/20 - 37s - loss: 0.3771 - accuracy: 0.8469 - val_loss: 0.3374 - val_accuracy: 0.8281
Epoch 33/100
20/20 - 37s - loss: 0.3689 - accuracy: 0.8555 - val_loss: 0.3363 - val_accuracy: 0.8281
Epoch 34/100
20/20 - 36s - loss: 0.3848 - accuracy: 0.8541 - val_loss: 0.3361 - val_accuracy: 0.8281
Epoch 35/100
20/20 - 37s - loss: 0.3801 - accuracy: 0.8344 - val_loss: 0.3359 - val_accuracy: 0.8281
Epoch 36/100
20/20 - 37s - loss: 0.3506 - accuracy: 0.8586 - val_loss: 0.3361 - val_accuracy: 0.8359
Epoch 37/100
20/20 - 36s - loss: 0.3615 - accuracy: 0.8422 - val_loss: 0.3354 - val_accuracy: 0.8281
Epoch 38/100
20/20 - 37s - loss: 0.3820 - accuracy: 0.8336 - val_loss: 0.3358 - val_accuracy: 0.8281
Epoch 39/100
20/20 - 37s - loss: 0.3692 - accuracy: 0.8430 - val_loss: 0.3361 - val_accuracy: 0.8281
Epoch 40/100
20/20 - 37s - loss: 0.3553 - accuracy: 0.8492 - val_loss: 0.3346 - val_accuracy: 0.8281
Epoch 41/100
20/20 - 36s - loss: 0.3500 - accuracy: 0.8609 - val_loss: 0.3340 - val_accuracy: 0.8281
Epoch 42/100
20/20 - 36s - loss: 0.3375 - accuracy: 0.8656 - val_loss: 0.3333 - val_accuracy: 0.8281
Epoch 43/100
20/20 - 36s - loss: 0.3413 - accuracy: 0.8602 - val_loss: 0.3320 - val_accuracy: 0.8281
Epoch 44/100
20/20 - 37s - loss: 0.3449 - accuracy: 0.8508 - val_loss: 0.3311 - val_accuracy: 0.8203
Epoch 45/100
20/20 - 37s - loss: 0.3695 - accuracy: 0.8555 - val_loss: 0.3314 - val_accuracy: 0.8203
Epoch 46/100
20/20 - 36s - loss: 0.3703 - accuracy: 0.8313 - val_loss: 0.3309 - val_accuracy: 0.8203
Epoch 47/100
20/20 - 37s - loss: 0.3834 - accuracy: 0.8320 - val_loss: 0.3313 - val_accuracy: 0.8203

```
Epoch 00047: ReduceLROnPlateau reducing learning rate to 1.000000656873453e-06.
Epoch 48/100
20/20 - 37s - loss: 0.3587 - accuracy: 0.8633 - val_loss: 0.3313 - val_accuracy: 0.8203
Epoch 49/100
20/20 - 37s - loss: 0.3605 - accuracy: 0.8414 - val_loss: 0.3313 - val_accuracy: 0.8203
Epoch 50/100
20/20 - 37s - loss: 0.3587 - accuracy: 0.8461 - val_loss: 0.3313 - val_accuracy: 0.8203
Epoch 51/100
20/20 - 37s - loss: 0.3561 - accuracy: 0.8438 - val_loss: 0.3312 - val_accuracy: 0.8203
Epoch 52/100
20/20 - 37s - loss: 0.3868 - accuracy: 0.8313 - val_loss: 0.3311 - val_accuracy: 0.8203
Restoring model weights from the end of the best epoch.

Epoch 00052: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.
Epoch 00052: early stopping
Out[33]: <tensorflow.python.keras.callbacks.History at 0x2342b6d0550>
```

```
In [34]: # Evaluate the model on test set
score = model.evaluate(x_test, y_test_gender, verbose=0)

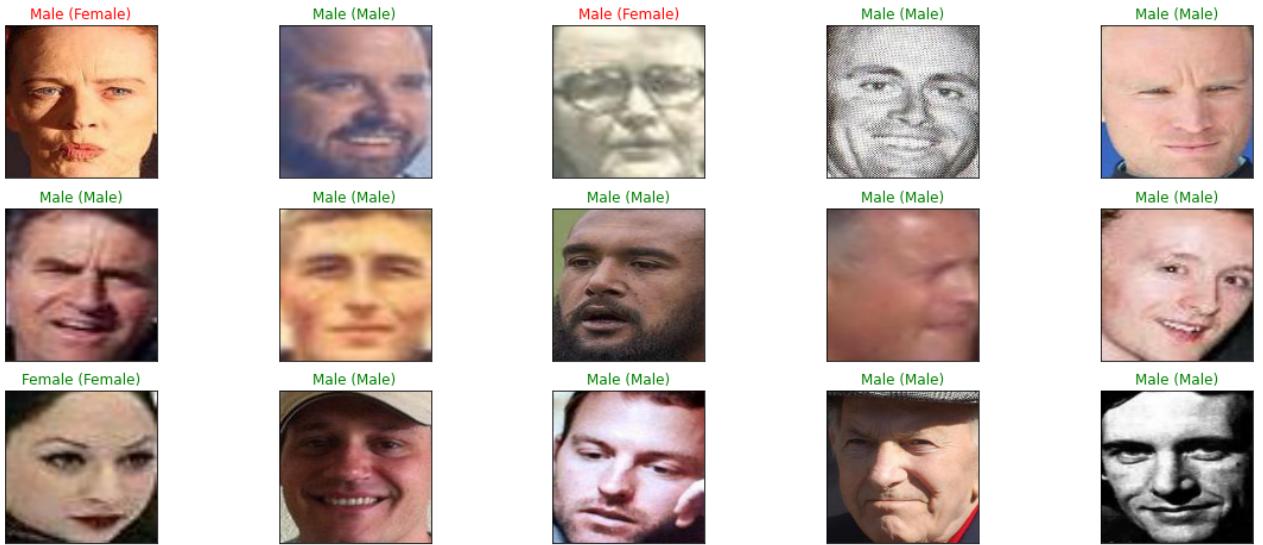
# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.3826124370098114, 0.8376982808113098]
```

```
In [36]: labels =["Female", # index 0
           "Male",      # index 1
           ]
```

```
In [37]: y_hat = model.predict(x_test)

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test_gender[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                color=("green" if predict_index == true_index else "red"))
plt.show()
```



Old Version

```
In [26]: import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

Tue Apr 27 04:36:00 2021

```
In [27]: model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
```

```
In [28]: model.fit(x_train,
                 y_train_gender,
                 batch_size=64,
                 epochs=10,
                 validation_data=(x_valid, y_valid_gender),)
```

Epoch 1/10
339/339 [=====] - 688s 2s/step - loss: 0.6201 - accuracy: 0.719
8 - val_loss: 0.4406 - val_accuracy: 0.8077
Epoch 2/10
339/339 [=====] - 628s 2s/step - loss: 0.3956 - accuracy: 0.830
2 - val_loss: 0.3680 - val_accuracy: 0.8410
Epoch 3/10
339/339 [=====] - 385s 1s/step - loss: 0.3298 - accuracy: 0.864
1 - val_loss: 0.3307 - val_accuracy: 0.8613
Epoch 4/10
339/339 [=====] - 384s 1s/step - loss: 0.2924 - accuracy: 0.882
8 - val_loss: 0.3239 - val_accuracy: 0.8653
Epoch 5/10
339/339 [=====] - 385s 1s/step - loss: 0.2715 - accuracy: 0.889
6 - val_loss: 0.2986 - val_accuracy: 0.8672
Epoch 6/10
339/339 [=====] - 383s 1s/step - loss: 0.2479 - accuracy: 0.900
1 - val_loss: 0.3233 - val_accuracy: 0.8627
Epoch 7/10
339/339 [=====] - 383s 1s/step - loss: 0.2276 - accuracy: 0.910
2 - val_loss: 0.2705 - val_accuracy: 0.8871
Epoch 8/10
339/339 [=====] - 385s 1s/step - loss: 0.1939 - accuracy: 0.923

```
1 - val_loss: 0.2660 - val_accuracy: 0.8919
Epoch 9/10
339/339 [=====] - 383s 1s/step - loss: 0.1814 - accuracy: 0.927
6 - val_loss: 0.2700 - val_accuracy: 0.8915
Epoch 10/10
339/339 [=====] - 386s 1s/step - loss: 0.1631 - accuracy: 0.938
1 - val_loss: 0.2748 - val_accuracy: 0.8978
Out[28]: <tensorflow.python.keras.callbacks.History at 0x1af2a9faa60>
```

```
In [29]: # Evaluate the model on test set
score = model.evaluate(x_test, y_test_gender, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.27913227677345276, 0.897823691368103]
```

```
In [30]: labels =["Female", # index 0
            "Male",      # index 1
            ]
```

```
In [31]: y_hat = model.predict(x_test)

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test_gender[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                color=("green" if predict_index == true_index else "red"))
plt.show()
```



Build the multilayer CNN (change fully connected layer)

2-3. Two-layer CNN (3 fully connected layers)

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- kernel size = 5
- Dropout = 0.3

```
In [38]: model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=5, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=5, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.3))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(64, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(2, activation='sigmoid'))

# Take a Look at the model summary
model.summary()
```

Model: "sequential_4"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_8 (Conv2D)	(None, 128, 128, 64)	4864
<hr/>		
max_pooling2d_8 (MaxPooling2D)	(None, 64, 64, 64)	0
<hr/>		
dropout_13 (Dropout)	(None, 64, 64, 64)	0
<hr/>		
conv2d_9 (Conv2D)	(None, 64, 64, 32)	51232
<hr/>		
max_pooling2d_9 (MaxPooling2D)	(None, 32, 32, 32)	0
<hr/>		
dropout_14 (Dropout)	(None, 32, 32, 32)	0
<hr/>		
flatten_4 (Flatten)	(None, 32768)	0
<hr/>		
dense_9 (Dense)	(None, 128)	4194432
<hr/>		
dropout_15 (Dropout)	(None, 128)	0
<hr/>		
dense_10 (Dense)	(None, 64)	8256
<hr/>		
dropout_16 (Dropout)	(None, 64)	0
<hr/>		
dense_11 (Dense)	(None, 2)	130
<hr/>		
Total params: 4,258,914		
Trainable params: 4,258,914		
Non-trainable params: 0		

In [39]:

```
import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

Tue Apr 27 19:29:46 2021

In [40]:

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [41]:

```
model.fit(x_train,y_train_gender,
           validation_data=(x_valid, y_valid_gender),
           batch_size=64,
           steps_per_epoch=20,
           epochs=100,
           validation_steps=2,
           verbose=2,
           workers=1,
           use_multiprocessing = False,
           callbacks=[earlystop, plateau])
```

Epoch 1/100
20/20 - 39s - loss: 0.6611 - accuracy: 0.6719 - val_loss: 0.5868 - val_accuracy: 0.8047
Epoch 2/100
20/20 - 38s - loss: 0.5868 - accuracy: 0.7484 - val_loss: 0.5885 - val_accuracy: 0.8047
Epoch 3/100
20/20 - 37s - loss: 0.6163 - accuracy: 0.6945 - val_loss: 0.5519 - val_accuracy: 0.8047
Epoch 4/100
20/20 - 38s - loss: 0.6164 - accuracy: 0.6867 - val_loss: 0.5618 - val_accuracy: 0.8047
Epoch 5/100
20/20 - 38s - loss: 0.5915 - accuracy: 0.7188 - val_loss: 0.5316 - val_accuracy: 0.8047
Epoch 6/100
20/20 - 38s - loss: 0.5901 - accuracy: 0.7125 - val_loss: 0.5237 - val_accuracy: 0.8047
Epoch 7/100
20/20 - 37s - loss: 0.5978 - accuracy: 0.6891 - val_loss: 0.4999 - val_accuracy: 0.8047

Epoch 00007: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.
Epoch 8/100
20/20 - 38s - loss: 0.5595 - accuracy: 0.7180 - val_loss: 0.4900 - val_accuracy: 0.8047
Epoch 9/100
20/20 - 40s - loss: 0.5968 - accuracy: 0.6703 - val_loss: 0.4866 - val_accuracy: 0.8047
Epoch 10/100
20/20 - 39s - loss: 0.5571 - accuracy: 0.7094 - val_loss: 0.4749 - val_accuracy: 0.8047
Epoch 11/100
20/20 - 37s - loss: 0.5463 - accuracy: 0.7023 - val_loss: 0.4603 - val_accuracy: 0.8047
Epoch 12/100
20/20 - 37s - loss: 0.5425 - accuracy: 0.7234 - val_loss: 0.4582 - val_accuracy: 0.8047
Restoring model weights from the end of the best epoch.

Epoch 00012: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
Epoch 00012: early stopping

Out[41]: <tensorflow.python.keras.callbacks.History at 0x2342c20fe80>

In [42]:

```
# Evaluate the model on test set
score = model.evaluate(x_test, y_test_gender, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.591965913772583, 0.794909656047821]
```

In [43]:

```
y_hat = model.predict(x_test)

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test_gender[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                  color=("green" if predict_index == true_index else "red"))
plt.show()
```



In []:

Build the multilayer CNN (change dropout probabilities)

3-2. Two-layer CNN (2 fully connected layers)

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- kernel size = 2
- Dropout = 0.2 for covolutional layers

In [24]:

```
model = tf.keras.Sequential()

# Must define the input shape in the first Layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.2))
```

```

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.2))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(2, activation='sigmoid'))

# Take a look at the model summary
model.summary()

```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_2 (Conv2D)	(None, 128, 128, 64)	832
<hr/>		
max_pooling2d_2 (MaxPooling2D)	(None, 64, 64, 64)	0
<hr/>		
dropout_3 (Dropout)	(None, 64, 64, 64)	0
<hr/>		
conv2d_3 (Conv2D)	(None, 64, 64, 32)	8224
<hr/>		
max_pooling2d_3 (MaxPooling2D)	(None, 32, 32, 32)	0
<hr/>		
dropout_4 (Dropout)	(None, 32, 32, 32)	0
<hr/>		
flatten_1 (Flatten)	(None, 32768)	0
<hr/>		
dense_2 (Dense)	(None, 256)	8388864
<hr/>		
dropout_5 (Dropout)	(None, 256)	0
<hr/>		
dense_3 (Dense)	(None, 2)	514
<hr/>		
Total params: 8,398,434		
Trainable params: 8,398,434		
Non-trainable params: 0		

In [25]:

```

import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)

```

Tue Apr 27 21:03:08 2021

In [26]:

```

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

```

In [27]:

```

model.fit(x_train,y_train_gender,
           validation_data=(x_valid, y_valid_gender),
           batch_size=64,
           steps_per_epoch=20,
           epochs=100,
           validation_steps=2,
           verbose=2,
           workers=1,

```

```
use_multiprocessing = False,  
callbacks=[earlystop, plateau])
```

Epoch 1/100
20/20 - 21s - loss: 0.9006 - accuracy: 0.6586 - val_loss: 0.6674 - val_accuracy: 0.8047
Epoch 2/100
20/20 - 19s - loss: 0.6396 - accuracy: 0.7102 - val_loss: 0.6689 - val_accuracy: 0.8047
Epoch 3/100
20/20 - 20s - loss: 0.6264 - accuracy: 0.6953 - val_loss: 0.6392 - val_accuracy: 0.8047
Epoch 4/100
20/20 - 20s - loss: 0.6012 - accuracy: 0.7180 - val_loss: 0.6160 - val_accuracy: 0.8047
Epoch 5/100
20/20 - 19s - loss: 0.5923 - accuracy: 0.7000 - val_loss: 0.5519 - val_accuracy: 0.8047
Epoch 6/100
20/20 - 20s - loss: 0.5815 - accuracy: 0.7156 - val_loss: 0.5375 - val_accuracy: 0.8047
Epoch 7/100
20/20 - 20s - loss: 0.5460 - accuracy: 0.7305 - val_loss: 0.5329 - val_accuracy: 0.8047
Epoch 8/100
20/20 - 20s - loss: 0.5631 - accuracy: 0.7047 - val_loss: 0.4991 - val_accuracy: 0.8047
Epoch 9/100
20/20 - 20s - loss: 0.5392 - accuracy: 0.7102 - val_loss: 0.4815 - val_accuracy: 0.8047
Epoch 10/100
20/20 - 21s - loss: 0.5171 - accuracy: 0.7414 - val_loss: 0.4518 - val_accuracy: 0.8047
Epoch 11/100
20/20 - 20s - loss: 0.4973 - accuracy: 0.7352 - val_loss: 0.4437 - val_accuracy: 0.8047
Epoch 12/100
20/20 - 19s - loss: 0.5373 - accuracy: 0.7219 - val_loss: 0.4330 - val_accuracy: 0.8047
Epoch 13/100
20/20 - 19s - loss: 0.4836 - accuracy: 0.7586 - val_loss: 0.4473 - val_accuracy: 0.8047
Epoch 14/100
20/20 - 19s - loss: 0.4906 - accuracy: 0.7320 - val_loss: 0.4120 - val_accuracy: 0.8438
Epoch 15/100
20/20 - 19s - loss: 0.4627 - accuracy: 0.7758 - val_loss: 0.4031 - val_accuracy: 0.8047
Epoch 16/100
20/20 - 19s - loss: 0.4607 - accuracy: 0.7609 - val_loss: 0.4186 - val_accuracy: 0.8047
Epoch 17/100
20/20 - 19s - loss: 0.4318 - accuracy: 0.7918 - val_loss: 0.4115 - val_accuracy: 0.7969
Epoch 18/100
20/20 - 19s - loss: 0.4487 - accuracy: 0.7891 - val_loss: 0.3886 - val_accuracy: 0.8125
Epoch 19/100
20/20 - 19s - loss: 0.4296 - accuracy: 0.7945 - val_loss: 0.3917 - val_accuracy: 0.8203
Epoch 20/100
20/20 - 19s - loss: 0.4291 - accuracy: 0.8062 - val_loss: 0.3904 - val_accuracy: 0.8203
Epoch 21/100
20/20 - 19s - loss: 0.4380 - accuracy: 0.8047 - val_loss: 0.4368 - val_accuracy: 0.8281
Epoch 22/100
20/20 - 19s - loss: 0.4176 - accuracy: 0.8164 - val_loss: 0.3974 - val_accuracy: 0.8281
Epoch 23/100
20/20 - 19s - loss: 0.4291 - accuracy: 0.8219 - val_loss: 0.3995 - val_accuracy: 0.8281
Epoch 24/100
20/20 - 19s - loss: 0.4085 - accuracy: 0.8242 - val_loss: 0.3796 - val_accuracy: 0.8359
Epoch 25/100
20/20 - 19s - loss: 0.3979 - accuracy: 0.8180 - val_loss: 0.3892 - val_accuracy: 0.8281
Epoch 26/100
20/20 - 19s - loss: 0.4068 - accuracy: 0.8266 - val_loss: 0.3954 - val_accuracy: 0.8516
Epoch 27/100
20/20 - 19s - loss: 0.4346 - accuracy: 0.8094 - val_loss: 0.3799 - val_accuracy: 0.8203
Epoch 28/100
20/20 - 19s - loss: 0.4255 - accuracy: 0.8203 - val_loss: 0.3635 - val_accuracy: 0.8359
Epoch 29/100
20/20 - 19s - loss: 0.3983 - accuracy: 0.8219 - val_loss: 0.3769 - val_accuracy: 0.8438
Epoch 30/100
20/20 - 19s - loss: 0.4195 - accuracy: 0.8242 - val_loss: 0.3843 - val_accuracy: 0.8438
Epoch 31/100

```
20/20 - 19s - loss: 0.4108 - accuracy: 0.8242 - val_loss: 0.3977 - val_accuracy: 0.8281
Epoch 00031: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.
Epoch 32/100
20/20 - 19s - loss: 0.3940 - accuracy: 0.8305 - val_loss: 0.3861 - val_accuracy: 0.8281
Epoch 33/100
20/20 - 19s - loss: 0.4298 - accuracy: 0.8234 - val_loss: 0.3857 - val_accuracy: 0.8359
Epoch 34/100
20/20 - 19s - loss: 0.3889 - accuracy: 0.8352 - val_loss: 0.3802 - val_accuracy: 0.8516
Epoch 35/100
20/20 - 19s - loss: 0.3656 - accuracy: 0.8656 - val_loss: 0.3713 - val_accuracy: 0.8359
Epoch 36/100
20/20 - 19s - loss: 0.3529 - accuracy: 0.8602 - val_loss: 0.3702 - val_accuracy: 0.8516
Epoch 37/100
20/20 - 19s - loss: 0.4021 - accuracy: 0.8422 - val_loss: 0.3723 - val_accuracy: 0.8438
Epoch 38/100
20/20 - 19s - loss: 0.3747 - accuracy: 0.8523 - val_loss: 0.3740 - val_accuracy: 0.8438
Epoch 39/100
20/20 - 19s - loss: 0.3806 - accuracy: 0.8359 - val_loss: 0.3780 - val_accuracy: 0.8359
Epoch 40/100
20/20 - 19s - loss: 0.3814 - accuracy: 0.8461 - val_loss: 0.3784 - val_accuracy: 0.8359

Epoch 00040: ReduceLROnPlateau reducing learning rate to 1.000000474974514e-05.
Epoch 41/100
20/20 - 19s - loss: 0.3602 - accuracy: 0.8687 - val_loss: 0.3778 - val_accuracy: 0.8359
Epoch 42/100
20/20 - 19s - loss: 0.3722 - accuracy: 0.8445 - val_loss: 0.3773 - val_accuracy: 0.8359
Epoch 43/100
20/20 - 19s - loss: 0.3676 - accuracy: 0.8602 - val_loss: 0.3767 - val_accuracy: 0.8359
Epoch 44/100
20/20 - 19s - loss: 0.3963 - accuracy: 0.8352 - val_loss: 0.3764 - val_accuracy: 0.8359
Epoch 45/100
20/20 - 19s - loss: 0.3692 - accuracy: 0.8383 - val_loss: 0.3762 - val_accuracy: 0.8359
Epoch 46/100
20/20 - 19s - loss: 0.3913 - accuracy: 0.8383 - val_loss: 0.3756 - val_accuracy: 0.8359

Epoch 00046: ReduceLROnPlateau reducing learning rate to 1.000000656873453e-06.
Epoch 47/100
20/20 - 19s - loss: 0.3753 - accuracy: 0.8391 - val_loss: 0.3756 - val_accuracy: 0.8359
Epoch 48/100
20/20 - 19s - loss: 0.3508 - accuracy: 0.8492 - val_loss: 0.3755 - val_accuracy: 0.8359
Epoch 49/100
20/20 - 19s - loss: 0.3522 - accuracy: 0.8531 - val_loss: 0.3755 - val_accuracy: 0.8359
Epoch 50/100
20/20 - 19s - loss: 0.3447 - accuracy: 0.8508 - val_loss: 0.3754 - val_accuracy: 0.8359
Epoch 51/100
20/20 - 19s - loss: 0.3682 - accuracy: 0.8502 - val_loss: 0.3754 - val_accuracy: 0.8359
Restoring model weights from the end of the best epoch.

Epoch 00051: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.
Epoch 00051: early stopping
```

Out[27]: <tensorflow.python.keras.callbacks.History at 0x2c22a392730>

```
In [28]: # Evaluate the model on test set
score = model.evaluate(x_test, y_test_gender, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

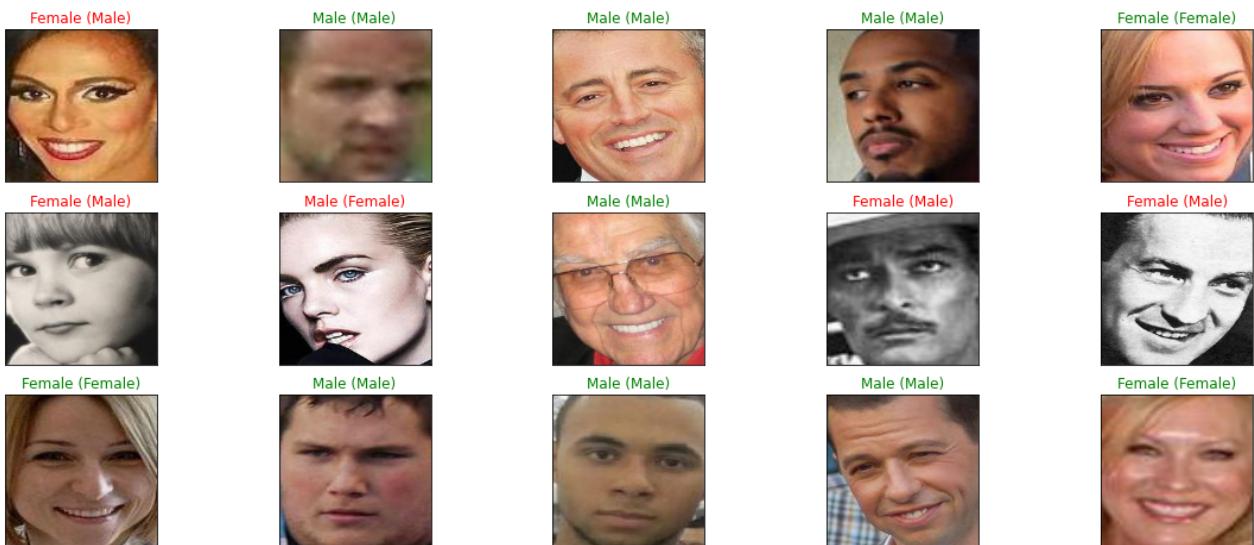
Test accuracy: [0.3691970109939575, 0.8561416268348694]

```
In [29]: labels =["Female", # index 0
```

```
"Male",      # index 1  
]
```

In [30]:

```
y_hat = model.predict(x_test)  
  
# Plot a random sample of 10 test images, their predicted labels and ground truth  
figure = plt.figure(figsize=(20, 8))  
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):  
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks=[])  
    # Display each image  
    ax.imshow(np.squeeze(x_test[index]))  
    predict_index = np.argmax(y_hat[index])  
    true_index = np.argmax(y_test_gender[index])  
    # Set the title for each image  
    ax.set_title("{} ({})".format(labels[predict_index],  
                                  labels[true_index]),  
                color=("green" if predict_index == true_index else "red"))  
plt.show()
```



Build the multilayer CNN (change dropout probabilities)

3-3. Two-layer CNN (2 fully connected layers)

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- kernel size = 2
- Dropout = 0.1 for covolutional layers

In [31]:

```
model = tf.keras.Sequential()  
  
# Must define the input shape in the first Layer of the neural network  
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu'))  
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))  
model.add(tf.keras.layers.Dropout(0.1))  
  
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))  
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
```

```

model.add(tf.keras.layers.Dropout(0.1))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(2, activation='sigmoid'))

# Take a Look at the model summary
model.summary()

```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_4 (Conv2D)	(None, 128, 128, 64)	832
<hr/>		
max_pooling2d_4 (MaxPooling2D)	(None, 64, 64, 64)	0
<hr/>		
dropout_6 (Dropout)	(None, 64, 64, 64)	0
<hr/>		
conv2d_5 (Conv2D)	(None, 64, 64, 32)	8224
<hr/>		
max_pooling2d_5 (MaxPooling2D)	(None, 32, 32, 32)	0
<hr/>		
dropout_7 (Dropout)	(None, 32, 32, 32)	0
<hr/>		
flatten_2 (Flatten)	(None, 32768)	0
<hr/>		
dense_4 (Dense)	(None, 256)	8388864
<hr/>		
dropout_8 (Dropout)	(None, 256)	0
<hr/>		
dense_5 (Dense)	(None, 2)	514
<hr/>		
Total params: 8,398,434		
Trainable params: 8,398,434		
Non-trainable params: 0		

In [32]:

```

import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)

```

Tue Apr 27 21:19:44 2021

In [33]:

```

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

```

In [37]:

```

model.fit(x_train,y_train_gender,
           validation_data=(x_valid, y_valid_gender),
           batch_size=64,
           steps_per_epoch=20,
           epochs=100,
           validation_steps=2,
           verbose=2,
           workers=1,
           use_multiprocessing = False,
           callbacks=[earlystop, plateau])

```

Epoch 1/100
20/20 - 20s - loss: 0.7090 - accuracy: 0.6344 - val_loss: 0.6260 - val_accuracy: 0.8047
Epoch 2/100
20/20 - 19s - loss: 0.5894 - accuracy: 0.7211 - val_loss: 0.5499 - val_accuracy: 0.8047
Epoch 3/100
20/20 - 20s - loss: 0.5618 - accuracy: 0.7242 - val_loss: 0.4791 - val_accuracy: 0.7969
Epoch 4/100
20/20 - 19s - loss: 0.5469 - accuracy: 0.7570 - val_loss: 0.5029 - val_accuracy: 0.8125
Epoch 5/100
20/20 - 20s - loss: 0.5145 - accuracy: 0.7664 - val_loss: 0.4511 - val_accuracy: 0.7969
Epoch 6/100
20/20 - 20s - loss: 0.4606 - accuracy: 0.7984 - val_loss: 0.4362 - val_accuracy: 0.8203
Epoch 7/100
20/20 - 20s - loss: 0.4461 - accuracy: 0.8000 - val_loss: 0.4093 - val_accuracy: 0.8203
Epoch 8/100
20/20 - 20s - loss: 0.4351 - accuracy: 0.8016 - val_loss: 0.4147 - val_accuracy: 0.8281
Epoch 9/100
20/20 - 20s - loss: 0.4092 - accuracy: 0.8305 - val_loss: 0.4370 - val_accuracy: 0.7891
Epoch 10/100
20/20 - 20s - loss: 0.3843 - accuracy: 0.8461 - val_loss: 0.3979 - val_accuracy: 0.8125
Epoch 11/100
20/20 - 19s - loss: 0.3882 - accuracy: 0.8305 - val_loss: 0.4051 - val_accuracy: 0.8359
Epoch 12/100
20/20 - 20s - loss: 0.4288 - accuracy: 0.8234 - val_loss: 0.4286 - val_accuracy: 0.8438
Epoch 13/100
20/20 - 20s - loss: 0.4259 - accuracy: 0.8078 - val_loss: 0.3853 - val_accuracy: 0.8516
Epoch 14/100
20/20 - 20s - loss: 0.3852 - accuracy: 0.8344 - val_loss: 0.4191 - val_accuracy: 0.8125
Epoch 15/100
20/20 - 20s - loss: 0.4007 - accuracy: 0.8172 - val_loss: 0.3670 - val_accuracy: 0.8516

Epoch 00015: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.

Epoch 16/100
20/20 - 20s - loss: 0.4023 - accuracy: 0.8242 - val_loss: 0.3816 - val_accuracy: 0.8438
Epoch 17/100
20/20 - 20s - loss: 0.3535 - accuracy: 0.8415 - val_loss: 0.3777 - val_accuracy: 0.8516
Epoch 18/100
20/20 - 19s - loss: 0.3569 - accuracy: 0.8477 - val_loss: 0.3849 - val_accuracy: 0.8359
Epoch 19/100
20/20 - 20s - loss: 0.3765 - accuracy: 0.8422 - val_loss: 0.3852 - val_accuracy: 0.8594
Epoch 20/100
20/20 - 20s - loss: 0.3457 - accuracy: 0.8586 - val_loss: 0.3755 - val_accuracy: 0.8438
Epoch 21/100
20/20 - 20s - loss: 0.3440 - accuracy: 0.8594 - val_loss: 0.3800 - val_accuracy: 0.8672
Epoch 22/100
20/20 - 20s - loss: 0.3156 - accuracy: 0.8828 - val_loss: 0.3724 - val_accuracy: 0.8438
Epoch 23/100
20/20 - 20s - loss: 0.3457 - accuracy: 0.8578 - val_loss: 0.3801 - val_accuracy: 0.8672
Epoch 24/100
20/20 - 20s - loss: 0.3573 - accuracy: 0.8430 - val_loss: 0.3685 - val_accuracy: 0.8516
Epoch 25/100
20/20 - 20s - loss: 0.3340 - accuracy: 0.8656 - val_loss: 0.3771 - val_accuracy: 0.8594
Epoch 26/100
20/20 - 20s - loss: 0.3351 - accuracy: 0.8594 - val_loss: 0.3832 - val_accuracy: 0.8516
Epoch 27/100
20/20 - 20s - loss: 0.3323 - accuracy: 0.8625 - val_loss: 0.3815 - val_accuracy: 0.8516

Epoch 00027: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.

Epoch 28/100
20/20 - 20s - loss: 0.3392 - accuracy: 0.8570 - val_loss: 0.3796 - val_accuracy: 0.8516
Epoch 29/100
20/20 - 20s - loss: 0.3353 - accuracy: 0.8602 - val_loss: 0.3796 - val_accuracy: 0.8516
Epoch 30/100
20/20 - 20s - loss: 0.3565 - accuracy: 0.8461 - val_loss: 0.3799 - val_accuracy: 0.8594
Epoch 31/100

```
20/20 - 20s - loss: 0.3192 - accuracy: 0.8680 - val_loss: 0.3796 - val_accuracy: 0.8594
Epoch 32/100
20/20 - 19s - loss: 0.3285 - accuracy: 0.8727 - val_loss: 0.3788 - val_accuracy: 0.8516
Restoring model weights from the end of the best epoch.
```

```
Epoch 00032: ReduceLROnPlateau reducing learning rate to 1.000000656873453e-06.
Epoch 00032: early stopping
```

```
Out[37]: <tensorflow.python.keras.callbacks.History at 0x2c22d9d0ca0>
```

```
In [38]: # Evaluate the model on test set
```

```
score = model.evaluate(x_test, y_test_gender, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.34170055389404297, 0.8590925931930542]
```

```
In [39]: labels =["Female", # index 0
            "Male",      # index 1
            ]
```

```
In [42]: y_hat = model.predict(x_test)
```

```
# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x_test[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test_gender[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                color=("green" if predict_index == true_index else "red"))
plt.show()
```



Build the multilayer CNN (change dropout probabilities)

3-3. Two-layer CNN (2 fully connected layers)

- Using Activation Function "ReLU"
- Using Maxing Pooling method
- kernel size = 2
- Dropout = 0.5 for convolutional layers

```
In [53]: model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.5))

model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
model.add(tf.keras.layers.Dropout(0.5))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(2, activation='sigmoid'))

# Take a Look at the model summary
model.summary()
```

Model: "sequential_5"

Layer (type)	Output Shape	Param #
<hr/>		
conv2d_10 (Conv2D)	(None, 128, 128, 64)	832
<hr/>		
max_pooling2d_10 (MaxPooling)	(None, 64, 64, 64)	0
<hr/>		
dropout_15 (Dropout)	(None, 64, 64, 64)	0
<hr/>		
conv2d_11 (Conv2D)	(None, 64, 64, 32)	8224
<hr/>		
max_pooling2d_11 (MaxPooling)	(None, 32, 32, 32)	0
<hr/>		
dropout_16 (Dropout)	(None, 32, 32, 32)	0
<hr/>		
flatten_5 (Flatten)	(None, 32768)	0
<hr/>		
dense_10 (Dense)	(None, 256)	8388864
<hr/>		
dropout_17 (Dropout)	(None, 256)	0
<hr/>		
dense_11 (Dense)	(None, 2)	514
<hr/>		
Total params: 8,398,434		
Trainable params: 8,398,434		
Non-trainable params: 0		

```
In [54]: import time
localtime = time.asctime( time.localtime(time.time()) )
print(localtime)
```

Wed Apr 28 02:41:17 2021

In [55]:

```
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=[ 'accuracy'])
```

In [56]:

```
model.fit(x_train,y_train_gender,
           validation_data=(x_valid, y_valid_gender),
           batch_size=64,
           steps_per_epoch=20,
           epochs=100,
           validation_steps=2,
           verbose=2,
           workers=1,
           use_multiprocessing = False,
           callbacks=[earlystop, plateau])
```

Epoch 1/100
20/20 - 103s - loss: 1.1738 - accuracy: 0.6672 - val_loss: 0.6906 - val_accuracy: 0.8047
Epoch 2/100
20/20 - 90s - loss: 0.6193 - accuracy: 0.7094 - val_loss: 0.6842 - val_accuracy: 0.8047
Epoch 3/100
20/20 - 108s - loss: 0.5940 - accuracy: 0.7102 - val_loss: 0.6807 - val_accuracy: 0.8047
Epoch 4/100
20/20 - 89s - loss: 0.5996 - accuracy: 0.7125 - val_loss: 0.6709 - val_accuracy: 0.8047
Epoch 5/100
20/20 - 100s - loss: 0.5847 - accuracy: 0.7258 - val_loss: 0.6353 - val_accuracy: 0.8047
Epoch 6/100
20/20 - 106s - loss: 0.5375 - accuracy: 0.7531 - val_loss: 0.6018 - val_accuracy: 0.8047
Epoch 7/100
20/20 - 68s - loss: 0.5591 - accuracy: 0.7484 - val_loss: 0.6132 - val_accuracy: 0.7891
Epoch 8/100
20/20 - 87s - loss: 0.5649 - accuracy: 0.7203 - val_loss: 0.6195 - val_accuracy: 0.8125
Epoch 9/100
20/20 - 68s - loss: 0.5500 - accuracy: 0.7547 - val_loss: 0.6060 - val_accuracy: 0.8047
Epoch 10/100
20/20 - 110s - loss: 0.5041 - accuracy: 0.7680 - val_loss: 0.5430 - val_accuracy: 0.8125
Epoch 11/100
20/20 - 95s - loss: 0.5337 - accuracy: 0.7539 - val_loss: 0.5548 - val_accuracy: 0.8125
Epoch 12/100
20/20 - 84s - loss: 0.5090 - accuracy: 0.7727 - val_loss: 0.5473 - val_accuracy: 0.8125
Epoch 13/100
20/20 - 89s - loss: 0.4961 - accuracy: 0.7688 - val_loss: 0.5260 - val_accuracy: 0.8047
Epoch 14/100
20/20 - 75s - loss: 0.4789 - accuracy: 0.7883 - val_loss: 0.5338 - val_accuracy: 0.8203
Epoch 15/100
20/20 - 90s - loss: 0.4990 - accuracy: 0.7641 - val_loss: 0.5333 - val_accuracy: 0.7969
Epoch 16/100
20/20 - 106s - loss: 0.4844 - accuracy: 0.7727 - val_loss: 0.5000 - val_accuracy: 0.8281
Epoch 17/100
20/20 - 77s - loss: 0.4733 - accuracy: 0.7689 - val_loss: 0.5135 - val_accuracy: 0.7969
Epoch 18/100
20/20 - 78s - loss: 0.4509 - accuracy: 0.8031 - val_loss: 0.4640 - val_accuracy: 0.8203
Epoch 19/100
20/20 - 87s - loss: 0.4919 - accuracy: 0.7805 - val_loss: 0.4674 - val_accuracy: 0.8125
Epoch 20/100
20/20 - 77s - loss: 0.4525 - accuracy: 0.7844 - val_loss: 0.4569 - val_accuracy: 0.8125
Epoch 21/100
20/20 - 66s - loss: 0.4504 - accuracy: 0.8086 - val_loss: 0.4415 - val_accuracy: 0.8125
Epoch 22/100
20/20 - 76s - loss: 0.4230 - accuracy: 0.8156 - val_loss: 0.4586 - val_accuracy: 0.8047
Epoch 23/100

```
20/20 - 91s - loss: 0.4270 - accuracy: 0.8070 - val_loss: 0.4463 - val_accuracy: 0.8203
Epoch 24/100
20/20 - 105s - loss: 0.4244 - accuracy: 0.8211 - val_loss: 0.4469 - val_accuracy: 0.8203
Epoch 25/100
20/20 - 75s - loss: 0.4230 - accuracy: 0.8258 - val_loss: 0.4481 - val_accuracy: 0.8047
Epoch 26/100
20/20 - 83s - loss: 0.4225 - accuracy: 0.8102 - val_loss: 0.4309 - val_accuracy: 0.8125
Epoch 27/100
20/20 - 115s - loss: 0.4049 - accuracy: 0.8219 - val_loss: 0.4362 - val_accuracy: 0.8125
Epoch 28/100
20/20 - 88s - loss: 0.4119 - accuracy: 0.8172 - val_loss: 0.4418 - val_accuracy: 0.8203
Epoch 29/100
20/20 - 57s - loss: 0.4122 - accuracy: 0.8242 - val_loss: 0.4252 - val_accuracy: 0.8125
Epoch 30/100
20/20 - 62s - loss: 0.4049 - accuracy: 0.8102 - val_loss: 0.4119 - val_accuracy: 0.8203

Epoch 00030: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.
Epoch 31/100
20/20 - 74s - loss: 0.4022 - accuracy: 0.8313 - val_loss: 0.4077 - val_accuracy: 0.8203
Epoch 32/100
20/20 - 89s - loss: 0.3710 - accuracy: 0.8414 - val_loss: 0.4033 - val_accuracy: 0.8281
Epoch 33/100
20/20 - 82s - loss: 0.3830 - accuracy: 0.8453 - val_loss: 0.4177 - val_accuracy: 0.8125
Epoch 34/100
20/20 - 67s - loss: 0.3656 - accuracy: 0.8494 - val_loss: 0.4189 - val_accuracy: 0.8125
Epoch 35/100
20/20 - 45s - loss: 0.3847 - accuracy: 0.8578 - val_loss: 0.4190 - val_accuracy: 0.8125
Epoch 36/100
20/20 - 46s - loss: 0.3672 - accuracy: 0.8438 - val_loss: 0.4157 - val_accuracy: 0.8203
Epoch 37/100
20/20 - 48s - loss: 0.3677 - accuracy: 0.8508 - val_loss: 0.4124 - val_accuracy: 0.8203
Epoch 38/100
20/20 - 45s - loss: 0.3341 - accuracy: 0.8562 - val_loss: 0.4079 - val_accuracy: 0.8203
Epoch 39/100
20/20 - 49s - loss: 0.3574 - accuracy: 0.8414 - val_loss: 0.3990 - val_accuracy: 0.8281
Epoch 40/100
20/20 - 50s - loss: 0.3671 - accuracy: 0.8453 - val_loss: 0.4030 - val_accuracy: 0.8125

Epoch 00040: ReduceLROnPlateau reducing learning rate to 1.000000474974514e-05.
Epoch 41/100
20/20 - 50s - loss: 0.3722 - accuracy: 0.8492 - val_loss: 0.4047 - val_accuracy: 0.8125
Epoch 42/100
20/20 - 57s - loss: 0.3849 - accuracy: 0.8391 - val_loss: 0.4057 - val_accuracy: 0.8125
Epoch 43/100
20/20 - 51s - loss: 0.3623 - accuracy: 0.8398 - val_loss: 0.4060 - val_accuracy: 0.8203
Epoch 44/100
20/20 - 54s - loss: 0.4052 - accuracy: 0.8328 - val_loss: 0.4069 - val_accuracy: 0.8203
Epoch 45/100
20/20 - 83s - loss: 0.3649 - accuracy: 0.8531 - val_loss: 0.4077 - val_accuracy: 0.8203
Restoring model weights from the end of the best epoch.
```

```
Epoch 00045: ReduceLROnPlateau reducing learning rate to 1.000000656873453e-06.
Epoch 00045: early stopping
```

```
Out[56]: <tensorflow.python.keras.callbacks.History at 0x2c22d69c430>
```

```
In [57]:
```

```
# Evaluate the model on test set
score = model.evaluate(x_test, y_test_gender, verbose=0)

# Print test accuracy
print('\n', 'Test accuracy:', score)
```

```
Test accuracy: [0.4001685380935669, 0.8399114608764648]
```

```
In [58]: labels =["Female", # index 0  
           "Male",      # index 1  
          ]
```

```
In [60]: y_hat = model.predict(x_test)  
  
# Plot a random sample of 10 test images, their predicted labels and ground truth  
figure = plt.figure(figsize=(20, 8))  
for i, index in enumerate(np.random.choice(x_test.shape[0], size=15, replace=False)):  
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks=[])  
    # Display each image  
    ax.imshow(np.squeeze(x_test[index]))  
    predict_index = np.argmax(y_hat[index])  
    true_index = np.argmax(y_test_gender[index])  
    # Set the title for each image  
    ax.set_title("{} ({})".format(labels[predict_index],  
                                 labels[true_index]),  
                color=("green" if predict_index == true_index else "red"))  
plt.show()
```



Conclusion for the previous models

- Kernel size = 2
- Dropout rate = 0.3
- #Fully connected layers do not affect much, so still use two layers: 256, 5.
- For other fixed info:
 - Activation function: ReLU,
 - Pooling Methods: Max Pooling,
 - #Convolution layers = 2

```
In [ ]:
```

VGG-age

April 29, 2021

```
[1]: import numpy as np
import pandas as pd
import math
import cv2
import matplotlib.pyplot as plt
import os
import seaborn as sns
import umap
from PIL import Image
from scipy import misc
from os import listdir
from os.path import isfile, join
import numpy as np
from scipy import misc
from random import shuffle
from collections import Counter
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.utils.np_utils import to_categorical
```

```
[2]: import pandas as pd
imageinfo = pd.read_csv("wikifinal.csv")
```

```
[3]: len(imageinfo)
```

```
[3]: 27123
```

```
[5]: imageinfo[:5]
```

```
[5]:   AGE  GENDER          NAME        PATH
  0    27        1  Sami Jauhojärvi  1_crop.jpg
  1    59        1      Marc Okrand  2_crop.jpg
```

```

2   50      0   Krista Tippett  3_crop.jpg
3   32      1   Bernie Whitebear  4_crop.jpg
4   41      1   Carl Greenberg  5_crop.jpg

```

```
[17]: age_classes = []
Y_age = imageinfo["AGE"]
for i in Y_age:
    if i <= 18:
        age_classes.append("(0,18]")
    elif i <= 30:
        age_classes.append("(18,30]")
    elif i <= 40:
        age_classes.append("(30,40]")
    elif i <= 60:
        age_classes.append("(40,60]")
    elif i > 60:
        age_classes.append("60+")

```

```
[18]: class_label = ['(0,18)', '(18,30)', '(30,40)', '(40,60)', '60+']
```

```
[20]: len(age_classes)
```

```
[20]: 27123
```

```
[3]: train = pd.read_csv("train.csv")
validation_dir = pd.read_csv("valid.csv")
test_dir = pd.read_csv("test.csv")
```

```
[21]: len(imageinfo)
```

```
[21]: 27123
```

```
[22]: imageinfo['age_classes'] = age_classes
imageinfo.to_csv("wikifinal.csv", mode = 'a', index = False)

print(imageinfo)
```

	AGE	GENDER	NAME	PATH	age_classes
0	27	1	Sami Jauhojärvi	1_crop.jpg	(18,30]
1	59	1	Marc Okrand	2_crop.jpg	(40,60]
2	50	0	Krista Tippett	3_crop.jpg	(40,60]
3	32	1	Bernie Whitebear	4_crop.jpg	(30,40]
4	41	1	Carl Greenberg	5_crop.jpg	(40,60]
...
27118	29	1	Manuel Velázquez	27155_crop.jpg	(18,30]
27119	22	1	Pachán	27156_crop.jpg	(18,30]
27120	30	1	Coen Moulijn	27157_crop.jpg	(18,30]

```
27121 23      1  Theo van Duivenbode  27158_crop.jpg      (18,30]
27122 39      1    Michael Wiesinger  27159_crop.jpg      (30,40]
```

[27123 rows x 5 columns]

```
[4]: import os
import shutil
```

```
[5]: vgg16_with_top = VGG16(include_top=True, weights='imagenet',
                           input_tensor=None, input_shape=None,
                           pooling=None,
                           classes=1000)
vgg16_with_top.summary()
```

```
[17]: def make_age_data(data):

    count = 0
    num_file = 2000
    i = 0
    while True:
        writer = tf.python_io.TFRecordWriter('age' + str(i) + '.tfrecords')
        for j in range(num_file):
            if count == len(data):
                break
            label = data[count][1]
            img = cv2.imread(data[count][0])
            img = cv2.resize(img, (img_size, img_size))
            if img is not None:
                image_raw = img.tostring()
                feature = {'label': tf.train.Feature(int64_list=tf.train.
                                          Int64List(value=[label])),
                           'image_raw': tf.train.Feature(bytes_list=tf.train.
                                          BytesList(value=[image_raw]))}
                example = tf.train.Example(features=tf.train.
                                           Features(feature=feature))
                writer.write(example.SerializeToString())
            count += 1
        i += 1
        if count == len(data):
            break
```

```
[18]: # cite from the paper (see reference in report)
import tensorflow as tf
import numpy as np
import os
import time
import cv2
```

```

import shutil
import datetime
import vgg
import cnn_model

VGG_FACE_MODEL_WEIGHT = './face.weight'

# reading tf record data

class DataReader:
    def __init__(self, data_dir, batch_size, num_epochs, is_training):
        files = os.listdir(data_dir)
        files = map(lambda x: os.path.join(data_dir, x), files)
        self.file_names = list(files)
        self.batch_size = batch_size
        self.num_epochs = num_epochs
        self.is_training = is_training

    def input(self):
        file_queue = tf.train.string_input_producer(self.file_names, num_epochs=self.num_epochs)
        image, label = self.read_and_decode(file_queue)

        if self.is_training:
            image_batch, label_batch = tf.train.shuffle_batch([image, label], batch_size=self.batch_size)

        else:
            image_batch, label_batch = tf.train.batch([image, label], batch_size=self.batch_size,)

        return image_batch, tf.reshape(label_batch, [self.batch_size])

    def read_and_decode(self, file_queue):
        reader = tf.TFRecordReader()
        _, serialized_example = reader.read(file_queue)
        features = tf.parse_single_example(
            serialized_example,
            features={
                'image_raw': tf.FixedLenFeature([], tf.string),
                'label': tf.FixedLenFeature([], tf.int64),
            })
        # label
        label = tf.cast(features['label'], tf.int32)

        # image data

```

```

    image = tf.decode_raw(features['imageinfo'], tf.uint8)
    image = tf.reshape(image, [FLAGS.image_size, FLAGS.image_size, FLAGS.
    ↪image_channel])

    image = self.preprocess(image)

    return image, label


def train():
    # train data batch
    train_data_reader = DataReader(FLAGS.train_data_dir, FLAGS.batch_size, □
    ↪FLAGS.epoch_num, True)
    train_images, train_labels = train_data_reader.input()

    # validation data batch
    val_data_reader = DataReader(FLAGS.val_data_dir, FLAGS.val_batch_size, □
    ↪None, False)
    val_images, val_labels = val_data_reader.input()

    # place holder
    image_batch = tf.placeholder(dtype=tf.float32, shape=[None, FLAGS.
    ↪input_size, FLAGS.input_size, FLAGS.image_channel],
                                 name='image_batch')
    label_batch = tf.placeholder(dtype=tf.int32, name='label_batch')
    training_or_not = tf.placeholder(dtype=tf.bool, name='training_or_not')

    # build model
    vggface = vgg.VGGFace(VGG_FACE_MODEL_WEIGHT)
    y_result = vggface.build(image_batch, training_or_not, FLAGS.class_num)

    # loss and accuracy
    loss = tf.reduce_mean(tf.nn.
    ↪sparse_softmax_cross_entropy_with_logits(labels=label_batch, □
    ↪logits=y_result),
                           name='loss')
    reg_loss = tf.reduce_sum(tf.get_collection(tf.GraphKeys.
    ↪REGULARIZATION_LOSSES))
    loss += reg_loss
    labels_l = tf.cast(label_batch, tf.int64)
    accuracy = tf.reduce_mean(tf.cast(tf.equal(tf.argmax(y_result, 1), □
    ↪labels_l), tf.float32), name='accuracy')

    # train method
    update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)

```

```

    with tf.control_dependencies(update_ops):
        train_method = tf.train.AdamOptimizer(learning_rate=0.005).
        ↪minimize(loss)

    # train summary op
    train_summary_op = tf.summary.merge([tf.summary.scalar('loss', loss),
                                         tf.summary.scalar('accuracy', ↪
                                         ↪accuracy)])]

    # validation summary op
    val_loss_pl = tf.placeholder(tf.float32, name='val_loss_pl')
    val_accuracy_pl = tf.placeholder(tf.float32, name='val_accuracy_pl')
    val_summary_op = tf.summary.merge([tf.summary.scalar('loss', val_loss_pl),
                                       tf.summary.scalar('accuracy', ↪
                                         ↪val_accuracy_pl)])]

    # session
    gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.9)
    sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options))

    # The op for initializing the variables.
    init_op = tf.group(tf.global_variables_initializer(), tf.
    ↪local_variables_initializer())
    sess.run(init_op)

    print('total para: ', np.sum([np.prod(v.get_shape().as_list()) for v in tf.
    ↪trainable_variables()]))]

    # modele saver
    saver = tf.train.Saver(max_to_keep=100)

    if not os.path.exists(FLAGS.model_path):
        os.mkdir(FLAGS.model_path)
    else:
        shutil.rmtree(FLAGS.model_path)

    try:
        while not coord.should_stop():
            start_time = time.time()

            # images and label
            train_images_batch, train_labels_batch = sess.run([train_images, ↪
            ↪train_labels])
            feed_dict = {image_batch: train_images_batch, label_batch: ↪
            ↪train_labels_batch}

```

```

# for test
if step == 0:
    output_images(train_images_batch, '7011')

# back propagation and update params
feed_dict[training_or_not] = True
sess.run(train_method, feed_dict=feed_dict)

# save model
if step % steps_one_epoch == 0:
    saver.save(sess, FLAGS.model_path + FLAGS.model_name, ↴
↪global_step=step)

    n = 10
    mean_loss = 0.0
    mean_accuracy = 0.0
    for i in range(n):
        val_images_batch, val_labels_batch = sess.run([val_images, ↴
↪val_labels])

        val_images_batch = val_images_batch[:, 14:14 + 192, 14:14 + ↴
↪192, :]
        feed_dict[image_batch] = val_images_batch
        feed_dict[label_batch] = val_labels_batch
        feed_dict[training_or_not] = False
        loss_val, accuracy_val = sess.run([loss, accuracy], ↴
↪feed_dict=feed_dict)
        mean_loss += loss_val
        mean_accuracy += accuracy_val

        mean_loss /= n
        mean_accuracy /= n
        val_summary = sess.run(val_summary_op, feed_dict={val_loss_pl: ↴
↪mean_loss,
↪val_accuracy_pl: mean_accuracy})
        val_writer.add_summary(val_summary, step)

        step += 1
sess.close()
train_end_time = datetime.datetime.now()

def test(model_name):

```

```

# os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
tf.reset_default_graph()

# validation data batch
val_data_reader = DataReader(FLAGS.val_data_dir, FLAGS.val_batch_size, 1, ↴
↪False)
val_images, val_labels = val_data_reader.input()

# session
gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.8)
sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options))

# The op for initializing the variables.
init_op = tf.group(tf.global_variables_initializer(), tf.
↪local_variables_initializer())
sess.run(init_op)

# restore model
saver = tf.train.import_meta_graph(model_name+'.meta')
saver.restore(sess, model_name)

graph = tf.get_default_graph()
image_batch = graph.get_tensor_by_name('image_batch:0')
label_batch = graph.get_tensor_by_name('label_batch:0')
training_or_not = graph.get_tensor_by_name('training_or_not:0')
predict_label = graph.get_tensor_by_name('softmax:0')

right_total = 0
data_total = 0

# start input enqueue threads.
coord = tf.train.Coordinator()
threads = tf.train.start_queue_runners(sess=sess, coord=coord)

try:
    while not coord.should_stop():

        # image and label
        val_images_batch, val_labels_batch = sess.run([val_images, ↴
↪val_labels])

        # if using multi crop to cal accuracy
        if not FLAGS.multi_crop:
            image = val_images_batch[:, 14:14+192, 14:14+192, :]
            result = sess.run(predict_label, feed_dict={image_batch: image,
                                                        label_batch: ↴
↪val_labels_batch,

```

```

    training_or_not:False})
        label = np.argmax(result, axis=1)
    else:
        label = 0
        croped_images_batch = multi_crop(val_images_batch)
        for images_batch in croped_images_batch:
            result = sess.run(predict_label, feed_dict={image_batch:images_batch,
                                             val_labels_batch,
                                             training_or_not:False})
            label += result
        label = np.argmax(label, axis=1)

        right = np.sum(np.equal(label, val_labels_batch))
        right_total += right
        data_total += len(val_labels_batch)

    except tf.errors.OutOfRangeError:
        pass
    finally:
        # When done, ask the threads to stop.
        coord.request_stop()

```

```

def inference(imageinfo):

    tf.reset_default_graph()

    # session
    sess = tf.Session()

    # restore model
    model_name = os.path.join(FLAGS.model_path, 'model.ckpt-3934')
    saver = tf.train.import_meta_graph(model_name + '.meta')
    saver.restore(sess, model_name)

    # useful tensors
    graph = tf.get_default_graph()
    image_batch = graph.get_tensor_by_name('image_batch:0')
    label_batch = graph.get_tensor_by_name('label_batch:0')
    training_or_not = graph.get_tensor_by_name('training_or_not:0')
    predict_label = graph.get_tensor_by_name('softmax:0')

```

```

def main(_):
    if FLAGS.run_type == 0:
        train()
    elif FLAGS.run_type == 1:
        test_all_saved_model()
    elif FLAGS.run_type == 2:
        model_name = ''
        test(model_name)
    else:
        image = cv2.imread(FLAGS.test_pic)
        inference(image)

if __name__ == '__main__':
    tf.app.run()

```

```

[19]: # model init
import tensorflow as tf
import pickle
import torch
import source.models as models
import source.worker as worker
import source.loader as loader

class VGGFace:
    def __init__(self, weight_path):
        self.weight = None
        with open(weight_path, 'rb') as f:
            self.weight = pickle.load(f)

    def build(self, image, is_training, output_unit):
        with tf.variable_scope('vgg_face'):
            conv1_1 = self.conv_layer(image, 64, 'conv1_1', self.weight[0], ↴
                                     self.weight[1])
            conv1_2 = self.conv_layer(conv1_1, 64, 'conv1_2', self.weight[2], ↴
                                     self.weight[3])
            pool1 = self.max_pool(conv1_2, 'pool1')

            conv2_1 = self.conv_layer(pool1, 128, 'conv2_1', self.weight[4], ↴
                                     self.weight[5])
            conv2_2 = self.conv_layer(conv2_1, 128, 'conv2_2', self.weight[6], ↴
                                     self.weight[7])
            pool2 = self.max_pool(conv2_2, 'pool2')

            conv3_1 = self.conv_layer(pool2, 256, 'conv3_1', self.weight[8], ↴
                                     self.weight[9])

```

```

        conv3_2 = self.conv_layer(conv3_1, 256, 'conv3_2', self.weight[10],  

→self.weight[11])
        conv3_3 = self.conv_layer(conv3_2, 256, 'conv3_3', self.weight[12],  

→self.weight[13])
        pool3 = self.max_pool(conv3_3, 'pool3')

        conv4_1 = self.conv_layer(pool3, 512, 'conv4_1', self.weight[14],  

→self.weight[15])
        conv4_2 = self.conv_layer(conv4_1, 512, 'conv4_2', self.weight[16],  

→self.weight[17])
        conv4_3 = self.conv_layer(conv4_2, 512, 'conv4_3', self.weight[18],  

→self.weight[19])
        pool4 = self.max_pool(conv4_3, 'pool4')

        conv5_1 = self.conv_layer(pool4, 512, 'conv5_1', self.weight[20],  

→self.weight[21])
        conv5_2 = self.conv_layer(conv5_1, 512, 'conv5_2', self.weight[22],  

→self.weight[23])
        conv5_3 = self.conv_layer(conv5_2, 512, 'conv5_3', self.weight[24],  

→self.weight[25])
        pool5 = self.max_pool(conv5_3, 'pool5')

        flatten = tf.layers.flatten(pool5, name='flatten')
        with tf.variable_scope('fine_tune'):
            y = tf.layers.dense(flatten, 512, activation=tf.nn.relu,  

→name='dense1')
            y = tf.layers.dropout(y, 0.5, training=is_training, name='dp1')
            y = tf.layers.dense(y, 512, activation=tf.nn.relu, name='dens2')
            y = tf.layers.dropout(y, 0.5, training=is_training, name='dp2')
            y = tf.layers.dense(y, output_unit, name='y')

# result = tf.identity(y, name='y_output')
softmax_res = tf.nn.softmax(y, name='softmax')

return y

def conv_layer(self, input, filters, name, kernel_weight, bias_weight,  

→ksize=3):
    with tf.variable_scope(name):
        return tf.layers.conv2d(inputs=input, filters=filters,  

→kernel_size=[ksize, ksize], padding='same',
                               activation=tf.nn.relu,  

→kernel_initializer=self.get_initial(kernel_weight),
                               bias_initializer=self.  

→get_initial(bias_weight),
                               trainable=False)

```

```

def fc_layer(self, input, name, unit, kernel_weight, bias_weight, ↴
↪use_dropout=False, trainable=False, use_relu=True):
    with tf.variable_scope(name):
        if use_relu:
            ac = tf.nn.relu
        else:
            ac = None

        fc = tf.layers.dense(inputs=input, units=unit, activation=ac,
                             kernel_initializer=self.
↪get_initial(kernel_weight),
                             bias_initializer=self.
↪get_initial(bias_weight), trainable=trainable)
        dropout = tf.layers.dropout(fc, 0.5, training=use_dropout)
    return dropout

def max_pool(input, name=None, stride=2):
    return tf.layers.max_pooling2d(inputs=input, pool_size=[2, 2], ↴
↪strides=stride, padding='same', name=name)

def get_initial(data):
    return tf.constant_initializer(data)

```

[14]: # pretrained vgg-face
model.load_weights(state)

features.0.weight	Loaded
features.0.bias	Loaded
features.2.weight	Loaded
features.2.bias	Loaded
features.5.weight	Loaded
features.5.bias	Loaded
features.7.weight	Loaded
features.7.bias	Loaded
features.10.weight	Loaded
features.10.bias	Loaded
features.12.weight	Loaded
features.12.bias	Loaded
features.14.weight	Loaded
features.14.bias	Loaded
features.17.weight	Loaded
features.17.bias	Loaded
features.19.weight	Loaded
features.19.bias	Loaded
features.21.weight	Loaded

```
features.21.bias           Loaded
features.24.weight         Loaded
features.24.bias           Loaded
features.26.weight         Loaded
features.26.bias           Loaded
features.28.weight         Loaded
features.28.bias           Loaded
classifier.0.weight        Ignored
classifier.0.bias           Ignored
classifier.3.weight        Ignored
classifier.3.bias           Ignored
classifier.6.weight        Ignored
classifier.6.bias           Ignored
```

```
[15]: # gives parameter count and memory
model.memory_usage()
```

```
Conv    : 14714688
FC     : 13112328
-----
Total   : 27827016
Memory  : 106.15MB
```

```
[16]: # train for one epoch
worker.train(model, loaders, lr=0.1, epochs=3)
```

```
=> training vgg16
=> found cuda compatible gpu
=> checkpoints will be saved as checkpoint.pth
=> training started at Mar-10 17:26:50

EPOCH : 0
(466/466) [=====]          Loss: 0.5370 (1.2275)      Acc@1: 77.778
(52.007)      Acc@5: 100.000 (96.167)
VALIDATION :
(117/117) [=====]          Loss: 0.9250 (0.8929)      Acc@1: 50.000
(62.997)      Acc@5: 100.000 (99.436)

EPOCH : 1
(466/466) [=====]          Loss: 0.5494 (0.9337)      Acc@1: 77.778
(64.015)      Acc@5: 100.000 (99.107)
VALIDATION :
(117/117) [=====]          Loss: 0.5059 (0.8315)      Acc@1: 66.667
(65.602)      Acc@5: 100.000 (99.329)

EPOCH : 2
(466/466) [=====]          Loss: 0.8179 (0.7982)      Acc@1: 66.667
```

```
(68.640)          Acc@5: 100.000 (99.483)
VALIDATION :
(117/117) [=====]          Loss: 0.8233 (0.8045)          Acc@1: 83.333
(69.603)          Acc@5: 100.000 (99.329)
```

```
[ ]:
```

0.0.1 Gender Prediction

```
[21]: import os
import matplotlib.pyplot as plt
import torchvision.transforms as transforms

[21]: classes = os.listdir(age_class)
classes.sort()
idx_to_class = {i:classes[i] for i in range(len(classes))}

[3]: # turn model to evaluation and move to pu
model.eval()
batch_size = 32
model.to(torch.device("gpu"))

[4]: # inverse transform to show images
mean = loader.mean
std = loader.std
mn_inv = [-m/s for m, s in zip(mean, std)]
sd_inv = [1/s for s in std]
inv_transform = transforms.Normalize(mean=mn_inv, std=sd_inv)

[5]: with torch.no_grad():
    for i, (input, target) in enumerate(valid_loader):

        output = model(input)
        _, preds = torch.max(output, 1)

        fig=plt.figure(figsize=(15, 15))
        columns = 4
        rows = 5

        for i in range(1, columns*rows + 1):

            pred_class = idx_to_class[int(preds[i])]
            real_class = idx_to_class[int(target[i])]

            ax = fig.add_subplot(rows, columns, i)
            ax.title.set_text("Pred Result:" + pred_class + "Ground Truth:" + real_class)
```

```
    ax.axis("off")

    plt.imshow(inv_transform(input[i]).permute(1, 2, 0))

    break

plt.show()
```

0.1 Confusion Matrix

```
[14]: valid_loader = loaders[1]
conf_mat = worker.confusion_matrix(model, valid_loader)
```

```
[13]: for row in conf_mat:
        for elem in row:
            print("%.2f" % (elem * 100), end="\t")
        print("")
```

63.84	26.76	7.56	1.77	0.07
9.32	68.14	17.29	4.38	0.87
1.42	21.58	41.85	27.71	7.44
0.07	1.62	17.81	48.91	31.59
0.01	0.83	4.37	31.28	63.51

VGG-gender

April 29, 2021

0.0.1 Import package

```
[2]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
from PIL import Image
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.layers import Input, Flatten, Dense
from keras.models import Model
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D
```

```
[3]: import math
import cv2
import matplotlib.pyplot as plt
import os
import seaborn as sns
!pip install umap
import umap
from PIL import Image
from scipy import misc
from os import listdir
from os.path import isfile, join
import numpy as np
from scipy import misc
from random import shuffle
from collections import Counter
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.utils import np_utils
```

```
[2]: imageinfo = pd.read_csv("wikifinal.csv")
```

```
[1]: import tensorflow as tf
import pickle
import time
```

0.0.2 import VGG architecture and weights

```
[3]: from build_model_get_weight import get_vgg16
```

```
[ ]: from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.layers import Input, Flatten, Dense
from keras.models import Model
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D
```

```
[4]: import imageio
os.chdir('crop')
```

```
[5]: vgg16_with_top = VGG16(include_top=True, weights='imagenet',
                           input_tensor=None, input_shape=None,
                           pooling=None,
                           classes=1000)
vgg16_with_top.summary()
```

0.0.3 Use only convolutional part of VGG16 and add dense layer¶

```
[12]: # Get back the convolutional part of a VGG network trained on ImageNet
model_vgg16_conv = VGG16(weights='imagenet', include_top=False)
model_vgg16_conv.summary()

# Stop to train weights of convolutional layers, if you want to fit them, unmark down the following two lines
for layer in model_vgg16_conv.layers:
    layer.trainable = False

# Create your own input format (here 224 X 224 X 3)
inputs = Input(shape=(224,224,3),name = 'image_input')

# Use the generated model
output_vgg16_conv = model_vgg16_conv(inputs)

# Add the fully-connected layers
x = Flatten(name='flatten')(output_vgg16_conv)
#x = Dense(128, activation='relu', name='fc1')(x)
```

```

#x = Dense(128, activation='relu', name='fc2')(x)
x = Dense(1000, activation='softmax', name='predictions')(x)

# Create your own model
my_model = Model(inputs=inputs, outputs=x)

# In the summary, weights and layers from VGG part will be hidden, and they
# will not be fit during training
my_model.summary()

```

0.0.4 Fine Tune VGG model parameters

```

[13]: # Generate a model with all layers (with top)
my_vgg16_model = VGG16(weights='imagenet', include_top=True)

# Stop to train weights of VGG16 layers
for layer in my_vgg16_model.layers:
    layer.trainable = False

# Add a layer where input is the output of the second last layer
#x = Flatten(name='flatten')(my_vgg16_model.layers[-2].output)

# Add a layer where input is the output of the fourth last layer
x = Dropout(0.9, noise_shape=None, seed=None)(my_vgg16_model.layers[-4].output)
#x = Dense(128, activation='relu', name='fc1')(x)
#x = Dense(128, activation='relu', name='fc2')(x)
x = Dense(1000, activation='softmax', name='predictions')(x)

# Then create the corresponding model
my_model = Model(inputs=my_vgg16_model.input, outputs=x)
my_model.summary()

```

```

[15]: from keras.preprocessing.image import ImageDataGenerator, array_to_img,
       img_to_array, load_img
import concurrent.futures
import os

```

```

[16]: # The variable 'number' gives the numbers of images generated from one image
def datagen(filename, destination, number):
    datagen = ImageDataGenerator(
        rotation_range=0.2,
        width_shift_range=0.2,
        height_shift_range=0.2,
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,

```

```

        horizontal_flip=True,
        fill_mode='nearest')

img = load_img(filename)
x = img_to_array(img)
x = x.reshape((1,) + x.shape)
filename = filename.split('/')[-1]

i = 0
for batch in datagen.flow(x, batch_size=1,
                           save_to_dir=destination, save_prefix=filename, □
→save_format='jpg'):
    i += 1
    if i > number:
        break

```

[17]:

```

photoDir = 'crop'
photoList = os.listdir(photoDir)
direction = 'crop_new'

# Use multi-threading to speed up
with concurrent.futures.ThreadPoolExecutor(max_workers=20) as executor:
    for photo in photoList:
        try:
            filename = photoDir+photo
            executor.submit(datagen, filename, direction, 20)
        except Exception as exc:
            print(exc)

```

[18]:

```

model = load_model('gender_classify_middle_hiар_man.h5')
label_dict = {0:'female', 1:'male'}

```

[19]:

```

# image prediction
def cropAndPred(image, x, y, w, h):
    center_x = x+w/2
    center_y = y+h/2
    b_dim = min(max(w,h)*1.2,image.width, image.height)
    box = (center_x-b_dim/2, center_y-b_dim/2, center_x+b_dim/2, center_y+b_dim/
→2)
    crpim = image.crop(box).resize((96,96))

    im2Arr = np.asarray(crpim)
    x_test_nor = im2Arr.astype('float32') / 255.0
    x_test_nor = x_test_nor[np.newaxis,:]

    Prediction = model.predict_classes(x_test_nor)

```

```
    return Prediction[0]
```

```
[20]: # show image
def rectAndShow(filename, scaleFactor=1.05, minNeighbor=3, minSize=(100,100)):
    faceCascade = cv2.CascadeClassifier(pathf)
    image = cv2.imread(filename)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray, scaleFactor, minNeighbor,
                                         minSize=minSize)

    im = Image.open(filename)
    for (x, y, w, h) in faces:
        cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 4)
        label = cropAndPred(im, x, y, w, h)
        cv2.putText(image, label_dict[int(label)], (x,y), cv2.
                                         FONT_HERSHEY_SIMPLEX, w/80, (0, 255, 0), 4)
    plt.figure(figsize=(10,10))
    plt.imshow(image)
```

0.0.5 Define hyperparameters using vgg16

```
[5]: # learning rate
lr = 0.001
# number of training epochs
epochs = 50
# number of batch_size
batch_size = 32
total_batch = int(27123/batch_size)
num_steps = (epochs+1) * total_batch
num_classes = 2
W = 224
H = 224
channel = 3

# Placeholder¶
tf.reset_default_graph()
X = tf.placeholder(tf.float32,[None,W, H, channel],name='X')
Y = tf.placeholder(tf.int32,[None,num_classes],name='Y')
training = tf.placeholder_with_default(False, shape=())
```

```
[4]: # Loss and Optimization¶
logits = get_vgg16(X)

loss = tf.nn.softmax_cross_entropy_with_logits_v2(labels=Y, logits=logits)
correct_pred = tf.equal(tf.argmax(logits,1),tf.argmax(Y,1))
accuracy = tf.reduce_mean(tf.cast(correct_pred,tf.float32),name='accuracy')
```

```

optimizer = tf.train.AdamOptimizer(learning_rate=lr)
train_op = optimizer.minimize(loss)
init = tf.global_variables_initializer()

```

```

[8]: # A function to get batch for training and validation
def get_batch(i, size, data):
    inp = [inp[0] for inp in data[(i * size):((i + 1) * size)]]
    label = [lab[1] for lab in data[(i * size):((i + 1) * size)]]
    X_inp = np.empty((len(inp), 224, 224, 3), dtype=np.float32)
    y_label = np.empty((len(inp), 2), dtype=np.float32)
    for j in range(len(inp)):
        X_inp[j] = inp[j]
        y_label[j] = label[j]
    return (X_inp, y_label)

```

0.0.6 Training

```

[9]: saver = tf.train.Saver()
with tf.Session() as sess:
    sess.run(init)
    for i in range(epochs):
        start = time.time()
        for j in range(total_batch):
            batch_x, batch_y = get_batch(j, batch_size, train)
            sess.run(train_op, feed_dict={X:batch_x, Y:batch_y, training:True})

            if j == total_batch - 1:
                tr_acc = sess.run(accuracy, feed_dict={X:batch_x, Y:batch_y})
                print("Epoch "+str(i+1)+", Training accuracy= {:.3f}".
→format(tr_acc))
                acc = 0
                for j in range(20):
                    val_x, val_y = get_batch(j, 25, val)
                    val_acc = sess.run(accuracy, feed_dict={X:val_x, Y:val_y})
                    acc += val_acc
                print("Epoch "+str(i+1)+", Validation accuracy= {:.3f}".format(acc/20))
                print("Training epoch "+str(i+1)+" takes", int(time.time() - start), u
→"seconds")
                save_path = saver.save(sess, 'vgg_face.ckpt')
                print("Training finished!")

```

Epoch 1, Training accuracy= 0.781
 Epoch 1, Validation accuracy= 0.910
 Training epoch 1 takes 203 seconds
 Epoch 2, Training accuracy= 0.812
 Epoch 2, Validation accuracy= 0.874
 Training epoch 2 takes 196 seconds

```
Epoch 3, Training accuracy= 0.875
Epoch 3, Validation accuracy= 0.896
Training epoch 3 takes 197 seconds
Epoch 4, Training accuracy= 0.844
Epoch 4, Validation accuracy= 0.912
Training epoch 4 takes 196 seconds
Epoch 5, Training accuracy= 0.812
Epoch 5, Validation accuracy= 0.924
Training epoch 5 takes 196 seconds
Epoch 6, Training accuracy= 0.906
Epoch 6, Validation accuracy= 0.878
Training epoch 6 takes 196 seconds
Epoch 7, Training accuracy= 0.812
Epoch 7, Validation accuracy= 0.864
Training epoch 7 takes 196 seconds
Epoch 8, Training accuracy= 0.844
Epoch 8, Validation accuracy= 0.902
Training epoch 8 takes 196 seconds
Epoch 9, Training accuracy= 0.844
Epoch 9, Validation accuracy= 0.914
Training epoch 9 takes 196 seconds
Epoch 10, Training accuracy= 0.875
Epoch 10, Validation accuracy= 0.862
Training epoch 10 takes 196 seconds
Training finished!
```

```
[10]: # Inference
import matplotlib.pyplot as plt
import cv2 as cv
# GT as Pred
TP = 0
# F - F
FP = 0
# M - F
TN = 0
# M - M
FN = 0
# F - M

test = []
with open('test_gender', 'rb') as f:
    test += pickle.load(f)
test_np = np.array(test)
img = test_np[:,0]
lab = test_np[:,1]
```

```
[12]: with tf.Session() as sess:
    saver.restore(sess, "vgg_face.ckpt")
    for i in range(len(test)):
        label = ''
        if lab[i][0] == 1:
            label = 'Female'
        else:
            label = 'Male'
        test_img, test_label = get_batch(i, 1, test)
        pred = sess.run(logits, feed_dict = {X:test_img})
        pred = np.squeeze(pred)
        if (pred[1] < pred[0]):
            result = "Female: "+str(pred[0])
            if label == 'Female':
                TP += 1
            else:
                FP += 1
        else:
            result = "Male: "+str(pred[1])
            if label == 'Male':
                TN += 1
            else:
                FN += 1
        if i < 10:
            b,g,r = cv.split(img[i])
            img[i] = cv.merge([r,g,b])
            title = 'Predicted Result: '+result+ " Ground Truth: "+label
            fig = plt.figure()
            plt.imshow(np.squeeze(img[i]))
            plt.title(title)
```

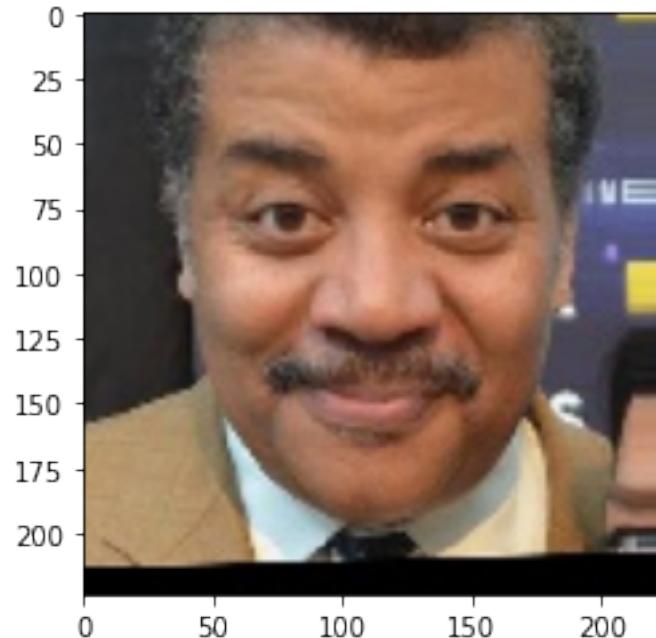
WARNING:tensorflow:From C:\ProgramData\Anaconda2\envs\EE596HW\lib\site-packages\tensorflow\python\training\saver.py:1266: checkpoint_exists (from tensorflow.python.training.checkpoint_management) is deprecated and will be removed in a future version.

Instructions for updating:

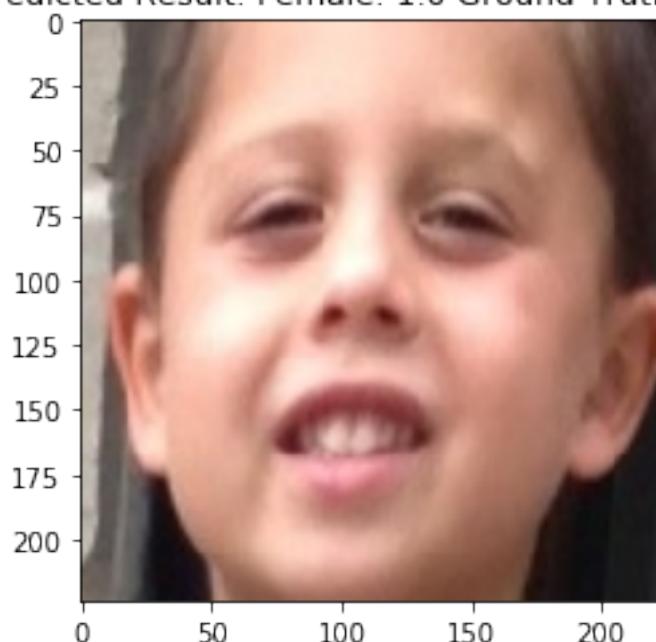
Use standard file APIs to check for files with this prefix.

INFO:tensorflow:Restoring parameters from vgg_face.ckpt

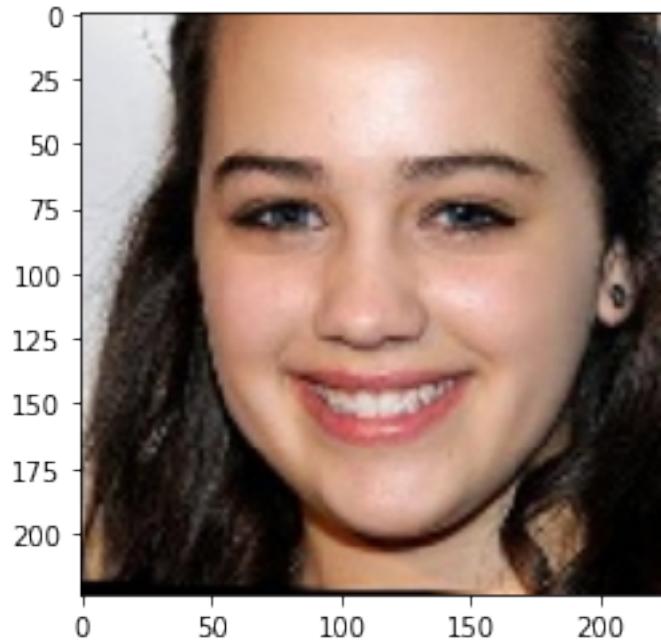
Predicted Result: Male: 1.0 Ground Truth: Male



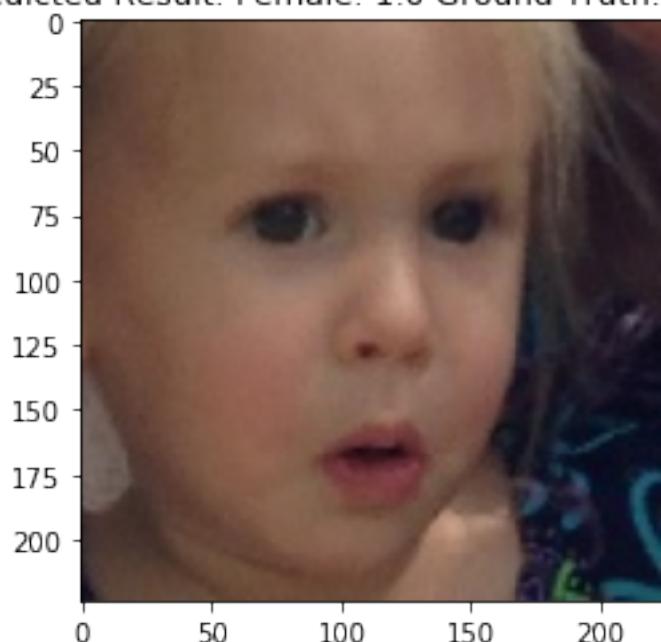
Predicted Result: Female: 1.0 Ground Truth: Male



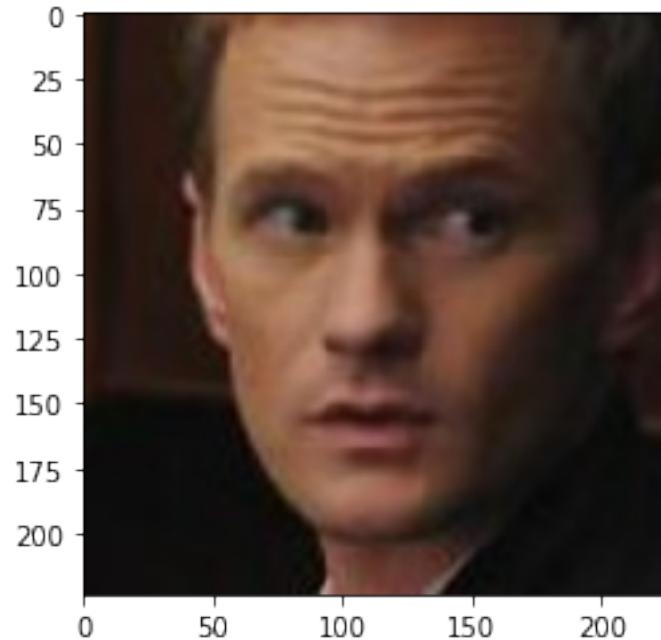
Predicted Result: Female: 1.0 Ground Truth: Female



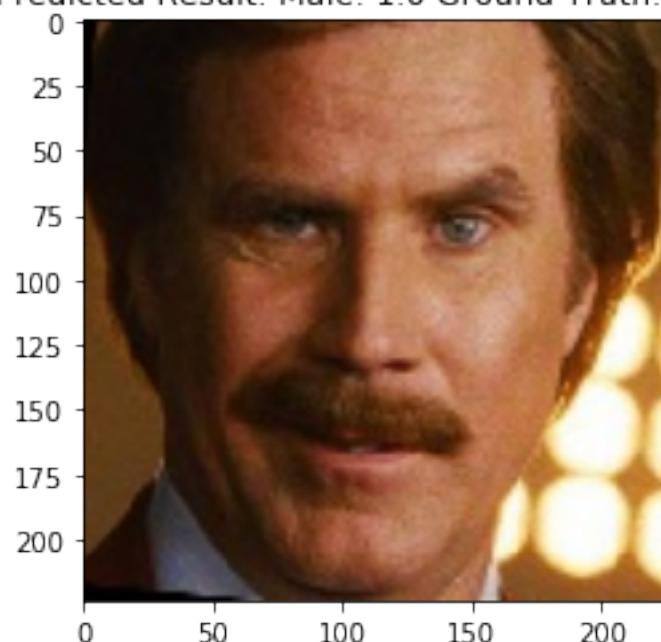
Predicted Result: Female: 1.0 Ground Truth: Female



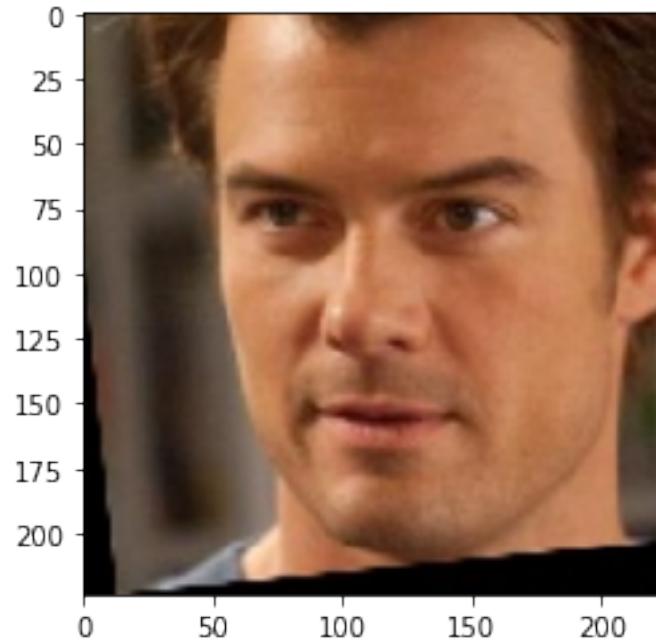
Predicted Result: Male: 1.0 Ground Truth: Male



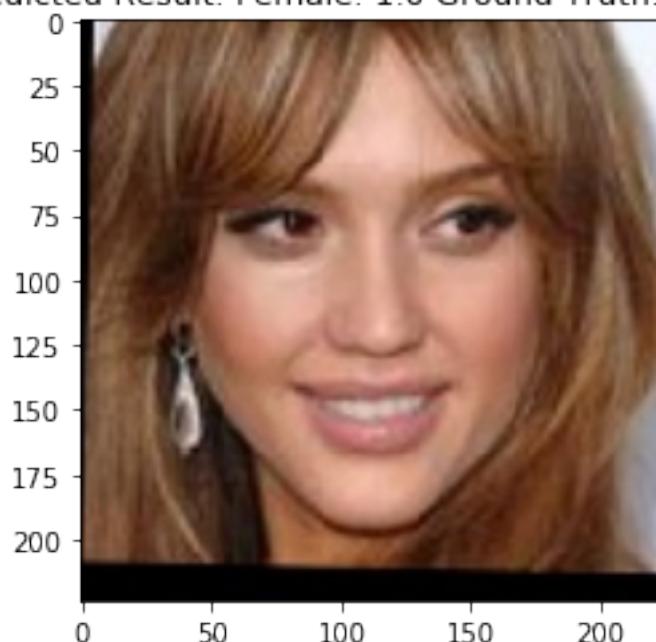
Predicted Result: Male: 1.0 Ground Truth: Male



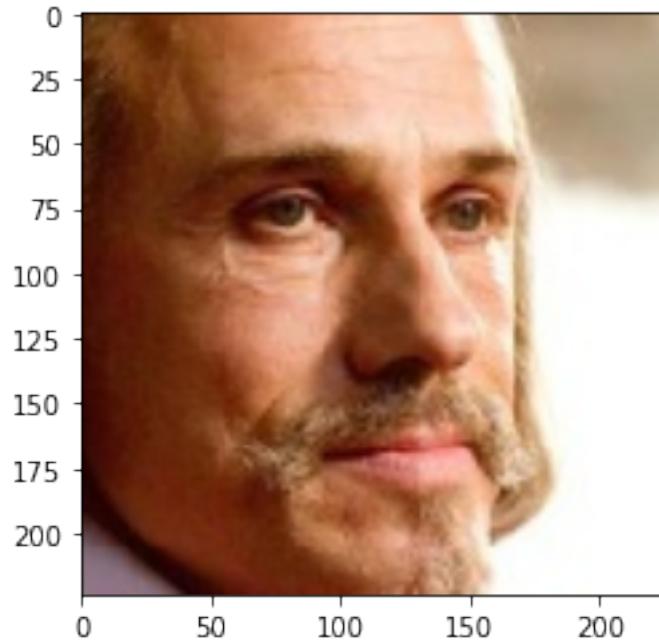
Predicted Result: Male: 1.0 Ground Truth: Male



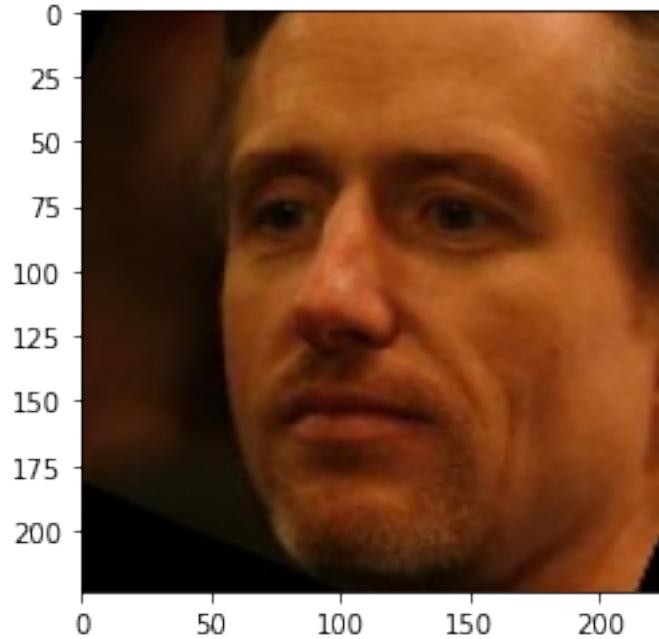
Predicted Result: Female: 1.0 Ground Truth: Female



Predicted Result: Male: 1.0 Ground Truth: Male



Predicted Result: Male: 1.0 Ground Truth: Male



```
[ ]: with tf.Session() as sess:  
    saver.restore(sess, "vgg_face.ckpt")
```

```

for i in range(len(test)):
    label = ''
    if lab[i][0] == 1:
        label = 'Female'
    else:
        label = 'Male'
    test_img, test_label = get_batch(i, 1, test)
    pred = sess.run(logits, feed_dict = {X:test_img})
    pred = np.squeeze(pred)
    if (pred[1] < pred[0]):
        result = "Female: "+str(pred[0])
        if label == 'Female':
            TP += 1
        else:
            FP += 1
    else:
        result = "Male: "+str(pred[1])
        if label == 'Male':
            TN += 1
        else:
            FN += 1
    if i < 10:
        b,g,r = cv.split(img[i])
        img[i] = cv.merge([r,g,b])
        title = 'Predicted Result: '+result+ " Ground Truth: "+label
        fig = plt.figure()
        plt.imshow(np.squeeze(img[i]))
        plt.title(title)

```

[15]: # test accuracy

```

p = TP/(TP+FP)
r = TP/(TP+FN)
f1 = (2*p*r)/(p+r)
acc = (TP+TN)/(TP+FP+TN+FN)
print('Precesion:{:.3f}'.format(p), 'Recall:{:.3f}'.format(r), 'F1:{:.3f}'.
      format(f1), "Accuracy:{:.3f}".format(acc))

```

Precesion:0.759 Recall:0.989 F1:0.859 Accuracy:0.845

Inception V3

- 1 Import libraries
- 2 Download model weights, import model, load weights into model
- 3 Defining Data Generators
 - 3.1 Age
 - 3.2 Gender
- 4 Defing Function
 - 4.1 Early Stopping
 - 4.2 Learning Rate Scheduler
 - 4.3 Plot Result
- 5 Model of Age
 - 5.1 Set layers to be non-trainable for pre-trained model
 - 5.2 Model summary of Inception v3
 - 5.3 Obtain last layer output of the pre-trained model
 - 5.4 Adding dense layers after pre-trained model
 - 5.5 Model summary of Inception v3 with dense layers
 - 5.6 Fitting model
 - 5.7 Model result
- 6 Model of Gender
 - 6.1 Set layers to be non-trainable for pre-trained model
 - 6.2 Model summary of Inception v3
 - 6.3 Obtain last layer output of the pre-trained model
 - 6.4 Adding dense layers after pre-trained model
 - 6.5 Model summary of Inception v3 with dense layers
 - 6.6 Fitting model
 - 6.7 Model result

Import libraries

In [34]:

```
import os
import tensorflow as tf
from tensorflow.keras.optimizers import RMSprop, Adam
from tensorflow.keras import layers
from tensorflow.keras import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import plot_model
from tensorflow.keras import backend as K
from keras.callbacks import LearningRateScheduler, EarlyStopping, Callback, ReduceLR
import matplotlib.pyplot as plt
import math
import imageio
```

Download model weights, import model, load weights into model

```
In [35]: # Download Inception v3 weights to local machine
!wget --no-check-certificate \
    https://storage.googleapis.com/mledu-datasets/inception_v3_weights_tf_dim_orderi
-O /tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5

# Import Inception v3 Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
pre_trained_model = InceptionV3(input_shape=(150, 150, 3), include_top=False, weight

# Load Inception v3 weights into model
# Local_weights_file = '/tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h
# pre_trained_model.load_weights(local_weights_file)
```

'wget' is not recognized as an internal or external command,
operable program or batch file.

Defining Data Generators

Age

```
In [36]: train_dir = "./data/age/train"
validation_dir = "./data/age/val"
test_dir = "./data/age/test"

train_datagen = ImageDataGenerator(rescale=1./255,
                                    rotation_range=40,
                                    width_shift_range=0.2,
                                    height_shift_range=0.2,
                                    zoom_range=0.2,
                                    shear_range=0.2,
                                    horizontal_flip=True,
                                    fill_mode='nearest') # with data augmentation fo

valid_datagen = ImageDataGenerator(rescale=1./255) # no augmentation for validation

test_datagen = ImageDataGenerator(rescale=1./255) # no augmentation for test set

train_age_generator = train_datagen.flow_from_directory(train_dir,
                                                       batch_size=32,
                                                       class_mode='categorical',
                                                       target_size=(150, 150))

val_age_generator = valid_datagen.flow_from_directory(validation_dir,
                                                       batch_size=32,
                                                       class_mode='categorical',
                                                       target_size=(150, 150))

test_age_generator = valid_datagen.flow_from_directory(test_dir,
                                                       batch_size=32,
                                                       class_mode='categorical',
                                                       target_size=(150, 150))
```

Found 21698 images belonging to 5 classes.
 Found 2712 images belonging to 5 classes.
 Found 2713 images belonging to 5 classes.

Gender

```
In [37]: train_dir = "./data/gender/train"
validation_dir = "./data/gender/val"
test_dir = "./data/gender/test"
```

```

train_datagen = ImageDataGenerator(rescale=1./255,
                                    rotation_range=40,
                                    width_shift_range=0.2,
                                    height_shift_range=0.2,
                                    zoom_range=0.2,
                                    shear_range=0.2,
                                    horizontal_flip=True,
                                    fill_mode='nearest') # with data augmentation for training

valid_datagen = ImageDataGenerator(rescale=1./255) # no augmentation for validation

test_datagen = ImageDataGenerator(rescale=1./255) # no augmentation for test set

train_gender_generator = train_datagen.flow_from_directory(train_dir,
                                                          batch_size=32,
                                                          class_mode='categorical',
                                                          target_size=(150, 150))

val_gender_generator = valid_datagen.flow_from_directory(validation_dir,
                                                          batch_size=32,
                                                          class_mode='categorical',
                                                          target_size=(150, 150))

test_gender_generator = valid_datagen.flow_from_directory(test_dir,
                                                          batch_size=32,
                                                          class_mode='categorical',
                                                          target_size=(150, 150))

```

Found 21698 images belonging to 2 classes.

Found 2712 images belonging to 2 classes.

Found 2713 images belonging to 2 classes.

Defining Function

Early Stopping

```
In [38]: earlystop = EarlyStopping(monitor='val_accuracy',
                                 min_delta=0,
                                 patience=10,
                                 verbose=2,
                                 mode='max',
                                 baseline=None,
                                 restore_best_weights=True)
```

Learning Rate Scheduler

```
In [39]: def scheduler(epoch):
    # 每隔10个epoch, 学习率减小为原来的1/10
    if epoch % 10 == 0 and epoch != 0:
        lr = K.get_value(model.optimizer.lr)
        K.set_value(model.optimizer.lr, lr * 0.1)
        print("lr changed to {}".format(lr * 0.1))
    return K.get_value(model.optimizer.lr)
reduce_lr = LearningRateScheduler(scheduler)
```

```
In [40]: def step_decay(epoch):
    initial_lrate = 0.1
    drop = 0.5
    epochs_drop = 10.0
```

```

lrate = initial_lrate * math.pow(drop,
    math.floor((1+epoch)/epochs_drop))
return lrate
lrate = LearningRateScheduler(step_decay)

```

In [41]:

```

# Plateau
# 当评价指标不再提升时，减少学习率
plateau = ReduceLROnPlateau(monitor='val_accuracy',
                            factor=0.1,
                            patience=5,
                            verbose=2,
                            mode='max',
                            min_delta=0.0001,
                            cooldown=0,
                            min_lr=0)

```

In [42]:

```

class LossHistory(Callback):
    def on_train_begin(self, logs={}):
        self.losses = []
        self.lr = []

    def on_epoch_end(self, batch, logs={}):
        self.losses.append(logs.get('loss'))
        self.lr.append(step_decay(len(self.losses)))
loss_history = LossHistory()

```

Plot Result

In [43]:

```

def plot_acc(history):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    epochs = range(len(acc))

    plt.plot(epochs, acc, 'r', label='Training accuracy')
    plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
    plt.title('Training and validation accuracy')
    plt.legend(loc=0)
    plt.figure()
    plt.show()

```

In [44]:

```

def plot_loss(history):
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    epochs = range(len(acc))

    plt.plot(epochs, loss, 'r', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.legend(loc=0)
    plt.figure()
    plt.show()

```

Model of Age

Set layers to be non-trainable for pre-trained model

```
In [17]: for layer in pre_trained_model.layers:
    layer.trainable = False
```

Model summary of Inception v3

```
In [18]: pre_trained_model.summary()
plot_model(pre_trained_model, to_file='inception_v3_model.png', show_shapes=False, s
Model: "inception_v3"
```

Layer (type)	Output Shape	Param #	Connected to
<hr/>			
input_1 (InputLayer)	[None, 150, 150, 3]	0	
conv2d (Conv2D)	(None, 74, 74, 32)	864	input_1[0][0]
batch_normalization (BatchNorma	(None, 74, 74, 32)	96	conv2d[0][0]
activation (Activation) [0][0]	(None, 74, 74, 32)	0	batch_normalization
conv2d_1 (Conv2D)	(None, 72, 72, 32)	9216	activation[0][0]
batch_normalization_1 (BatchNor	(None, 72, 72, 32)	96	conv2d_1[0][0]
activation_1 (Activation) _1[0][0]	(None, 72, 72, 32)	0	batch_normalization
conv2d_2 (Conv2D)	(None, 72, 72, 64)	18432	activation_1[0][0]
batch_normalization_2 (BatchNor	(None, 72, 72, 64)	192	conv2d_2[0][0]
activation_2 (Activation) _2[0][0]	(None, 72, 72, 64)	0	batch_normalization
max_pooling2d (MaxPooling2D)	(None, 35, 35, 64)	0	activation_2[0][0]
conv2d_3 (Conv2D)	(None, 35, 35, 80)	5120	max_pooling2d[0][0]
batch_normalization_3 (BatchNor	(None, 35, 35, 80)	240	conv2d_3[0][0]
activation_3 (Activation) _3[0][0]	(None, 35, 35, 80)	0	batch_normalization
conv2d_4 (Conv2D)	(None, 33, 33, 192)	138240	activation_3[0][0]
batch_normalization_4 (BatchNor	(None, 33, 33, 192)	576	conv2d_4[0][0]

activation_4 (Activation)	(None, 33, 33, 192)	0	batch_normalization_4[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 192)	0	activation_4[0][0]
conv2d_8 (Conv2D)	(None, 16, 16, 64)	12288	max_pooling2d_1[0]
batch_normalization_8 (BatchNor	(None, 16, 16, 64)	192	conv2d_8[0][0]
activation_8 (Activation)	(None, 16, 16, 64)	0	batch_normalization_8[0][0]
conv2d_6 (Conv2D)	(None, 16, 16, 48)	9216	max_pooling2d_1[0]
conv2d_9 (Conv2D)	(None, 16, 16, 96)	55296	activation_8[0][0]
batch_normalization_6 (BatchNor	(None, 16, 16, 48)	144	conv2d_6[0][0]
batch_normalization_9 (BatchNor	(None, 16, 16, 96)	288	conv2d_9[0][0]
activation_6 (Activation)	(None, 16, 16, 48)	0	batch_normalization_6[0][0]
activation_9 (Activation)	(None, 16, 16, 96)	0	batch_normalization_9[0][0]
average_pooling2d (AveragePooli	(None, 16, 16, 192)	0	max_pooling2d_1[0]
conv2d_5 (Conv2D)	(None, 16, 16, 64)	12288	max_pooling2d_1[0]
conv2d_7 (Conv2D)	(None, 16, 16, 64)	76800	activation_6[0][0]
conv2d_10 (Conv2D)	(None, 16, 16, 96)	82944	activation_9[0][0]
conv2d_11 (Conv2D)	(None, 16, 16, 32)	6144	average_pooling2d[0][0]
batch_normalization_5 (BatchNor	(None, 16, 16, 64)	192	conv2d_5[0][0]
batch_normalization_7 (BatchNor	(None, 16, 16, 64)	192	conv2d_7[0][0]
batch_normalization_10 (BatchNo	(None, 16, 16, 96)	288	conv2d_10[0][0]
batch_normalization_11 (BatchNo	(None, 16, 16, 32)	96	conv2d_11[0][0]

activation_5 (Activation) _5[0][0]	(None, 16, 16, 64)	0	batch_normalization
activation_7 (Activation) _7[0][0]	(None, 16, 16, 64)	0	batch_normalization
activation_10 (Activation) _10[0][0]	(None, 16, 16, 96)	0	batch_normalization
activation_11 (Activation) _11[0][0]	(None, 16, 16, 32)	0	batch_normalization
mixed0 (Concatenate)	(None, 16, 16, 256)	0	activation_5[0][0] activation_7[0][0] activation_10[0][0] activation_11[0][0]
conv2d_15 (Conv2D)	(None, 16, 16, 64)	16384	mixed0[0][0]
batch_normalization_15 (BatchNo Normalizer) (None, 16, 16, 64)	(None, 16, 16, 64)	192	conv2d_15[0][0]
activation_15 (Activation) _15[0][0]	(None, 16, 16, 64)	0	batch_normalization
conv2d_13 (Conv2D)	(None, 16, 16, 48)	12288	mixed0[0][0]
conv2d_16 (Conv2D)	(None, 16, 16, 96)	55296	activation_15[0][0]
batch_normalization_13 (BatchNo Normalizer) (None, 16, 16, 48)	(None, 16, 16, 48)	144	conv2d_13[0][0]
batch_normalization_16 (BatchNo Normalizer) (None, 16, 16, 96)	(None, 16, 16, 96)	288	conv2d_16[0][0]
activation_13 (Activation) _13[0][0]	(None, 16, 16, 48)	0	batch_normalization
activation_16 (Activation) _16[0][0]	(None, 16, 16, 96)	0	batch_normalization
average_pooling2d_1 (AveragePoo ling) (None, 16, 16, 256)	(None, 16, 16, 256)	0	mixed0[0][0]
conv2d_12 (Conv2D)	(None, 16, 16, 64)	16384	mixed0[0][0]
conv2d_14 (Conv2D)	(None, 16, 16, 64)	76800	activation_13[0][0]
conv2d_17 (Conv2D)	(None, 16, 16, 96)	82944	activation_16[0][0]
conv2d_18 (Conv2D) [0][0]	(None, 16, 16, 64)	16384	average_pooling2d_1

batch_normalization_12 (BatchNo (None, 16, 16, 64)	192	conv2d_12[0][0]
batch_normalization_14 (BatchNo (None, 16, 16, 64)	192	conv2d_14[0][0]
batch_normalization_17 (BatchNo (None, 16, 16, 96)	288	conv2d_17[0][0]
batch_normalization_18 (BatchNo (None, 16, 16, 64)	192	conv2d_18[0][0]
activation_12 (Activation) (None, 16, 16, 64)	0	batch_normalization_12[0][0]
activation_14 (Activation) (None, 16, 16, 64)	0	batch_normalization_14[0][0]
activation_17 (Activation) (None, 16, 16, 96)	0	batch_normalization_17[0][0]
activation_18 (Activation) (None, 16, 16, 64)	0	batch_normalization_18[0][0]
mixed1 (Concatenate) (None, 16, 16, 288)	0	activation_12[0][0] activation_14[0][0] activation_17[0][0] activation_18[0][0]
conv2d_22 (Conv2D) (None, 16, 16, 64)	18432	mixed1[0][0]
batch_normalization_22 (BatchNo (None, 16, 16, 64)	192	conv2d_22[0][0]
activation_22 (Activation) (None, 16, 16, 64)	0	batch_normalization_22[0][0]
conv2d_20 (Conv2D) (None, 16, 16, 48)	13824	mixed1[0][0]
conv2d_23 (Conv2D) (None, 16, 16, 96)	55296	activation_22[0][0]
batch_normalization_20 (BatchNo (None, 16, 16, 48)	144	conv2d_20[0][0]
batch_normalization_23 (BatchNo (None, 16, 16, 96)	288	conv2d_23[0][0]
activation_20 (Activation) (None, 16, 16, 48)	0	batch_normalization_20[0][0]
activation_23 (Activation) (None, 16, 16, 96)	0	batch_normalization_23[0][0]
average_pooling2d_2 (AveragePoo (None, 16, 16, 288)	0	mixed1[0][0]
conv2d_19 (Conv2D) (None, 16, 16, 64)	18432	mixed1[0][0]

conv2d_21 (Conv2D)	(None, 16, 16, 64)	76800	activation_20[0][0]
conv2d_24 (Conv2D)	(None, 16, 16, 96)	82944	activation_23[0][0]
conv2d_25 (Conv2D) [0][0]	(None, 16, 16, 64)	18432	average_pooling2d_2 [0][0]
batch_normalization_19 (BatchNo)	(None, 16, 16, 64)	192	conv2d_19[0][0]
batch_normalization_21 (BatchNo)	(None, 16, 16, 64)	192	conv2d_21[0][0]
batch_normalization_24 (BatchNo)	(None, 16, 16, 96)	288	conv2d_24[0][0]
batch_normalization_25 (BatchNo)	(None, 16, 16, 64)	192	conv2d_25[0][0]
activation_19 (Activation) [0][0]	(None, 16, 16, 64)	0	batch_normalization [0][0]
activation_21 (Activation) [0][0]	(None, 16, 16, 64)	0	batch_normalization [0][0]
activation_24 (Activation) [0][0]	(None, 16, 16, 96)	0	batch_normalization [0][0]
activation_25 (Activation) [0][0]	(None, 16, 16, 64)	0	batch_normalization [0][0]
mixed2 (Concatenate)	(None, 16, 16, 288)	0	activation_19[0][0] activation_21[0][0] activation_24[0][0] activation_25[0][0]
conv2d_27 (Conv2D)	(None, 16, 16, 64)	18432	mixed2[0][0]
batch_normalization_27 (BatchNo)	(None, 16, 16, 64)	192	conv2d_27[0][0]
activation_27 (Activation) [0][0]	(None, 16, 16, 64)	0	batch_normalization [0][0]
conv2d_28 (Conv2D)	(None, 16, 16, 96)	55296	activation_27[0][0]
batch_normalization_28 (BatchNo)	(None, 16, 16, 96)	288	conv2d_28[0][0]
activation_28 (Activation) [0][0]	(None, 16, 16, 96)	0	batch_normalization [0][0]
conv2d_26 (Conv2D)	(None, 7, 7, 384)	995328	mixed2[0][0]
conv2d_29 (Conv2D)	(None, 7, 7, 96)	82944	activation_28[0][0]

batch_normalization_26 (BatchNo	(None, 7, 7, 384)	1152	conv2d_26[0][0]
batch_normalization_29 (BatchNo	(None, 7, 7, 96)	288	conv2d_29[0][0]
activation_26 (Activation)	(None, 7, 7, 384)	0	batch_normalization_26[0][0]
activation_29 (Activation)	(None, 7, 7, 96)	0	batch_normalization_29[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 7, 7, 288)	0	mixed2[0][0]
mixed3 (Concatenate)	(None, 7, 7, 768)	0	activation_26[0][0] activation_29[0][0] max_pooling2d_2[0]
conv2d_34 (Conv2D)	(None, 7, 7, 128)	98304	mixed3[0][0]
batch_normalization_34 (BatchNo	(None, 7, 7, 128)	384	conv2d_34[0][0]
activation_34 (Activation)	(None, 7, 7, 128)	0	batch_normalization_34[0][0]
conv2d_35 (Conv2D)	(None, 7, 7, 128)	114688	activation_34[0][0]
batch_normalization_35 (BatchNo	(None, 7, 7, 128)	384	conv2d_35[0][0]
activation_35 (Activation)	(None, 7, 7, 128)	0	batch_normalization_35[0][0]
conv2d_31 (Conv2D)	(None, 7, 7, 128)	98304	mixed3[0][0]
conv2d_36 (Conv2D)	(None, 7, 7, 128)	114688	activation_35[0][0]
batch_normalization_31 (BatchNo	(None, 7, 7, 128)	384	conv2d_31[0][0]
batch_normalization_36 (BatchNo	(None, 7, 7, 128)	384	conv2d_36[0][0]
activation_31 (Activation)	(None, 7, 7, 128)	0	batch_normalization_31[0][0]
activation_36 (Activation)	(None, 7, 7, 128)	0	batch_normalization_36[0][0]
conv2d_32 (Conv2D)	(None, 7, 7, 128)	114688	activation_31[0][0]
conv2d_37 (Conv2D)	(None, 7, 7, 128)	114688	activation_36[0][0]

batch_normalization_32 (BatchNo (None, 7, 7, 128))	384	conv2d_32[0][0]
batch_normalization_37 (BatchNo (None, 7, 7, 128))	384	conv2d_37[0][0]
activation_32 (Activation) (None, 7, 7, 128)	0	batch_normalization_32[0][0]
activation_37 (Activation) (None, 7, 7, 128)	0	batch_normalization_37[0][0]
average_pooling2d_3 (AveragePoo (None, 7, 7, 768))	0	mixed3[0][0]
conv2d_30 (Conv2D) (None, 7, 7, 192)	147456	mixed3[0][0]
conv2d_33 (Conv2D) (None, 7, 7, 192)	172032	activation_32[0][0]
conv2d_38 (Conv2D) (None, 7, 7, 192)	172032	activation_37[0][0]
conv2d_39 (Conv2D) (None, 7, 7, 192)	147456	average_pooling2d_3[0][0]
batch_normalization_30 (BatchNo (None, 7, 7, 192))	576	conv2d_30[0][0]
batch_normalization_33 (BatchNo (None, 7, 7, 192))	576	conv2d_33[0][0]
batch_normalization_38 (BatchNo (None, 7, 7, 192))	576	conv2d_38[0][0]
batch_normalization_39 (BatchNo (None, 7, 7, 192))	576	conv2d_39[0][0]
activation_30 (Activation) (None, 7, 7, 192)	0	batch_normalization_30[0][0]
activation_33 (Activation) (None, 7, 7, 192)	0	batch_normalization_33[0][0]
activation_38 (Activation) (None, 7, 7, 192)	0	batch_normalization_38[0][0]
activation_39 (Activation) (None, 7, 7, 192)	0	batch_normalization_39[0][0]
mixed4 (Concatenate) (None, 7, 7, 768)	0	activation_30[0][0] activation_33[0][0] activation_38[0][0] activation_39[0][0]
conv2d_44 (Conv2D) (None, 7, 7, 160)	122880	mixed4[0][0]

batch_normalization_44 (BatchNo	(None, 7, 7, 160)	480	conv2d_44[0][0]
activation_44 (Activation)	(None, 7, 7, 160)	0	batch_normalization_44[0][0]
conv2d_45 (Conv2D)	(None, 7, 7, 160)	179200	activation_44[0][0]
batch_normalization_45 (BatchNo	(None, 7, 7, 160)	480	conv2d_45[0][0]
activation_45 (Activation)	(None, 7, 7, 160)	0	batch_normalization_45[0][0]
conv2d_41 (Conv2D)	(None, 7, 7, 160)	122880	mixed4[0][0]
conv2d_46 (Conv2D)	(None, 7, 7, 160)	179200	activation_45[0][0]
batch_normalization_41 (BatchNo	(None, 7, 7, 160)	480	conv2d_41[0][0]
batch_normalization_46 (BatchNo	(None, 7, 7, 160)	480	conv2d_46[0][0]
activation_41 (Activation)	(None, 7, 7, 160)	0	batch_normalization_41[0][0]
activation_46 (Activation)	(None, 7, 7, 160)	0	batch_normalization_46[0][0]
conv2d_42 (Conv2D)	(None, 7, 7, 160)	179200	activation_41[0][0]
conv2d_47 (Conv2D)	(None, 7, 7, 160)	179200	activation_46[0][0]
batch_normalization_42 (BatchNo	(None, 7, 7, 160)	480	conv2d_42[0][0]
batch_normalization_47 (BatchNo	(None, 7, 7, 160)	480	conv2d_47[0][0]
activation_42 (Activation)	(None, 7, 7, 160)	0	batch_normalization_42[0][0]
activation_47 (Activation)	(None, 7, 7, 160)	0	batch_normalization_47[0][0]
average_pooling2d_4 (AveragePoo	(None, 7, 7, 768)	0	mixed4[0][0]
conv2d_40 (Conv2D)	(None, 7, 7, 192)	147456	mixed4[0][0]
conv2d_43 (Conv2D)	(None, 7, 7, 192)	215040	activation_42[0][0]
conv2d_48 (Conv2D)	(None, 7, 7, 192)	215040	activation_47[0][0]

conv2d_49 (Conv2D) [0][0]	(None, 7, 7, 192)	147456	average_pooling2d_4
batch_normalization_40 (BatchNo [0][0])	(None, 7, 7, 192)	576	conv2d_40[0][0]
batch_normalization_43 (BatchNo [0][0])	(None, 7, 7, 192)	576	conv2d_43[0][0]
batch_normalization_48 (BatchNo [0][0])	(None, 7, 7, 192)	576	conv2d_48[0][0]
batch_normalization_49 (BatchNo [0][0])	(None, 7, 7, 192)	576	conv2d_49[0][0]
activation_40 (Activation) _40[0][0]	(None, 7, 7, 192)	0	batch_normalization
activation_43 (Activation) _43[0][0]	(None, 7, 7, 192)	0	batch_normalization
activation_48 (Activation) _48[0][0]	(None, 7, 7, 192)	0	batch_normalization
activation_49 (Activation) _49[0][0]	(None, 7, 7, 192)	0	batch_normalization
mixed5 (Concatenate)	(None, 7, 7, 768)	0	activation_40[0][0] activation_43[0][0] activation_48[0][0] activation_49[0][0]
conv2d_54 (Conv2D)	(None, 7, 7, 160)	122880	mixed5[0][0]
batch_normalization_54 (BatchNo [0][0])	(None, 7, 7, 160)	480	conv2d_54[0][0]
activation_54 (Activation) _54[0][0]	(None, 7, 7, 160)	0	batch_normalization
conv2d_55 (Conv2D)	(None, 7, 7, 160)	179200	activation_54[0][0]
batch_normalization_55 (BatchNo [0][0])	(None, 7, 7, 160)	480	conv2d_55[0][0]
activation_55 (Activation) _55[0][0]	(None, 7, 7, 160)	0	batch_normalization
conv2d_51 (Conv2D)	(None, 7, 7, 160)	122880	mixed5[0][0]
conv2d_56 (Conv2D)	(None, 7, 7, 160)	179200	activation_55[0][0]
batch_normalization_51 (BatchNo [0][0])	(None, 7, 7, 160)	480	conv2d_51[0][0]
batch_normalization_56 (BatchNo [0][0])	(None, 7, 7, 160)	480	conv2d_56[0][0]

activation_51 (Activation)	(None, 7, 7, 160)	0	batch_normalization_51[0][0]
activation_56 (Activation)	(None, 7, 7, 160)	0	batch_normalization_56[0][0]
conv2d_52 (Conv2D)	(None, 7, 7, 160)	179200	activation_51[0][0]
conv2d_57 (Conv2D)	(None, 7, 7, 160)	179200	activation_56[0][0]
batch_normalization_52 (BatchNo)	(None, 7, 7, 160)	480	conv2d_52[0][0]
batch_normalization_57 (BatchNo)	(None, 7, 7, 160)	480	conv2d_57[0][0]
activation_52 (Activation)	(None, 7, 7, 160)	0	batch_normalization_52[0][0]
activation_57 (Activation)	(None, 7, 7, 160)	0	batch_normalization_57[0][0]
average_pooling2d_5 (AveragePoo	(None, 7, 7, 768)	0	mixed5[0][0]
conv2d_50 (Conv2D)	(None, 7, 7, 192)	147456	mixed5[0][0]
conv2d_53 (Conv2D)	(None, 7, 7, 192)	215040	activation_52[0][0]
conv2d_58 (Conv2D)	(None, 7, 7, 192)	215040	activation_57[0][0]
conv2d_59 (Conv2D)	(None, 7, 7, 192)	147456	average_pooling2d_5[0][0]
batch_normalization_50 (BatchNo)	(None, 7, 7, 192)	576	conv2d_50[0][0]
batch_normalization_53 (BatchNo)	(None, 7, 7, 192)	576	conv2d_53[0][0]
batch_normalization_58 (BatchNo)	(None, 7, 7, 192)	576	conv2d_58[0][0]
batch_normalization_59 (BatchNo)	(None, 7, 7, 192)	576	conv2d_59[0][0]
activation_50 (Activation)	(None, 7, 7, 192)	0	batch_normalization_50[0][0]
activation_53 (Activation)	(None, 7, 7, 192)	0	batch_normalization_53[0][0]
activation_58 (Activation)	(None, 7, 7, 192)	0	batch_normalization_58[0][0]

activation_59 (Activation) _59[0][0]	(None, 7, 7, 192)	0	batch_normalization
mixed6 (Concatenate)	(None, 7, 7, 768)	0	activation_50[0][0] activation_53[0][0] activation_58[0][0] activation_59[0][0]
conv2d_64 (Conv2D)	(None, 7, 7, 192)	147456	mixed6[0][0]
batch_normalization_64 (BatchNo)	(None, 7, 7, 192)	576	conv2d_64[0][0]
activation_64 (Activation) _64[0][0]	(None, 7, 7, 192)	0	batch_normalization
conv2d_65 (Conv2D)	(None, 7, 7, 192)	258048	activation_64[0][0]
batch_normalization_65 (BatchNo)	(None, 7, 7, 192)	576	conv2d_65[0][0]
activation_65 (Activation) _65[0][0]	(None, 7, 7, 192)	0	batch_normalization
conv2d_61 (Conv2D)	(None, 7, 7, 192)	147456	mixed6[0][0]
conv2d_66 (Conv2D)	(None, 7, 7, 192)	258048	activation_65[0][0]
batch_normalization_61 (BatchNo)	(None, 7, 7, 192)	576	conv2d_61[0][0]
batch_normalization_66 (BatchNo)	(None, 7, 7, 192)	576	conv2d_66[0][0]
activation_61 (Activation) _61[0][0]	(None, 7, 7, 192)	0	batch_normalization
activation_66 (Activation) _66[0][0]	(None, 7, 7, 192)	0	batch_normalization
conv2d_62 (Conv2D)	(None, 7, 7, 192)	258048	activation_61[0][0]
conv2d_67 (Conv2D)	(None, 7, 7, 192)	258048	activation_66[0][0]
batch_normalization_62 (BatchNo)	(None, 7, 7, 192)	576	conv2d_62[0][0]
batch_normalization_67 (BatchNo)	(None, 7, 7, 192)	576	conv2d_67[0][0]
activation_62 (Activation) _62[0][0]	(None, 7, 7, 192)	0	batch_normalization
activation_67 (Activation) _67[0][0]	(None, 7, 7, 192)	0	batch_normalization

average_pooling2d_6 (AveragePooling2D)	(None, 7, 7, 768)	0	mixed6[0][0]
conv2d_60 (Conv2D)	(None, 7, 7, 192)	147456	mixed6[0][0]
conv2d_63 (Conv2D)	(None, 7, 7, 192)	258048	activation_62[0][0]
conv2d_68 (Conv2D)	(None, 7, 7, 192)	258048	activation_67[0][0]
conv2d_69 (Conv2D)	(None, 7, 7, 192)	147456	average_pooling2d_6[0][0]
batch_normalization_60 (BatchNormalization)	(None, 7, 7, 192)	576	conv2d_60[0][0]
batch_normalization_63 (BatchNormalization)	(None, 7, 7, 192)	576	conv2d_63[0][0]
batch_normalization_68 (BatchNormalization)	(None, 7, 7, 192)	576	conv2d_68[0][0]
batch_normalization_69 (BatchNormalization)	(None, 7, 7, 192)	576	conv2d_69[0][0]
activation_60 (Activation)	(None, 7, 7, 192)	0	batch_normalization_60[0][0]
activation_63 (Activation)	(None, 7, 7, 192)	0	batch_normalization_63[0][0]
activation_68 (Activation)	(None, 7, 7, 192)	0	batch_normalization_68[0][0]
activation_69 (Activation)	(None, 7, 7, 192)	0	batch_normalization_69[0][0]
mixed7 (Concatenate)	(None, 7, 7, 768)	0	activation_60[0][0] activation_63[0][0] activation_68[0][0] activation_69[0][0]
conv2d_72 (Conv2D)	(None, 7, 7, 192)	147456	mixed7[0][0]
batch_normalization_72 (BatchNormalization)	(None, 7, 7, 192)	576	conv2d_72[0][0]
activation_72 (Activation)	(None, 7, 7, 192)	0	batch_normalization_72[0][0]
conv2d_73 (Conv2D)	(None, 7, 7, 192)	258048	activation_72[0][0]
batch_normalization_73 (BatchNormalization)	(None, 7, 7, 192)	576	conv2d_73[0][0]
activation_73 (Activation)	(None, 7, 7, 192)	0	batch_normalization_73[0][0]

conv2d_70 (Conv2D)	(None, 7, 7, 192)	147456	mixed7[0][0]
conv2d_74 (Conv2D)	(None, 7, 7, 192)	258048	activation_73[0][0]
batch_normalization_70 (BatchNo)	(None, 7, 7, 192)	576	conv2d_70[0][0]
batch_normalization_74 (BatchNo)	(None, 7, 7, 192)	576	conv2d_74[0][0]
activation_70 (Activation)	(None, 7, 7, 192)	0	batch_normalization_70[0][0]
activation_74 (Activation)	(None, 7, 7, 192)	0	batch_normalization_74[0][0]
conv2d_71 (Conv2D)	(None, 3, 3, 320)	552960	activation_70[0][0]
conv2d_75 (Conv2D)	(None, 3, 3, 192)	331776	activation_74[0][0]
batch_normalization_71 (BatchNo)	(None, 3, 3, 320)	960	conv2d_71[0][0]
batch_normalization_75 (BatchNo)	(None, 3, 3, 192)	576	conv2d_75[0][0]
activation_71 (Activation)	(None, 3, 3, 320)	0	batch_normalization_71[0][0]
activation_75 (Activation)	(None, 3, 3, 192)	0	batch_normalization_75[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 3, 3, 768)	0	mixed7[0][0]
mixed8 (Concatenate)	(None, 3, 3, 1280)	0	activation_71[0][0] activation_75[0][0] max_pooling2d_3[0]
conv2d_80 (Conv2D)	(None, 3, 3, 448)	573440	mixed8[0][0]
batch_normalization_80 (BatchNo)	(None, 3, 3, 448)	1344	conv2d_80[0][0]
activation_80 (Activation)	(None, 3, 3, 448)	0	batch_normalization_80[0][0]
conv2d_77 (Conv2D)	(None, 3, 3, 384)	491520	mixed8[0][0]
conv2d_81 (Conv2D)	(None, 3, 3, 384)	1548288	activation_80[0][0]
batch_normalization_77 (BatchNo)	(None, 3, 3, 384)	1152	conv2d_77[0][0]

batch_normalization_81 (BatchNo	(None, 3, 3, 384)	1152	conv2d_81[0][0]
activation_77 (Activation)	(None, 3, 3, 384)	0	batch_normalization_77[0][0]
activation_81 (Activation)	(None, 3, 3, 384)	0	batch_normalization_81[0][0]
conv2d_78 (Conv2D)	(None, 3, 3, 384)	442368	activation_77[0][0]
conv2d_79 (Conv2D)	(None, 3, 3, 384)	442368	activation_77[0][0]
conv2d_82 (Conv2D)	(None, 3, 3, 384)	442368	activation_81[0][0]
conv2d_83 (Conv2D)	(None, 3, 3, 384)	442368	activation_81[0][0]
average_pooling2d_7 (AveragePoo	(None, 3, 3, 1280)	0	mixed8[0][0]
conv2d_76 (Conv2D)	(None, 3, 3, 320)	409600	mixed8[0][0]
batch_normalization_78 (BatchNo	(None, 3, 3, 384)	1152	conv2d_78[0][0]
batch_normalization_79 (BatchNo	(None, 3, 3, 384)	1152	conv2d_79[0][0]
batch_normalization_82 (BatchNo	(None, 3, 3, 384)	1152	conv2d_82[0][0]
batch_normalization_83 (BatchNo	(None, 3, 3, 384)	1152	conv2d_83[0][0]
conv2d_84 (Conv2D)	(None, 3, 3, 192)	245760	average_pooling2d_7[0][0]
batch_normalization_76 (BatchNo	(None, 3, 3, 320)	960	conv2d_76[0][0]
activation_78 (Activation)	(None, 3, 3, 384)	0	batch_normalization_78[0][0]
activation_79 (Activation)	(None, 3, 3, 384)	0	batch_normalization_79[0][0]
activation_82 (Activation)	(None, 3, 3, 384)	0	batch_normalization_82[0][0]
activation_83 (Activation)	(None, 3, 3, 384)	0	batch_normalization_83[0][0]
batch_normalization_84 (BatchNo	(None, 3, 3, 192)	576	conv2d_84[0][0]
activation_76 (Activation)	(None, 3, 3, 320)	0	batch_normalization_76[0][0]

_76[0][0]

<u>mixed9_0</u> (Concatenate)	(None, 3, 3, 768)	0	activation_78[0][0] activation_79[0][0]
<u>concatenate</u> (Concatenate)	(None, 3, 3, 768)	0	activation_82[0][0] activation_83[0][0]
<u>activation_84</u> (Activation)	(None, 3, 3, 192)	0	batch_normalization_84[0][0]
<u>mixed9</u> (Concatenate)	(None, 3, 3, 2048)	0	activation_76[0][0] mixed9_0[0][0] concatenate[0][0] activation_84[0][0]
<u>conv2d_89</u> (Conv2D)	(None, 3, 3, 448)	917504	mixed9[0][0]
<u>batch_normalization_89</u> (BatchNo)	(None, 3, 3, 448)	1344	conv2d_89[0][0]
<u>activation_89</u> (Activation)	(None, 3, 3, 448)	0	batch_normalization_89[0][0]
<u>conv2d_86</u> (Conv2D)	(None, 3, 3, 384)	786432	mixed9[0][0]
<u>conv2d_90</u> (Conv2D)	(None, 3, 3, 384)	1548288	activation_89[0][0]
<u>batch_normalization_86</u> (BatchNo)	(None, 3, 3, 384)	1152	conv2d_86[0][0]
<u>batch_normalization_90</u> (BatchNo)	(None, 3, 3, 384)	1152	conv2d_90[0][0]
<u>activation_86</u> (Activation)	(None, 3, 3, 384)	0	batch_normalization_86[0][0]
<u>activation_90</u> (Activation)	(None, 3, 3, 384)	0	batch_normalization_90[0][0]
<u>conv2d_87</u> (Conv2D)	(None, 3, 3, 384)	442368	activation_86[0][0]
<u>conv2d_88</u> (Conv2D)	(None, 3, 3, 384)	442368	activation_86[0][0]
<u>conv2d_91</u> (Conv2D)	(None, 3, 3, 384)	442368	activation_90[0][0]
<u>conv2d_92</u> (Conv2D)	(None, 3, 3, 384)	442368	activation_90[0][0]
<u>average_pooling2d_8</u> (AveragePoo	(None, 3, 3, 2048)	0	mixed9[0][0]
<u>conv2d_85</u> (Conv2D)	(None, 3, 3, 320)	655360	mixed9[0][0]

batch_normalization_87 (BatchNo	(None, 3, 3, 384)	1152	conv2d_87[0][0]
batch_normalization_88 (BatchNo	(None, 3, 3, 384)	1152	conv2d_88[0][0]
batch_normalization_91 (BatchNo	(None, 3, 3, 384)	1152	conv2d_91[0][0]
batch_normalization_92 (BatchNo	(None, 3, 3, 384)	1152	conv2d_92[0][0]
conv2d_93 (Conv2D	(None, 3, 3, 192)	393216	average_pooling2d_8[0][0]
batch_normalization_85 (BatchNo	(None, 3, 3, 320)	960	conv2d_85[0][0]
activation_87 (Activation	(None, 3, 3, 384)	0	batch_normalization_87[0][0]
activation_88 (Activation	(None, 3, 3, 384)	0	batch_normalization_88[0][0]
activation_91 (Activation	(None, 3, 3, 384)	0	batch_normalization_91[0][0]
activation_92 (Activation)	(None, 3, 3, 384)	0	batch_normalization_92[0][0]
batch_normalization_93 (BatchNo	(None, 3, 3, 192)	576	conv2d_93[0][0]
activation_85 (Activation)	(None, 3, 3, 320)	0	batch_normalization_85[0][0]
mixed9_1 (Concatenate)	(None, 3, 3, 768)	0	activation_87[0][0] activation_88[0][0]
concatenate_1 (Concatenate)	(None, 3, 3, 768)	0	activation_91[0][0] activation_92[0][0]
activation_93 (Activation)	(None, 3, 3, 192)	0	batch_normalization_93[0][0]
mixed10 (Concatenate)	(None, 3, 3, 2048)	0	activation_85[0][0] mixed9_1[0][0] concatenate_1[0][0] activation_93[0][0]
<hr/>			
<hr/>			
Total params: 21,802,784			
Trainable params: 0			
Non-trainable params: 21,802,784			
<hr/>			
('Failed to import pydot. You must `pip install pydot` and install graphviz (https://graphviz.gitlab.io/download/), ', 'for `pydotprint` to work.')			

Obtain last layer output of the pre-trained model

```
In [19]: last_layer = pre_trained_model.get_layer('mixed7') # identify Last Layer through model
last_output = last_layer.output
```

Adding dense layers after pre-trained model

```
In [20]: x = layers.Flatten()(last_output)
x = layers.Dense(256, activation='relu', kernel_initializer='he_uniform')(x)
x = layers.Dense(512, activation='relu', kernel_initializer='he_uniform')(x)
x = layers.Dense(1024, activation='relu', kernel_initializer='he_uniform')(x)
x = layers.Dense(1024, activation='relu', kernel_initializer='he_uniform')(x)
x = layers.Dropout(0.2)(x)
x = layers.Dense(5, activation='softmax')(x)

model = Model(pre_trained_model.input, x)

model.compile(optimizer=Adam(lr=0.0001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Model summary of Inception v3 with dense layers

```
In [21]: model.summary()
plot_model(model, to_file='inception_v3_with_dense_layers_model.png', show_shapes=False)
```

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
<hr/>			
input_1 (InputLayer)	[None, 150, 150, 3]	0	
conv2d (Conv2D)	(None, 74, 74, 32)	864	input_1[0][0]
batch_normalization (BatchNorma	(None, 74, 74, 32)	96	conv2d[0][0]
activation (Activation) [0][0]	(None, 74, 74, 32)	0	batch_normalization
conv2d_1 (Conv2D)	(None, 72, 72, 32)	9216	activation[0][0]
batch_normalization_1 (BatchNor	(None, 72, 72, 32)	96	conv2d_1[0][0]
activation_1 (Activation) _1[0][0]	(None, 72, 72, 32)	0	batch_normalization
conv2d_2 (Conv2D)	(None, 72, 72, 64)	18432	activation_1[0][0]
batch_normalization_2 (BatchNor	(None, 72, 72, 64)	192	conv2d_2[0][0]
activation_2 (Activation) _2[0][0]	(None, 72, 72, 64)	0	batch_normalization

max_pooling2d	(MaxPooling2D)	(None, 35, 35, 64)	0	activation_2[0][0]
conv2d_3	(Conv2D)	(None, 35, 35, 80)	5120	max_pooling2d[0][0]
batch_normalization_3	(BatchNor	(None, 35, 35, 80)	240	conv2d_3[0][0]
activation_3	(Activation)	(None, 35, 35, 80)	0	batch_normalization_3[0][0]
conv2d_4	(Conv2D)	(None, 33, 33, 192)	138240	activation_3[0][0]
batch_normalization_4	(BatchNor	(None, 33, 33, 192)	576	conv2d_4[0][0]
activation_4	(Activation)	(None, 33, 33, 192)	0	batch_normalization_4[0][0]
max_pooling2d_1	(MaxPooling2D)	(None, 16, 16, 192)	0	activation_4[0][0]
conv2d_8	(Conv2D)	(None, 16, 16, 64)	12288	max_pooling2d_1[0]
batch_normalization_8	(BatchNor	(None, 16, 16, 64)	192	conv2d_8[0][0]
activation_8	(Activation)	(None, 16, 16, 64)	0	batch_normalization_8[0][0]
conv2d_6	(Conv2D)	(None, 16, 16, 48)	9216	max_pooling2d_1[0]
conv2d_9	(Conv2D)	(None, 16, 16, 96)	55296	activation_8[0][0]
batch_normalization_6	(BatchNor	(None, 16, 16, 48)	144	conv2d_6[0][0]
batch_normalization_9	(BatchNor	(None, 16, 16, 96)	288	conv2d_9[0][0]
activation_6	(Activation)	(None, 16, 16, 48)	0	batch_normalization_6[0][0]
activation_9	(Activation)	(None, 16, 16, 96)	0	batch_normalization_9[0][0]
average_pooling2d	(AveragePooli	(None, 16, 16, 192)	0	max_pooling2d_1[0]
conv2d_5	(Conv2D)	(None, 16, 16, 64)	12288	max_pooling2d_1[0]
conv2d_7	(Conv2D)	(None, 16, 16, 64)	76800	activation_6[0][0]

conv2d_10 (Conv2D)	(None, 16, 16, 96)	82944	activation_9[0][0]
conv2d_11 (Conv2D)	(None, 16, 16, 32)	6144	average_pooling2d[0][0]
batch_normalization_5 (BatchNor	(None, 16, 16, 64)	192	conv2d_5[0][0]
batch_normalization_7 (BatchNor	(None, 16, 16, 64)	192	conv2d_7[0][0]
batch_normalization_10 (BatchNo	(None, 16, 16, 96)	288	conv2d_10[0][0]
batch_normalization_11 (BatchNo	(None, 16, 16, 32)	96	conv2d_11[0][0]
activation_5 (Activation)	(None, 16, 16, 64)	0	batch_normalization_5[0][0]
activation_7 (Activation)	(None, 16, 16, 64)	0	batch_normalization_7[0][0]
activation_10 (Activation)	(None, 16, 16, 96)	0	batch_normalization_10[0][0]
activation_11 (Activation)	(None, 16, 16, 32)	0	batch_normalization_11[0][0]
mixed0 (Concatenate)	(None, 16, 16, 256)	0	activation_5[0][0] activation_7[0][0] activation_10[0][0] activation_11[0][0]
conv2d_15 (Conv2D)	(None, 16, 16, 64)	16384	mixed0[0][0]
batch_normalization_15 (BatchNo	(None, 16, 16, 64)	192	conv2d_15[0][0]
activation_15 (Activation)	(None, 16, 16, 64)	0	batch_normalization_15[0][0]
conv2d_13 (Conv2D)	(None, 16, 16, 48)	12288	mixed0[0][0]
conv2d_16 (Conv2D)	(None, 16, 16, 96)	55296	activation_15[0][0]
batch_normalization_13 (BatchNo	(None, 16, 16, 48)	144	conv2d_13[0][0]
batch_normalization_16 (BatchNo	(None, 16, 16, 96)	288	conv2d_16[0][0]
activation_13 (Activation)	(None, 16, 16, 48)	0	batch_normalization_13[0][0]
activation_16 (Activation)	(None, 16, 16, 96)	0	batch_normalization_16[0][0]

_16[0][0]

<u>average_pooling2d_1</u> (AveragePoo	(None, 16, 16, 256)	0	mixed0[0][0]
<u>conv2d_12</u> (Conv2D)	(None, 16, 16, 64)	16384	mixed0[0][0]
<u>conv2d_14</u> (Conv2D)	(None, 16, 16, 64)	76800	activation_13[0][0]
<u>conv2d_17</u> (Conv2D)	(None, 16, 16, 96)	82944	activation_16[0][0]
<u>conv2d_18</u> (Conv2D)	(None, 16, 16, 64)	16384	average_pooling2d_1[0][0]
<u>batch_normalization_12</u> (BatchNo	(None, 16, 16, 64)	192	conv2d_12[0][0]
<u>batch_normalization_14</u> (BatchNo	(None, 16, 16, 64)	192	conv2d_14[0][0]
<u>batch_normalization_17</u> (BatchNo	(None, 16, 16, 96)	288	conv2d_17[0][0]
<u>batch_normalization_18</u> (BatchNo	(None, 16, 16, 64)	192	conv2d_18[0][0]
<u>activation_12</u> (Activation)	(None, 16, 16, 64)	0	batch_normalization_12[0][0]
<u>activation_14</u> (Activation)	(None, 16, 16, 64)	0	batch_normalization_14[0][0]
<u>activation_17</u> (Activation)	(None, 16, 16, 96)	0	batch_normalization_17[0][0]
<u>activation_18</u> (Activation)	(None, 16, 16, 64)	0	batch_normalization_18[0][0]
<u>mixed1</u> (Concatenate)	(None, 16, 16, 288)	0	activation_12[0][0] activation_14[0][0] activation_17[0][0] activation_18[0][0]
<u>conv2d_22</u> (Conv2D)	(None, 16, 16, 64)	18432	mixed1[0][0]
<u>batch_normalization_22</u> (BatchNo	(None, 16, 16, 64)	192	conv2d_22[0][0]
<u>activation_22</u> (Activation)	(None, 16, 16, 64)	0	batch_normalization_22[0][0]
<u>conv2d_20</u> (Conv2D)	(None, 16, 16, 48)	13824	mixed1[0][0]
<u>conv2d_23</u> (Conv2D)	(None, 16, 16, 96)	55296	activation_22[0][0]

batch_normalization_20 (BatchNo (None, 16, 16, 48)	144	conv2d_20[0][0]
batch_normalization_23 (BatchNo (None, 16, 16, 96)	288	conv2d_23[0][0]
activation_20 (Activation) (None, 16, 16, 48)	0	batch_normalization_20[0][0]
activation_23 (Activation) (None, 16, 16, 96)	0	batch_normalization_23[0][0]
average_pooling2d_2 (AveragePoo (None, 16, 16, 288)	0	mixed1[0][0]
conv2d_19 (Conv2D) (None, 16, 16, 64)	18432	mixed1[0][0]
conv2d_21 (Conv2D) (None, 16, 16, 64)	76800	activation_20[0][0]
conv2d_24 (Conv2D) (None, 16, 16, 96)	82944	activation_23[0][0]
conv2d_25 (Conv2D) (None, 16, 16, 64)	18432	average_pooling2d_2[0][0]
batch_normalization_19 (BatchNo (None, 16, 16, 64)	192	conv2d_19[0][0]
batch_normalization_21 (BatchNo (None, 16, 16, 64)	192	conv2d_21[0][0]
batch_normalization_24 (BatchNo (None, 16, 16, 96)	288	conv2d_24[0][0]
batch_normalization_25 (BatchNo (None, 16, 16, 64)	192	conv2d_25[0][0]
activation_19 (Activation) (None, 16, 16, 64)	0	batch_normalization_19[0][0]
activation_21 (Activation) (None, 16, 16, 64)	0	batch_normalization_21[0][0]
activation_24 (Activation) (None, 16, 16, 96)	0	batch_normalization_24[0][0]
activation_25 (Activation) (None, 16, 16, 64)	0	batch_normalization_25[0][0]
mixed2 (Concatenate) (None, 16, 16, 288)	0	activation_19[0][0] activation_21[0][0] activation_24[0][0] activation_25[0][0]
conv2d_27 (Conv2D) (None, 16, 16, 64)	18432	mixed2[0][0]
batch_normalization_27 (BatchNo (None, 16, 16, 64)	192	conv2d_27[0][0]

activation_27 (Activation)	(None, 16, 16, 64)	0	batch_normalization_27[0][0]
conv2d_28 (Conv2D)	(None, 16, 16, 96)	55296	activation_27[0][0]
batch_normalization_28 (BatchNormalisation)	(None, 16, 16, 96)	288	conv2d_28[0][0]
activation_28 (Activation)	(None, 16, 16, 96)	0	batch_normalization_28[0][0]
conv2d_26 (Conv2D)	(None, 7, 7, 384)	995328	mixed2[0][0]
conv2d_29 (Conv2D)	(None, 7, 7, 96)	82944	activation_28[0][0]
batch_normalization_26 (BatchNormalisation)	(None, 7, 7, 384)	1152	conv2d_26[0][0]
batch_normalization_29 (BatchNormalisation)	(None, 7, 7, 96)	288	conv2d_29[0][0]
activation_26 (Activation)	(None, 7, 7, 384)	0	batch_normalization_26[0][0]
activation_29 (Activation)	(None, 7, 7, 96)	0	batch_normalization_29[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 7, 7, 288)	0	mixed2[0][0]
mixed3 (Concatenate)	(None, 7, 7, 768)	0	activation_26[0][0] activation_29[0][0] max_pooling2d_2[0]
conv2d_34 (Conv2D)	(None, 7, 7, 128)	98304	mixed3[0][0]
batch_normalization_34 (BatchNormalisation)	(None, 7, 7, 128)	384	conv2d_34[0][0]
activation_34 (Activation)	(None, 7, 7, 128)	0	batch_normalization_34[0][0]
conv2d_35 (Conv2D)	(None, 7, 7, 128)	114688	activation_34[0][0]
batch_normalization_35 (BatchNormalisation)	(None, 7, 7, 128)	384	conv2d_35[0][0]
activation_35 (Activation)	(None, 7, 7, 128)	0	batch_normalization_35[0][0]
conv2d_31 (Conv2D)	(None, 7, 7, 128)	98304	mixed3[0][0]
conv2d_36 (Conv2D)	(None, 7, 7, 128)	114688	activation_35[0][0]

batch_normalization_31 (BatchNo (None, 7, 7, 128))	384	conv2d_31[0][0]
batch_normalization_36 (BatchNo (None, 7, 7, 128))	384	conv2d_36[0][0]
activation_31 (Activation) (None, 7, 7, 128)	0	batch_normalization_31[0][0]
activation_36 (Activation) (None, 7, 7, 128)	0	batch_normalization_36[0][0]
conv2d_32 (Conv2D) (None, 7, 7, 128)	114688	activation_31[0][0]
conv2d_37 (Conv2D) (None, 7, 7, 128)	114688	activation_36[0][0]
batch_normalization_32 (BatchNo (None, 7, 7, 128))	384	conv2d_32[0][0]
batch_normalization_37 (BatchNo (None, 7, 7, 128))	384	conv2d_37[0][0]
activation_32 (Activation) (None, 7, 7, 128)	0	batch_normalization_32[0][0]
activation_37 (Activation) (None, 7, 7, 128)	0	batch_normalization_37[0][0]
average_pooling2d_3 (AveragePoo (None, 7, 7, 768))	0	mixed3[0][0]
conv2d_30 (Conv2D) (None, 7, 7, 192)	147456	mixed3[0][0]
conv2d_33 (Conv2D) (None, 7, 7, 192)	172032	activation_32[0][0]
conv2d_38 (Conv2D) (None, 7, 7, 192)	172032	activation_37[0][0]
conv2d_39 (Conv2D) (None, 7, 7, 192)	147456	average_pooling2d_3[0][0]
batch_normalization_30 (BatchNo (None, 7, 7, 192))	576	conv2d_30[0][0]
batch_normalization_33 (BatchNo (None, 7, 7, 192))	576	conv2d_33[0][0]
batch_normalization_38 (BatchNo (None, 7, 7, 192))	576	conv2d_38[0][0]
batch_normalization_39 (BatchNo (None, 7, 7, 192))	576	conv2d_39[0][0]
activation_30 (Activation) (None, 7, 7, 192)	0	batch_normalization_30[0][0]
activation_33 (Activation) (None, 7, 7, 192)	0	batch_normalization_33[0][0]

activation_38 (Activation)	(None, 7, 7, 192)	0	batch_normalization_38[0][0]
activation_39 (Activation)	(None, 7, 7, 192)	0	batch_normalization_39[0][0]
mixed4 (Concatenate)	(None, 7, 7, 768)	0	activation_30[0][0] activation_33[0][0] activation_38[0][0] activation_39[0][0]
conv2d_44 (Conv2D)	(None, 7, 7, 160)	122880	mixed4[0][0]
batch_normalization_44 (BatchNo)	(None, 7, 7, 160)	480	conv2d_44[0][0]
activation_44 (Activation)	(None, 7, 7, 160)	0	batch_normalization_44[0][0]
conv2d_45 (Conv2D)	(None, 7, 7, 160)	179200	activation_44[0][0]
batch_normalization_45 (BatchNo)	(None, 7, 7, 160)	480	conv2d_45[0][0]
activation_45 (Activation)	(None, 7, 7, 160)	0	batch_normalization_45[0][0]
conv2d_41 (Conv2D)	(None, 7, 7, 160)	122880	mixed4[0][0]
conv2d_46 (Conv2D)	(None, 7, 7, 160)	179200	activation_45[0][0]
batch_normalization_41 (BatchNo)	(None, 7, 7, 160)	480	conv2d_41[0][0]
batch_normalization_46 (BatchNo)	(None, 7, 7, 160)	480	conv2d_46[0][0]
activation_41 (Activation)	(None, 7, 7, 160)	0	batch_normalization_41[0][0]
activation_46 (Activation)	(None, 7, 7, 160)	0	batch_normalization_46[0][0]
conv2d_42 (Conv2D)	(None, 7, 7, 160)	179200	activation_41[0][0]
conv2d_47 (Conv2D)	(None, 7, 7, 160)	179200	activation_46[0][0]
batch_normalization_42 (BatchNo)	(None, 7, 7, 160)	480	conv2d_42[0][0]
batch_normalization_47 (BatchNo)	(None, 7, 7, 160)	480	conv2d_47[0][0]
activation_42 (Activation)	(None, 7, 7, 160)	0	batch_normalization

_42[0][0]

<u>activation_47</u> (Activation)	(None, 7, 7, 160)	0	batch_normalization_47[0][0]
<u>average_pooling2d_4</u> (AveragePooling2D)	(None, 7, 7, 768)	0	mixed4[0][0]
<u>conv2d_40</u> (Conv2D)	(None, 7, 7, 192)	147456	mixed4[0][0]
<u>conv2d_43</u> (Conv2D)	(None, 7, 7, 192)	215040	activation_42[0][0]
<u>conv2d_48</u> (Conv2D)	(None, 7, 7, 192)	215040	activation_47[0][0]
<u>conv2d_49</u> (Conv2D)	(None, 7, 7, 192)	147456	average_pooling2d_4[0][0]
<u>batch_normalization_40</u> (BatchNormalization)	(None, 7, 7, 192)	576	conv2d_40[0][0]
<u>batch_normalization_43</u> (BatchNormalization)	(None, 7, 7, 192)	576	conv2d_43[0][0]
<u>batch_normalization_48</u> (BatchNormalization)	(None, 7, 7, 192)	576	conv2d_48[0][0]
<u>batch_normalization_49</u> (BatchNormalization)	(None, 7, 7, 192)	576	conv2d_49[0][0]
<u>activation_40</u> (Activation)	(None, 7, 7, 192)	0	batch_normalization_40[0][0]
<u>activation_43</u> (Activation)	(None, 7, 7, 192)	0	batch_normalization_43[0][0]
<u>activation_48</u> (Activation)	(None, 7, 7, 192)	0	batch_normalization_48[0][0]
<u>activation_49</u> (Activation)	(None, 7, 7, 192)	0	batch_normalization_49[0][0]
<u>mixed5</u> (Concatenate)	(None, 7, 7, 768)	0	activation_40[0][0] activation_43[0][0] activation_48[0][0] activation_49[0][0]
<u>conv2d_54</u> (Conv2D)	(None, 7, 7, 160)	122880	mixed5[0][0]
<u>batch_normalization_54</u> (BatchNormalization)	(None, 7, 7, 160)	480	conv2d_54[0][0]
<u>activation_54</u> (Activation)	(None, 7, 7, 160)	0	batch_normalization_54[0][0]
<u>conv2d_55</u> (Conv2D)	(None, 7, 7, 160)	179200	activation_54[0][0]

batch_normalization_55 (BatchNo (None, 7, 7, 160)	480	conv2d_55[0][0]
activation_55 (Activation) (None, 7, 7, 160)	0	batch_normalization_55[0][0]
conv2d_51 (Conv2D) (None, 7, 7, 160)	122880	mixed5[0][0]
conv2d_56 (Conv2D) (None, 7, 7, 160)	179200	activation_55[0][0]
batch_normalization_51 (BatchNo (None, 7, 7, 160)	480	conv2d_51[0][0]
batch_normalization_56 (BatchNo (None, 7, 7, 160)	480	conv2d_56[0][0]
activation_51 (Activation) (None, 7, 7, 160)	0	batch_normalization_51[0][0]
activation_56 (Activation) (None, 7, 7, 160)	0	batch_normalization_56[0][0]
conv2d_52 (Conv2D) (None, 7, 7, 160)	179200	activation_51[0][0]
conv2d_57 (Conv2D) (None, 7, 7, 160)	179200	activation_56[0][0]
batch_normalization_52 (BatchNo (None, 7, 7, 160)	480	conv2d_52[0][0]
batch_normalization_57 (BatchNo (None, 7, 7, 160)	480	conv2d_57[0][0]
activation_52 (Activation) (None, 7, 7, 160)	0	batch_normalization_52[0][0]
activation_57 (Activation) (None, 7, 7, 160)	0	batch_normalization_57[0][0]
average_pooling2d_5 (AveragePoo (None, 7, 7, 768)	0	mixed5[0][0]
conv2d_50 (Conv2D) (None, 7, 7, 192)	147456	mixed5[0][0]
conv2d_53 (Conv2D) (None, 7, 7, 192)	215040	activation_52[0][0]
conv2d_58 (Conv2D) (None, 7, 7, 192)	215040	activation_57[0][0]
conv2d_59 (Conv2D) (None, 7, 7, 192)	147456	average_pooling2d_5[0][0]
batch_normalization_50 (BatchNo (None, 7, 7, 192)	576	conv2d_50[0][0]
batch_normalization_53 (BatchNo (None, 7, 7, 192)	576	conv2d_53[0][0]

batch_normalization_58 (BatchNo (None, 7, 7, 192)	576	conv2d_58[0][0]
batch_normalization_59 (BatchNo (None, 7, 7, 192)	576	conv2d_59[0][0]
activation_50 (Activation) (None, 7, 7, 192)	0	batch_normalization_50[0][0]
activation_53 (Activation) (None, 7, 7, 192)	0	batch_normalization_53[0][0]
activation_58 (Activation) (None, 7, 7, 192)	0	batch_normalization_58[0][0]
activation_59 (Activation) (None, 7, 7, 192)	0	batch_normalization_59[0][0]
mixed6 (Concatenate) (None, 7, 7, 768)	0	activation_50[0][0] activation_53[0][0] activation_58[0][0] activation_59[0][0]
conv2d_64 (Conv2D) (None, 7, 7, 192)	147456	mixed6[0][0]
batch_normalization_64 (BatchNo (None, 7, 7, 192)	576	conv2d_64[0][0]
activation_64 (Activation) (None, 7, 7, 192)	0	batch_normalization_64[0][0]
conv2d_65 (Conv2D) (None, 7, 7, 192)	258048	activation_64[0][0]
batch_normalization_65 (BatchNo (None, 7, 7, 192)	576	conv2d_65[0][0]
activation_65 (Activation) (None, 7, 7, 192)	0	batch_normalization_65[0][0]
conv2d_61 (Conv2D) (None, 7, 7, 192)	147456	mixed6[0][0]
conv2d_66 (Conv2D) (None, 7, 7, 192)	258048	activation_65[0][0]
batch_normalization_61 (BatchNo (None, 7, 7, 192)	576	conv2d_61[0][0]
batch_normalization_66 (BatchNo (None, 7, 7, 192)	576	conv2d_66[0][0]
activation_61 (Activation) (None, 7, 7, 192)	0	batch_normalization_61[0][0]
activation_66 (Activation) (None, 7, 7, 192)	0	batch_normalization_66[0][0]

InceptionV3				
conv2d_62 (Conv2D)	(None, 7, 7, 192)	258048	activation_61[0][0]	
conv2d_67 (Conv2D)	(None, 7, 7, 192)	258048	activation_66[0][0]	
batch_normalization_62 (BatchNo)	(None, 7, 7, 192)	576	conv2d_62[0][0]	
batch_normalization_67 (BatchNo)	(None, 7, 7, 192)	576	conv2d_67[0][0]	
activation_62 (Activation)	(None, 7, 7, 192)	0	batch_normalization_62[0][0]	
activation_67 (Activation)	(None, 7, 7, 192)	0	batch_normalization_67[0][0]	
average_pooling2d_6 (AveragePoo)	(None, 7, 7, 768)	0	mixed6[0][0]	
conv2d_60 (Conv2D)	(None, 7, 7, 192)	147456	mixed6[0][0]	
conv2d_63 (Conv2D)	(None, 7, 7, 192)	258048	activation_62[0][0]	
conv2d_68 (Conv2D)	(None, 7, 7, 192)	258048	activation_67[0][0]	
conv2d_69 (Conv2D)	(None, 7, 7, 192)	147456	average_pooling2d_6[0][0]	
batch_normalization_60 (BatchNo)	(None, 7, 7, 192)	576	conv2d_60[0][0]	
batch_normalization_63 (BatchNo)	(None, 7, 7, 192)	576	conv2d_63[0][0]	
batch_normalization_68 (BatchNo)	(None, 7, 7, 192)	576	conv2d_68[0][0]	
batch_normalization_69 (BatchNo)	(None, 7, 7, 192)	576	conv2d_69[0][0]	
activation_60 (Activation)	(None, 7, 7, 192)	0	batch_normalization_60[0][0]	
activation_63 (Activation)	(None, 7, 7, 192)	0	batch_normalization_63[0][0]	
activation_68 (Activation)	(None, 7, 7, 192)	0	batch_normalization_68[0][0]	
activation_69 (Activation)	(None, 7, 7, 192)	0	batch_normalization_69[0][0]	
mixed7 (Concatenate)	(None, 7, 7, 768)	0	activation_60[0][0] activation_63[0][0] activation_68[0][0] activation_69[0][0]	

flatten (Flatten)	(None, 37632)	0	mixed7[0][0]
dense (Dense)	(None, 256)	9634048	flatten[0][0]
dense_1 (Dense)	(None, 512)	131584	dense[0][0]
dense_2 (Dense)	(None, 1024)	525312	dense_1[0][0]
dense_3 (Dense)	(None, 1024)	1049600	dense_2[0][0]
dropout (Dropout)	(None, 1024)	0	dense_3[0][0]
dense_4 (Dense)	(None, 5)	5125	dropout[0][0]
<hr/>			
<hr/>			
Total params: 20,320,933			
Trainable params: 11,345,669			
Non-trainable params: 8,975,264			

('Failed to import pydot. You must `pip install pydot` and install graphviz ([http://graphviz.gitlab.io/download/](https://graphviz.gitlab.io/download/)), ', 'for `pydotprint` to work.')

Fitting model

```
In [22]: step_size_train = train_age_generator.n//train_age_generator.batch_size

history = model.fit_generator(generator=train_age_generator,
                               validation_data=val_age_generator,
                               steps_per_epoch=step_size_train,
                               epochs=100,
                               validation_steps=None,
                               verbose=2,
                               callbacks=[earlystop, plateau]) # this will take some
```

```
c:\Miniconda\lib\site-packages\tensorflow\python\keras\engine\training.py:1844: User
Warning: `Model.fit_generator` is deprecated and will be removed in a future versio
n. Please use `Model.fit`, which supports generators.
    warnings.warn(``Model.fit_generator`` is deprecated and '
Epoch 1/100
678/678 - 460s - loss: 1.2357 - accuracy: 0.4884 - val_loss: 1.1963 - val_accuracy:
0.5065
Epoch 2/100
678/678 - 267s - loss: 1.1597 - accuracy: 0.5228 - val_loss: 1.0795 - val_accuracy:
0.5581
Epoch 3/100
678/678 - 269s - loss: 1.1389 - accuracy: 0.5316 - val_loss: 1.1028 - val_accuracy:
0.5455
Epoch 4/100
678/678 - 269s - loss: 1.1264 - accuracy: 0.5396 - val_loss: 1.0557 - val_accuracy:
0.5662
Epoch 5/100
678/678 - 271s - loss: 1.1095 - accuracy: 0.5397 - val_loss: 1.0515 - val_accuracy:
0.5743
Epoch 6/100
678/678 - 268s - loss: 1.1024 - accuracy: 0.5447 - val_loss: 1.0366 - val_accuracy:
0.5769
Epoch 7/100
678/678 - 267s - loss: 1.0901 - accuracy: 0.5494 - val_loss: 1.0626 - val_accuracy:
```

0.5687
Epoch 8/100
678/678 - 263s - loss: 1.0867 - accuracy: 0.5551 - val_loss: 1.0153 - val_accuracy: 0.5872
Epoch 9/100
678/678 - 263s - loss: 1.0800 - accuracy: 0.5530 - val_loss: 1.0196 - val_accuracy: 0.5953
Epoch 10/100
678/678 - 267s - loss: 1.0731 - accuracy: 0.5571 - val_loss: 1.0648 - val_accuracy: 0.5732
Epoch 11/100
678/678 - 268s - loss: 1.0691 - accuracy: 0.5599 - val_loss: 1.0175 - val_accuracy: 0.5809
Epoch 12/100
678/678 - 266s - loss: 1.0629 - accuracy: 0.5606 - val_loss: 1.0138 - val_accuracy: 0.5868
Epoch 13/100
678/678 - 266s - loss: 1.0567 - accuracy: 0.5630 - val_loss: 1.0121 - val_accuracy: 0.5868
Epoch 14/100
678/678 - 265s - loss: 1.0541 - accuracy: 0.5670 - val_loss: 1.0278 - val_accuracy: 0.5820
Epoch 15/100
678/678 - 263s - loss: 1.0556 - accuracy: 0.5629 - val_loss: 1.0230 - val_accuracy: 0.5916
Epoch 16/100
678/678 - 261s - loss: 1.0509 - accuracy: 0.5621 - val_loss: 1.0115 - val_accuracy: 0.5901
Epoch 17/100
678/678 - 261s - loss: 1.0421 - accuracy: 0.5693 - val_loss: 1.0063 - val_accuracy: 0.5890
Epoch 18/100
678/678 - 259s - loss: 1.0450 - accuracy: 0.5696 - val_loss: 1.0124 - val_accuracy: 0.5898
Epoch 19/100
678/678 - 264s - loss: 1.0384 - accuracy: 0.5697 - val_loss: 0.9880 - val_accuracy: 0.6015
Epoch 20/100
678/678 - 266s - loss: 1.0398 - accuracy: 0.5679 - val_loss: 1.0015 - val_accuracy: 0.5853
Epoch 21/100
678/678 - 263s - loss: 1.0342 - accuracy: 0.5706 - val_loss: 1.0002 - val_accuracy: 0.5875
Epoch 22/100
678/678 - 264s - loss: 1.0290 - accuracy: 0.5742 - val_loss: 0.9874 - val_accuracy: 0.5997
Epoch 23/100
678/678 - 266s - loss: 1.0269 - accuracy: 0.5720 - val_loss: 0.9892 - val_accuracy: 0.5993
Epoch 24/100
678/678 - 265s - loss: 1.0283 - accuracy: 0.5742 - val_loss: 0.9896 - val_accuracy: 0.6041
Epoch 25/100
678/678 - 264s - loss: 1.0227 - accuracy: 0.5771 - val_loss: 0.9934 - val_accuracy: 0.6004
Epoch 26/100
678/678 - 258s - loss: 1.0224 - accuracy: 0.5759 - val_loss: 0.9986 - val_accuracy: 0.6012
Epoch 27/100
678/678 - 259s - loss: 1.0193 - accuracy: 0.5755 - val_loss: 1.0206 - val_accuracy: 0.5842

Epoch 00027: ReduceLROnPlateau reducing learning rate to 9.99999747378752e-06.
Epoch 28/100
678/678 - 263s - loss: 0.9951 - accuracy: 0.5839 - val_loss: 0.9889 - val_accuracy: 0.6038
Epoch 29/100
678/678 - 267s - loss: 0.9937 - accuracy: 0.5860 - val_loss: 0.9853 - val_accuracy: 0.6049

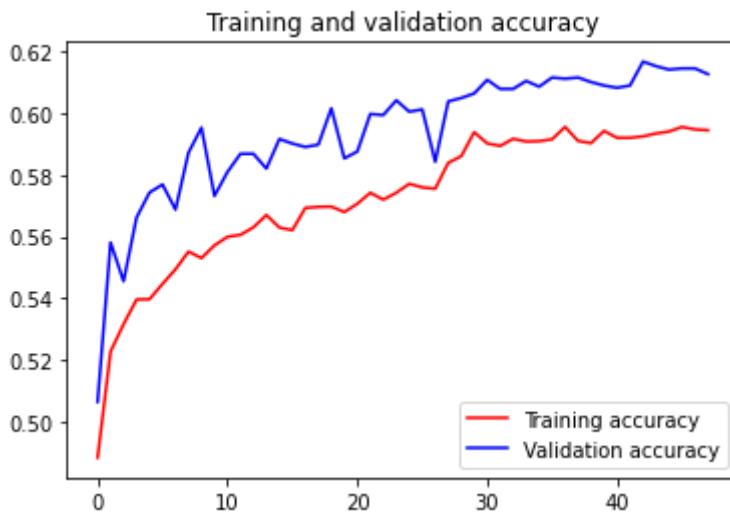
```
Epoch 30/100
678/678 - 265s - loss: 0.9842 - accuracy: 0.5938 - val_loss: 0.9906 - val_accuracy: 0.6063
Epoch 31/100
678/678 - 263s - loss: 0.9855 - accuracy: 0.5902 - val_loss: 0.9863 - val_accuracy: 0.6108
Epoch 32/100
678/678 - 264s - loss: 0.9855 - accuracy: 0.5894 - val_loss: 0.9841 - val_accuracy: 0.6078
Epoch 33/100
678/678 - 265s - loss: 0.9848 - accuracy: 0.5917 - val_loss: 0.9847 - val_accuracy: 0.6078
Epoch 34/100
678/678 - 264s - loss: 0.9865 - accuracy: 0.5908 - val_loss: 0.9882 - val_accuracy: 0.6104
Epoch 35/100
678/678 - 267s - loss: 0.9850 - accuracy: 0.5909 - val_loss: 0.9833 - val_accuracy: 0.6086
Epoch 36/100
678/678 - 259s - loss: 0.9796 - accuracy: 0.5915 - val_loss: 0.9835 - val_accuracy: 0.6115
Epoch 37/100
678/678 - 262s - loss: 0.9771 - accuracy: 0.5955 - val_loss: 0.9844 - val_accuracy: 0.6111
Epoch 38/100
678/678 - 265s - loss: 0.9838 - accuracy: 0.5910 - val_loss: 0.9815 - val_accuracy: 0.6115
Epoch 39/100
678/678 - 267s - loss: 0.9777 - accuracy: 0.5903 - val_loss: 0.9825 - val_accuracy: 0.6100
Epoch 40/100
678/678 - 265s - loss: 0.9797 - accuracy: 0.5942 - val_loss: 0.9829 - val_accuracy: 0.6089
Epoch 41/100
678/678 - 266s - loss: 0.9758 - accuracy: 0.5920 - val_loss: 0.9836 - val_accuracy: 0.6082
Epoch 42/100
678/678 - 265s - loss: 0.9797 - accuracy: 0.5920 - val_loss: 0.9827 - val_accuracy: 0.6089
Epoch 43/100
678/678 - 262s - loss: 0.9773 - accuracy: 0.5924 - val_loss: 0.9817 - val_accuracy: 0.6167

Epoch 00043: ReduceLROnPlateau reducing learning rate to 9.99999747378752e-07.
Epoch 44/100
678/678 - 259s - loss: 0.9766 - accuracy: 0.5934 - val_loss: 0.9815 - val_accuracy: 0.6152
Epoch 45/100
678/678 - 259s - loss: 0.9734 - accuracy: 0.5940 - val_loss: 0.9817 - val_accuracy: 0.6141
Epoch 46/100
678/678 - 263s - loss: 0.9754 - accuracy: 0.5955 - val_loss: 0.9817 - val_accuracy: 0.6144
Epoch 47/100
678/678 - 265s - loss: 0.9756 - accuracy: 0.5947 - val_loss: 0.9823 - val_accuracy: 0.6144
Epoch 48/100
678/678 - 266s - loss: 0.9729 - accuracy: 0.5944 - val_loss: 0.9821 - val_accuracy: 0.6126
Restoring model weights from the end of the best epoch.

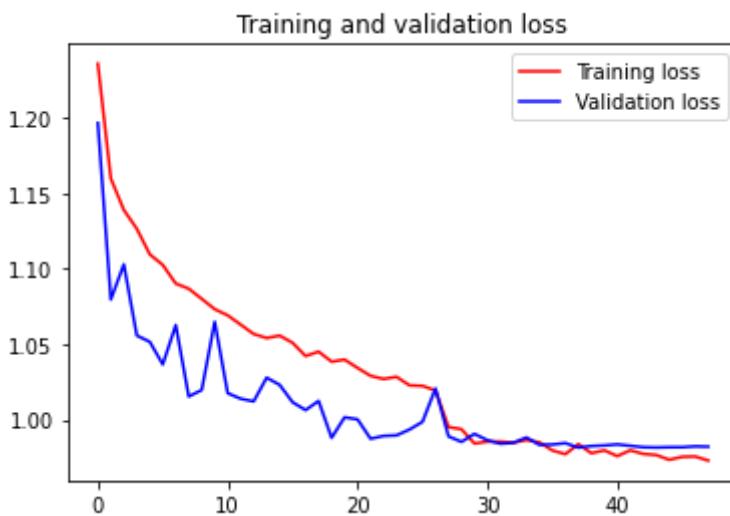
Epoch 00048: ReduceLROnPlateau reducing learning rate to 9.99999974752428e-08.
Epoch 00048: early stopping
```

Model result

In [23]: `plot_acc(history)`



<Figure size 432x288 with 0 Axes>

In [24]: `plot_loss(history)`

<Figure size 432x288 with 0 Axes>

In [25]:

```
# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(test_age_generator, batch_size=32)
print("test loss, test acc:", results)

# Generate predictions (probabilities -- the output of the last layer)
# on new data using `predict`
print("Generate predictions for 3 samples")
predictions = model.predict(test_age_generator)
print("predictions shape:", predictions.shape)
```

```
Evaluate on test data
85/85 [=====] - 32s 371ms/step - loss: 1.0047 - accuracy: 0.5874
test loss, test acc: [1.0046792030334473, 0.5873894095420837]
Generate predictions for 3 samples
predictions shape: (2712, 5)
```

In [37]:

```
import numpy as np

labels = ["(0, 18]", "(18, 30]", "(30, 40]", "(40, 60]", "(60, +]"]

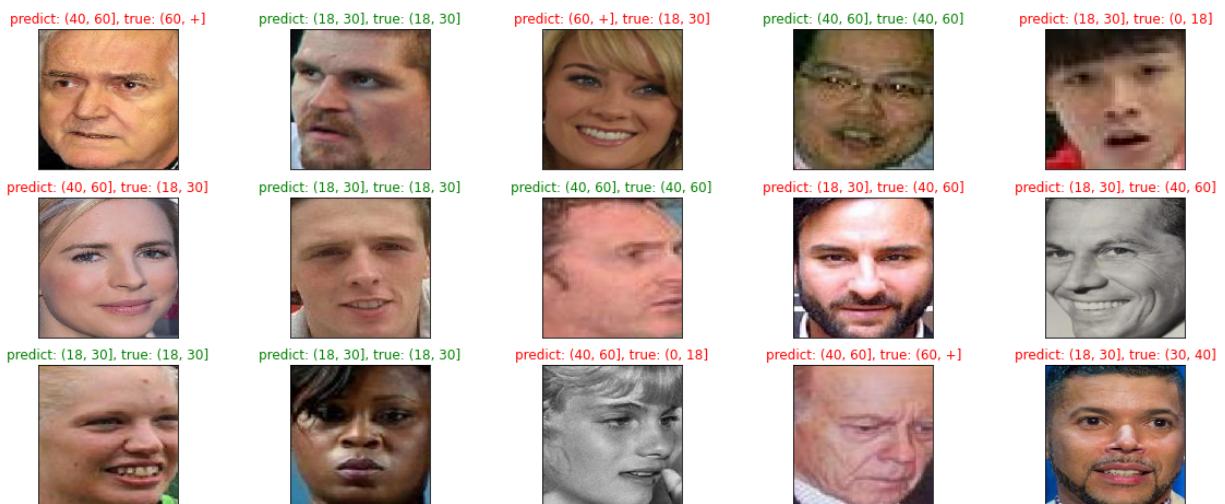
x, y = test_age_generator.next()

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
```

```

for i, index in enumerate(np.random.choice(x.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(x[index]))
    predict_index = np.argmax(predictions[index])
    true_index = np.argmax(y[index])
    # Set the title for each image
    ax.set_title("predict: {}, true: {}".format(labels[predict_index],
                                                labels[true_index]),
                color=("green" if predict_index == true_index else
                       "red"))
plt.show()

```



Model of Gender

Set layers to be non-trainable for pre-trained model

```
In [45]: for layer in pre_trained_model.layers:
    layer.trainable = False
```

Model summary of Inception v3

```
In [46]: pre_trained_model.summary()
plot_model(pre_trained_model, to_file='inception_v3_model.png', show_shapes=False, s
```

Model: "inception_v3"

Layer (type)	Output Shape	Param #	Connected to
<hr/>			
<hr/>			
input_2 (InputLayer)	(None, 150, 150, 3)	0	
conv2d_94 (Conv2D)	(None, 74, 74, 32)	864	input_2[0][0]
batch_normalization_94 (BatchNo	(None, 74, 74, 32)	96	conv2d_94[0][0]
activation_94 (Activation)	(None, 74, 74, 32)	0	batch_normalization_94[0][0]
conv2d_95 (Conv2D)	(None, 72, 72, 32)	9216	activation_94[0][0]
<hr/>			

batch_normalization_95 (BatchNo (None, 72, 72, 32)	96	conv2d_95[0][0]
activation_95 (Activation) (None, 72, 72, 32)	0	batch_normalization_95[0][0]
conv2d_96 (Conv2D) (None, 72, 72, 64)	18432	activation_95[0][0]
batch_normalization_96 (BatchNo (None, 72, 72, 64)	192	conv2d_96[0][0]
activation_96 (Activation) (None, 72, 72, 64)	0	batch_normalization_96[0][0]
max_pooling2d_4 (MaxPooling2D) (None, 35, 35, 64)	0	activation_96[0][0]
conv2d_97 (Conv2D) (None, 35, 35, 80)	5120	max_pooling2d_4[0]
batch_normalization_97 (BatchNo (None, 35, 35, 80)	240	conv2d_97[0][0]
activation_97 (Activation) (None, 35, 35, 80)	0	batch_normalization_97[0][0]
conv2d_98 (Conv2D) (None, 33, 33, 192)	138240	activation_97[0][0]
batch_normalization_98 (BatchNo (None, 33, 33, 192)	576	conv2d_98[0][0]
activation_98 (Activation) (None, 33, 33, 192)	0	batch_normalization_98[0][0]
max_pooling2d_5 (MaxPooling2D) (None, 16, 16, 192)	0	activation_98[0][0]
conv2d_102 (Conv2D) (None, 16, 16, 64)	12288	max_pooling2d_5[0]
batch_normalization_102 (BatchN (None, 16, 16, 64)	192	conv2d_102[0][0]
activation_102 (Activation) (None, 16, 16, 64)	0	batch_normalization_102[0][0]
conv2d_100 (Conv2D) (None, 16, 16, 48)	9216	max_pooling2d_5[0]
conv2d_103 (Conv2D) (None, 16, 16, 96)	55296	activation_102[0]
batch_normalization_100 (BatchN (None, 16, 16, 48)	144	conv2d_100[0][0]
batch_normalization_103 (BatchN (None, 16, 16, 96)	288	conv2d_103[0][0]

activation_100 (Activation) _100[0][0]	(None, 16, 16, 48)	0	batch_normalization
activation_103 (Activation) _103[0][0]	(None, 16, 16, 96)	0	batch_normalization
average_pooling2d_12 (AveragePo [0]	(None, 16, 16, 192)	0	max_pooling2d_5[0]
conv2d_99 (Conv2D) [0]	(None, 16, 16, 64)	12288	max_pooling2d_5[0]
conv2d_101 (Conv2D) [0]	(None, 16, 16, 64)	76800	activation_100[0]
conv2d_104 (Conv2D) [0]	(None, 16, 16, 96)	82944	activation_103[0]
conv2d_105 (Conv2D) 2[0][0]	(None, 16, 16, 32)	6144	average_pooling2d_1
batch_normalization_99 (BatchNo [0]	(None, 16, 16, 64)	192	conv2d_99[0][0]
batch_normalization_101 (BatchN) [0]	(None, 16, 16, 64)	192	conv2d_101[0][0]
batch_normalization_104 (BatchN) [0]	(None, 16, 16, 96)	288	conv2d_104[0][0]
batch_normalization_105 (BatchN) [0]	(None, 16, 16, 32)	96	conv2d_105[0][0]
activation_99 (Activation) _99[0][0]	(None, 16, 16, 64)	0	batch_normalization
activation_101 (Activation) _101[0][0]	(None, 16, 16, 64)	0	batch_normalization
activation_104 (Activation) _104[0][0]	(None, 16, 16, 96)	0	batch_normalization
activation_105 (Activation) _105[0][0]	(None, 16, 16, 32)	0	batch_normalization
mixed0 (Concatenate) [0]	(None, 16, 16, 256)	0	activation_99[0][0]
[0]			activation_101[0]
[0]			activation_104[0]
[0]			activation_105[0]
conv2d_109 (Conv2D)	(None, 16, 16, 64)	16384	mixed0[0][0]
batch_normalization_109 (BatchN) [0]	(None, 16, 16, 64)	192	conv2d_109[0][0]

<u>activation_109</u> (Activation)	(None, 16, 16, 64)	0	batch_normalization_109[0][0]
<u>conv2d_107</u> (Conv2D)	(None, 16, 16, 48)	12288	mixed0[0][0]
<u>conv2d_110</u> (Conv2D)	(None, 16, 16, 96)	55296	activation_109[0][0]
<u>batch_normalization_107</u> (BatchN)	(None, 16, 16, 48)	144	conv2d_107[0][0]
<u>batch_normalization_110</u> (BatchN)	(None, 16, 16, 96)	288	conv2d_110[0][0]
<u>activation_107</u> (Activation)	(None, 16, 16, 48)	0	batch_normalization_107[0][0]
<u>activation_110</u> (Activation)	(None, 16, 16, 96)	0	batch_normalization_110[0][0]
<u>average_pooling2d_13</u> (AveragePo	(None, 16, 16, 256)	0	mixed0[0][0]
<u>conv2d_106</u> (Conv2D)	(None, 16, 16, 64)	16384	mixed0[0][0]
<u>conv2d_108</u> (Conv2D)	(None, 16, 16, 64)	76800	activation_107[0][0]
<u>conv2d_111</u> (Conv2D)	(None, 16, 16, 96)	82944	activation_110[0][0]
<u>conv2d_112</u> (Conv2D)	(None, 16, 16, 64)	16384	average_pooling2d_13[0][0]
<u>batch_normalization_106</u> (BatchN)	(None, 16, 16, 64)	192	conv2d_106[0][0]
<u>batch_normalization_108</u> (BatchN)	(None, 16, 16, 64)	192	conv2d_108[0][0]
<u>batch_normalization_111</u> (BatchN)	(None, 16, 16, 96)	288	conv2d_111[0][0]
<u>batch_normalization_112</u> (BatchN)	(None, 16, 16, 64)	192	conv2d_112[0][0]
<u>activation_106</u> (Activation)	(None, 16, 16, 64)	0	batch_normalization_106[0][0]
<u>activation_108</u> (Activation)	(None, 16, 16, 64)	0	batch_normalization_108[0][0]
<u>activation_111</u> (Activation)	(None, 16, 16, 96)	0	batch_normalization_111[0][0]

activation_112 (Activation) _112[0][0]	(None, 16, 16, 64) 0	batch_normalization
mixed1 (Concatenate) [0]	(None, 16, 16, 288) 0	activation_106[0]
[0]		activation_108[0]
[0]		activation_111[0]
[0]		activation_112[0]
conv2d_116 (Conv2D)	(None, 16, 16, 64) 18432	mixed1[0][0]
batch_normalization_116 (BatchN (None, 16, 16, 64)	192	conv2d_116[0][0]
activation_116 (Activation) _116[0][0]	(None, 16, 16, 64) 0	batch_normalization
conv2d_114 (Conv2D)	(None, 16, 16, 48) 13824	mixed1[0][0]
conv2d_117 (Conv2D)	(None, 16, 16, 96) 55296	activation_116[0] [0]
batch_normalization_114 (BatchN (None, 16, 16, 48)	144	conv2d_114[0][0]
batch_normalization_117 (BatchN (None, 16, 16, 96)	288	conv2d_117[0][0]
activation_114 (Activation) _114[0][0]	(None, 16, 16, 48) 0	batch_normalization
activation_117 (Activation) _117[0][0]	(None, 16, 16, 96) 0	batch_normalization
average_pooling2d_14 (AveragePo	(None, 16, 16, 288) 0	mixed1[0][0]
conv2d_113 (Conv2D)	(None, 16, 16, 64) 18432	mixed1[0][0]
conv2d_115 (Conv2D)	(None, 16, 16, 64) 76800	activation_114[0] [0]
conv2d_118 (Conv2D)	(None, 16, 16, 96) 82944	activation_117[0] [0]
conv2d_119 (Conv2D)	(None, 16, 16, 64) 18432	average_pooling2d_1 4[0][0]
batch_normalization_113 (BatchN (None, 16, 16, 64)	192	conv2d_113[0][0]
batch_normalization_115 (BatchN (None, 16, 16, 64)	192	conv2d_115[0][0]

batch_normalization_118 (BatchN (None, 16, 16, 96)	288	conv2d_118[0][0]
batch_normalization_119 (BatchN (None, 16, 16, 64)	192	conv2d_119[0][0]
activation_113 (Activation) (None, 16, 16, 64)	0	batch_normalization_113[0][0]
activation_115 (Activation) (None, 16, 16, 64)	0	batch_normalization_115[0][0]
activation_118 (Activation) (None, 16, 16, 96)	0	batch_normalization_118[0][0]
activation_119 (Activation) (None, 16, 16, 64)	0	batch_normalization_119[0][0]
mixed2 (Concatenate) [0] [0] [0] [0]	0	activation_113[0] activation_115[0] activation_118[0] activation_119[0]
conv2d_121 (Conv2D) (None, 16, 16, 64)	18432	mixed2[0][0]
batch_normalization_121 (BatchN (None, 16, 16, 64)	192	conv2d_121[0][0]
activation_121 (Activation) (None, 16, 16, 64)	0	batch_normalization_121[0][0]
conv2d_122 (Conv2D) (None, 16, 16, 96)	55296	activation_121[0]
batch_normalization_122 (BatchN (None, 16, 16, 96)	288	conv2d_122[0][0]
activation_122 (Activation) (None, 16, 16, 96)	0	batch_normalization_122[0][0]
conv2d_120 (Conv2D) (None, 7, 7, 384)	995328	mixed2[0][0]
conv2d_123 (Conv2D) (None, 7, 7, 96)	82944	activation_122[0]
batch_normalization_120 (BatchN (None, 7, 7, 384)	1152	conv2d_120[0][0]
batch_normalization_123 (BatchN (None, 7, 7, 96)	288	conv2d_123[0][0]
activation_120 (Activation) (None, 7, 7, 384)	0	batch_normalization_120[0][0]

activation_123 (Activation)	(None, 7, 7, 96)	0	batch_normalization_123[0][0]
max_pooling2d_6 (MaxPooling2D)	(None, 7, 7, 288)	0	mixed2[0][0]
mixed3 (Concatenate)	(None, 7, 7, 768)	0	activation_120[0][0]
			activation_123[0][0]
			max_pooling2d_6[0][0]
conv2d_128 (Conv2D)	(None, 7, 7, 128)	98304	mixed3[0][0]
batch_normalization_128 (BatchN)	(None, 7, 7, 128)	384	conv2d_128[0][0]
activation_128 (Activation)	(None, 7, 7, 128)	0	batch_normalization_128[0][0]
conv2d_129 (Conv2D)	(None, 7, 7, 128)	114688	activation_128[0][0]
batch_normalization_129 (BatchN)	(None, 7, 7, 128)	384	conv2d_129[0][0]
activation_129 (Activation)	(None, 7, 7, 128)	0	batch_normalization_129[0][0]
conv2d_125 (Conv2D)	(None, 7, 7, 128)	98304	mixed3[0][0]
conv2d_130 (Conv2D)	(None, 7, 7, 128)	114688	activation_129[0][0]
batch_normalization_125 (BatchN)	(None, 7, 7, 128)	384	conv2d_125[0][0]
batch_normalization_130 (BatchN)	(None, 7, 7, 128)	384	conv2d_130[0][0]
activation_125 (Activation)	(None, 7, 7, 128)	0	batch_normalization_125[0][0]
activation_130 (Activation)	(None, 7, 7, 128)	0	batch_normalization_130[0][0]
conv2d_126 (Conv2D)	(None, 7, 7, 128)	114688	activation_125[0][0]
conv2d_131 (Conv2D)	(None, 7, 7, 128)	114688	activation_130[0][0]
batch_normalization_126 (BatchN)	(None, 7, 7, 128)	384	conv2d_126[0][0]

batch_normalization_131 (BatchN (None, 7, 7, 128)	384	conv2d_131[0][0]
activation_126 (Activation) (None, 7, 7, 128)	0	batch_normalization_126[0][0]
activation_131 (Activation) (None, 7, 7, 128)	0	batch_normalization_131[0][0]
average_pooling2d_15 (AveragePo (None, 7, 7, 768)	0	mixed3[0][0]
conv2d_124 (Conv2D) (None, 7, 7, 192)	147456	mixed3[0][0]
conv2d_127 (Conv2D) (None, 7, 7, 192)	172032	activation_126[0][0]
conv2d_132 (Conv2D) (None, 7, 7, 192)	172032	activation_131[0][0]
conv2d_133 (Conv2D) (None, 7, 7, 192)	147456	average_pooling2d_15[0][0]
batch_normalization_124 (BatchN (None, 7, 7, 192)	576	conv2d_124[0][0]
batch_normalization_127 (BatchN (None, 7, 7, 192)	576	conv2d_127[0][0]
batch_normalization_132 (BatchN (None, 7, 7, 192)	576	conv2d_132[0][0]
batch_normalization_133 (BatchN (None, 7, 7, 192)	576	conv2d_133[0][0]
activation_124 (Activation) (None, 7, 7, 192)	0	batch_normalization_124[0][0]
activation_127 (Activation) (None, 7, 7, 192)	0	batch_normalization_127[0][0]
activation_132 (Activation) (None, 7, 7, 192)	0	batch_normalization_132[0][0]
activation_133 (Activation) (None, 7, 7, 192)	0	batch_normalization_133[0][0]
mixed4 (Concatenate) [0] (None, 7, 7, 768)	0	activation_124[0][0]
[0]		activation_127[0][0]
[0]		activation_132[0][0]
[0]		activation_133[0][0]
conv2d_138 (Conv2D) (None, 7, 7, 160)	122880	mixed4[0][0]

batch_normalization_138 (BatchN (None, 7, 7, 160))	480	conv2d_138[0][0]
activation_138 (Activation) (None, 7, 7, 160)	0	batch_normalization_138[0][0]
conv2d_139 (Conv2D) (None, 7, 7, 160)	179200	activation_138[0][0]
batch_normalization_139 (BatchN (None, 7, 7, 160))	480	conv2d_139[0][0]
activation_139 (Activation) (None, 7, 7, 160)	0	batch_normalization_139[0][0]
conv2d_135 (Conv2D) (None, 7, 7, 160)	122880	mixed4[0][0]
conv2d_140 (Conv2D) (None, 7, 7, 160)	179200	activation_139[0][0]
batch_normalization_135 (BatchN (None, 7, 7, 160))	480	conv2d_135[0][0]
batch_normalization_140 (BatchN (None, 7, 7, 160))	480	conv2d_140[0][0]
activation_135 (Activation) (None, 7, 7, 160)	0	batch_normalization_135[0][0]
activation_140 (Activation) (None, 7, 7, 160)	0	batch_normalization_140[0][0]
conv2d_136 (Conv2D) (None, 7, 7, 160)	179200	activation_135[0][0]
conv2d_141 (Conv2D) (None, 7, 7, 160)	179200	activation_140[0][0]
batch_normalization_136 (BatchN (None, 7, 7, 160))	480	conv2d_136[0][0]
batch_normalization_141 (BatchN (None, 7, 7, 160))	480	conv2d_141[0][0]
activation_136 (Activation) (None, 7, 7, 160)	0	batch_normalization_136[0][0]
activation_141 (Activation) (None, 7, 7, 160)	0	batch_normalization_141[0][0]
average_pooling2d_16 (AveragePo (None, 7, 7, 768))	0	mixed4[0][0]
conv2d_134 (Conv2D) (None, 7, 7, 192)	147456	mixed4[0][0]
conv2d_137 (Conv2D) (None, 7, 7, 192)	215040	activation_136[0][0]

[0]

conv2d_142 (Conv2D) [0]	(None, 7, 7, 192)	215040	activation_141[0]
conv2d_143 (Conv2D) 6[0][0]	(None, 7, 7, 192)	147456	average_pooling2d_1
batch_normalization_134 (BatchN (None, 7, 7, 192))	576	conv2d_134[0][0]	
batch_normalization_137 (BatchN (None, 7, 7, 192))	576	conv2d_137[0][0]	
batch_normalization_142 (BatchN (None, 7, 7, 192))	576	conv2d_142[0][0]	
batch_normalization_143 (BatchN (None, 7, 7, 192))	576	conv2d_143[0][0]	
activation_134 (Activation) _134[0][0]	(None, 7, 7, 192)	0	batch_normalization
activation_137 (Activation) _137[0][0]	(None, 7, 7, 192)	0	batch_normalization
activation_142 (Activation) _142[0][0]	(None, 7, 7, 192)	0	batch_normalization
activation_143 (Activation) _143[0][0]	(None, 7, 7, 192)	0	batch_normalization
mixed5 (Concatenate) [0]	(None, 7, 7, 768)	0	activation_134[0]
[0]			activation_137[0]
[0]			activation_142[0]
[0]			activation_143[0]
conv2d_148 (Conv2D)	(None, 7, 7, 160)	122880	mixed5[0][0]
batch_normalization_148 (BatchN (None, 7, 7, 160))	480	conv2d_148[0][0]	
activation_148 (Activation) _148[0][0]	(None, 7, 7, 160)	0	batch_normalization
conv2d_149 (Conv2D) [0]	(None, 7, 7, 160)	179200	activation_148[0]
batch_normalization_149 (BatchN (None, 7, 7, 160))	480	conv2d_149[0][0]	
activation_149 (Activation) _149[0][0]	(None, 7, 7, 160)	0	batch_normalization

conv2d_145 (Conv2D)	(None, 7, 7, 160)	122880	mixed5[0][0]
conv2d_150 (Conv2D)	(None, 7, 7, 160)	179200	activation_149[0][0]
batch_normalization_145 (BatchN)	(None, 7, 7, 160)	480	conv2d_145[0][0]
batch_normalization_150 (BatchN)	(None, 7, 7, 160)	480	conv2d_150[0][0]
activation_145 (Activation)	(None, 7, 7, 160)	0	batch_normalization_145[0][0]
activation_150 (Activation)	(None, 7, 7, 160)	0	batch_normalization_150[0][0]
conv2d_146 (Conv2D)	(None, 7, 7, 160)	179200	activation_145[0][0]
conv2d_151 (Conv2D)	(None, 7, 7, 160)	179200	activation_150[0][0]
batch_normalization_146 (BatchN)	(None, 7, 7, 160)	480	conv2d_146[0][0]
batch_normalization_151 (BatchN)	(None, 7, 7, 160)	480	conv2d_151[0][0]
activation_146 (Activation)	(None, 7, 7, 160)	0	batch_normalization_146[0][0]
activation_151 (Activation)	(None, 7, 7, 160)	0	batch_normalization_151[0][0]
average_pooling2d_17 (AveragePo	(None, 7, 7, 768)	0	mixed5[0][0]
conv2d_144 (Conv2D)	(None, 7, 7, 192)	147456	mixed5[0][0]
conv2d_147 (Conv2D)	(None, 7, 7, 192)	215040	activation_146[0][0]
conv2d_152 (Conv2D)	(None, 7, 7, 192)	215040	activation_151[0][0]
conv2d_153 (Conv2D)	(None, 7, 7, 192)	147456	average_pooling2d_17[0][0]
batch_normalization_144 (BatchN)	(None, 7, 7, 192)	576	conv2d_144[0][0]
batch_normalization_147 (BatchN)	(None, 7, 7, 192)	576	conv2d_147[0][0]
batch_normalization_152 (BatchN)	(None, 7, 7, 192)	576	conv2d_152[0][0]

batch_normalization_153 (BatchN (None, 7, 7, 192)	576	conv2d_153[0][0]
activation_144 (Activation) (None, 7, 7, 192)	0	batch_normalization_144[0][0]
activation_147 (Activation) (None, 7, 7, 192)	0	batch_normalization_147[0][0]
activation_152 (Activation) (None, 7, 7, 192)	0	batch_normalization_152[0][0]
activation_153 (Activation) (None, 7, 7, 192)	0	batch_normalization_153[0][0]
mixed6 (Concatenate) [0] [0] [0] [0]	0	activation_144[0] activation_147[0] activation_152[0] activation_153[0]
conv2d_158 (Conv2D) (None, 7, 7, 192)	147456	mixed6[0][0]
batch_normalization_158 (BatchN (None, 7, 7, 192)	576	conv2d_158[0][0]
activation_158 (Activation) (None, 7, 7, 192)	0	batch_normalization_158[0][0]
conv2d_159 (Conv2D) [0] (None, 7, 7, 192)	258048	activation_158[0]
batch_normalization_159 (BatchN (None, 7, 7, 192)	576	conv2d_159[0][0]
activation_159 (Activation) (None, 7, 7, 192)	0	batch_normalization_159[0][0]
conv2d_155 (Conv2D) (None, 7, 7, 192)	147456	mixed6[0][0]
conv2d_160 (Conv2D) [0] (None, 7, 7, 192)	258048	activation_159[0]
batch_normalization_155 (BatchN (None, 7, 7, 192)	576	conv2d_155[0][0]
batch_normalization_160 (BatchN (None, 7, 7, 192)	576	conv2d_160[0][0]
activation_155 (Activation) (None, 7, 7, 192)	0	batch_normalization_155[0][0]

activation_160 (Activation) _160[0][0]	(None, 7, 7, 192)	0	batch_normalization
conv2d_156 (Conv2D) [0]	(None, 7, 7, 192)	258048	activation_155[0]
conv2d_161 (Conv2D) [0]	(None, 7, 7, 192)	258048	activation_160[0]
batch_normalization_156 (BatchN (None, 7, 7, 192)	576	conv2d_156[0][0]	
batch_normalization_161 (BatchN (None, 7, 7, 192)	576	conv2d_161[0][0]	
activation_156 (Activation) _156[0][0]	(None, 7, 7, 192)	0	batch_normalization
activation_161 (Activation) _161[0][0]	(None, 7, 7, 192)	0	batch_normalization
average_pooling2d_18 (AveragePo (None, 7, 7, 768)	0	mixed6[0]	[0]
conv2d_154 (Conv2D)	(None, 7, 7, 192)	147456	mixed6[0][0]
conv2d_157 (Conv2D) [0]	(None, 7, 7, 192)	258048	activation_156[0]
conv2d_162 (Conv2D) [0]	(None, 7, 7, 192)	258048	activation_161[0]
conv2d_163 (Conv2D) 8[0][0]	(None, 7, 7, 192)	147456	average_pooling2d_1
batch_normalization_154 (BatchN (None, 7, 7, 192)	576	conv2d_154[0]	[0]
batch_normalization_157 (BatchN (None, 7, 7, 192)	576	conv2d_157[0]	[0]
batch_normalization_162 (BatchN (None, 7, 7, 192)	576	conv2d_162[0]	[0]
batch_normalization_163 (BatchN (None, 7, 7, 192)	576	conv2d_163[0]	[0]
activation_154 (Activation) _154[0][0]	(None, 7, 7, 192)	0	batch_normalization
activation_157 (Activation) _157[0][0]	(None, 7, 7, 192)	0	batch_normalization
activation_162 (Activation) _162[0][0]	(None, 7, 7, 192)	0	batch_normalization
activation_163 (Activation)	(None, 7, 7, 192)	0	batch_normalization

_163[0][0]

<u>mixed7 (Concatenate)</u>	(None, 7, 7, 768)	0	activation_154[0]
[0]			activation_157[0]
[0]			activation_162[0]
[0]			activation_163[0]
<u>conv2d_166 (Conv2D)</u>	(None, 7, 7, 192)	147456	mixed7[0][0]
<u>batch_normalization_166 (BatchN)</u>	(None, 7, 7, 192)	576	conv2d_166[0][0]
<u>activation_166 (Activation)</u>	(None, 7, 7, 192)	0	batch_normalization_166[0][0]
<u>conv2d_167 (Conv2D)</u>	(None, 7, 7, 192)	258048	activation_166[0][0]
<u>batch_normalization_167 (BatchN)</u>	(None, 7, 7, 192)	576	conv2d_167[0][0]
<u>activation_167 (Activation)</u>	(None, 7, 7, 192)	0	batch_normalization_167[0][0]
<u>conv2d_164 (Conv2D)</u>	(None, 7, 7, 192)	147456	mixed7[0][0]
<u>conv2d_168 (Conv2D)</u>	(None, 7, 7, 192)	258048	activation_167[0][0]
<u>batch_normalization_164 (BatchN)</u>	(None, 7, 7, 192)	576	conv2d_164[0][0]
<u>batch_normalization_168 (BatchN)</u>	(None, 7, 7, 192)	576	conv2d_168[0][0]
<u>activation_164 (Activation)</u>	(None, 7, 7, 192)	0	batch_normalization_164[0][0]
<u>activation_168 (Activation)</u>	(None, 7, 7, 192)	0	batch_normalization_168[0][0]
<u>conv2d_165 (Conv2D)</u>	(None, 3, 3, 320)	552960	activation_164[0][0]
<u>conv2d_169 (Conv2D)</u>	(None, 3, 3, 192)	331776	activation_168[0][0]
<u>batch_normalization_165 (BatchN)</u>	(None, 3, 3, 320)	960	conv2d_165[0][0]
<u>batch_normalization_169 (BatchN)</u>	(None, 3, 3, 192)	576	conv2d_169[0][0]

activation_165 (Activation) _165[0][0]	(None, 3, 3, 320)	0	batch_normalization
activation_169 (Activation) _169[0][0]	(None, 3, 3, 192)	0	batch_normalization
max_pooling2d_7 (MaxPooling2D)	(None, 3, 3, 768)	0	mixed7[0][0]
mixed8 (Concatenate) [0]	(None, 3, 3, 1280)	0	activation_165[0] activation_169[0]
[0]			max_pooling2d_7[0] [0]
conv2d_174 (Conv2D)	(None, 3, 3, 448)	573440	mixed8[0][0]
batch_normalization_174 (BatchN)	(None, 3, 3, 448)	1344	conv2d_174[0][0]
activation_174 (Activation) _174[0][0]	(None, 3, 3, 448)	0	batch_normalization
conv2d_171 (Conv2D)	(None, 3, 3, 384)	491520	mixed8[0][0]
conv2d_175 (Conv2D) [0]	(None, 3, 3, 384)	1548288	activation_174[0] [0]
batch_normalization_171 (BatchN)	(None, 3, 3, 384)	1152	conv2d_171[0][0]
batch_normalization_175 (BatchN)	(None, 3, 3, 384)	1152	conv2d_175[0][0]
activation_171 (Activation) _171[0][0]	(None, 3, 3, 384)	0	batch_normalization
activation_175 (Activation) _175[0][0]	(None, 3, 3, 384)	0	batch_normalization
conv2d_172 (Conv2D) [0]	(None, 3, 3, 384)	442368	activation_171[0] [0]
conv2d_173 (Conv2D) [0]	(None, 3, 3, 384)	442368	activation_171[0]
conv2d_176 (Conv2D) [0]	(None, 3, 3, 384)	442368	activation_175[0] [0]
conv2d_177 (Conv2D) [0]	(None, 3, 3, 384)	442368	activation_175[0]
average_pooling2d_19 (AveragePo	(None, 3, 3, 1280)	0	mixed8[0][0]

conv2d_170 (Conv2D)	(None, 3, 3, 320)	409600	mixed8[0][0]
batch_normalization_172 (BatchN)	(None, 3, 3, 384)	1152	conv2d_172[0][0]
batch_normalization_173 (BatchN)	(None, 3, 3, 384)	1152	conv2d_173[0][0]
batch_normalization_176 (BatchN)	(None, 3, 3, 384)	1152	conv2d_176[0][0]
batch_normalization_177 (BatchN)	(None, 3, 3, 384)	1152	conv2d_177[0][0]
conv2d_178 (Conv2D)	(None, 3, 3, 192)	245760	average_pooling2d_1 9[0][0]
batch_normalization_170 (BatchN)	(None, 3, 3, 320)	960	conv2d_170[0][0]
activation_172 (Activation)	(None, 3, 3, 384)	0	batch_normalization_172[0][0]
activation_173 (Activation)	(None, 3, 3, 384)	0	batch_normalization_173[0][0]
activation_176 (Activation)	(None, 3, 3, 384)	0	batch_normalization_176[0][0]
activation_177 (Activation)	(None, 3, 3, 384)	0	batch_normalization_177[0][0]
batch_normalization_178 (BatchN)	(None, 3, 3, 192)	576	conv2d_178[0][0]
activation_170 (Activation)	(None, 3, 3, 320)	0	batch_normalization_170[0][0]
mixed9_0 (Concatenate)	(None, 3, 3, 768)	0	activation_172[0] activation_173[0] [0]
concatenate_2 (Concatenate)	(None, 3, 3, 768)	0	activation_176[0] activation_177[0] [0]
activation_178 (Activation)	(None, 3, 3, 192)	0	batch_normalization_178[0][0]
mixed9 (Concatenate)	(None, 3, 3, 2048)	0	activation_170[0] mixed9_0[0][0] concatenate_2[0][0] activation_178[0] [0]

InceptionV3				
conv2d_183 (Conv2D)	(None, 3, 3, 448)	917504	mixed9[0][0]	
batch_normalization_183 (BatchN (None, 3, 3, 448))		1344	conv2d_183[0][0]	
activation_183 (Activation)	(None, 3, 3, 448)	0	batch_normalization_183[0][0]	
conv2d_180 (Conv2D)	(None, 3, 3, 384)	786432	mixed9[0][0]	
conv2d_184 (Conv2D)	(None, 3, 3, 384)	1548288	activation_183[0][0]	
batch_normalization_180 (BatchN (None, 3, 3, 384))		1152	conv2d_180[0][0]	
batch_normalization_184 (BatchN (None, 3, 3, 384))		1152	conv2d_184[0][0]	
activation_180 (Activation)	(None, 3, 3, 384)	0	batch_normalization_180[0][0]	
activation_184 (Activation)	(None, 3, 3, 384)	0	batch_normalization_184[0][0]	
conv2d_181 (Conv2D)	(None, 3, 3, 384)	442368	activation_180[0][0]	
conv2d_182 (Conv2D)	(None, 3, 3, 384)	442368	activation_180[0][0]	
conv2d_185 (Conv2D)	(None, 3, 3, 384)	442368	activation_184[0][0]	
conv2d_186 (Conv2D)	(None, 3, 3, 384)	442368	activation_184[0][0]	
average_pooling2d_20 (AveragePo (None, 3, 3, 2048))		0	mixed9[0][0]	
conv2d_179 (Conv2D)	(None, 3, 3, 320)	655360	mixed9[0][0]	
batch_normalization_181 (BatchN (None, 3, 3, 384))		1152	conv2d_181[0][0]	
batch_normalization_182 (BatchN (None, 3, 3, 384))		1152	conv2d_182[0][0]	
batch_normalization_185 (BatchN (None, 3, 3, 384))		1152	conv2d_185[0][0]	
batch_normalization_186 (BatchN (None, 3, 3, 384))		1152	conv2d_186[0][0]	
conv2d_187 (Conv2D)	(None, 3, 3, 192)	393216	average_pooling2d_20[0][0]	

batch_normalization_179 (BatchN (None, 3, 3, 320)	960	conv2d_179[0][0]
activation_181 (Activation) (None, 3, 3, 384)	0	batch_normalization_181[0][0]
activation_182 (Activation) (None, 3, 3, 384)	0	batch_normalization_182[0][0]
activation_185 (Activation) (None, 3, 3, 384)	0	batch_normalization_185[0][0]
activation_186 (Activation) (None, 3, 3, 384)	0	batch_normalization_186[0][0]
batch_normalization_187 (BatchN (None, 3, 3, 192)	576	conv2d_187[0][0]
activation_179 (Activation) (None, 3, 3, 320)	0	batch_normalization_179[0][0]
mixed9_1 (Concatenate) [0]	0	activation_181[0]
		activation_182[0]
concatenate_3 (Concatenate) [0]	0	activation_185[0]
		activation_186[0]
activation_187 (Activation) (None, 3, 3, 192)	0	batch_normalization_187[0][0]
mixed10 (Concatenate) [0]	0	activation_179[0]
		mixed9_1[0][0]
		concatenate_3[0][0]
		activation_187[0]
<hr/>		
<hr/>		
Total params: 21,802,784		
Trainable params: 0		
Non-trainable params: 21,802,784		
<hr/>		
('Failed to import pydot. You must `pip install pydot` and install graphviz (http://graphviz.gitlab.io/download/), ', 'for `pydotprint` to work.')		

Obtain last layer output of the pre-trained model

```
In [47]: last_layer = pre_trained_model.get_layer('mixed7') # identify Last Layer through model
last_output = last_layer.output
```

Adding dense layers after pre-trained model

```
In [48]: x = layers.Flatten()(last_output)
x = layers.Dense(512, activation='relu', kernel_initializer='he_uniform')(x)
x = layers.Dense(1024, activation='relu', kernel_initializer='he_uniform')(x)
x = layers.Dense(1024, activation='relu', kernel_initializer='he_uniform')(x)
x = layers.Dropout(0.2)(x)
x = layers.Dense(2, activation='softmax')(x)

model = Model(pre_trained_model.input, x)

model.compile(optimizer=Adam(lr=0.0001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Model summary of Inception v3 with dense layers

```
In [50]: model.summary()
plot_model(model, to_file='inception_v3_with_dense_layers_model.png', show_shapes=False)
```

Model: "model_2"

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	[None, 150, 150, 3]	0	
conv2d_94 (Conv2D)	(None, 74, 74, 32)	864	input_2[0][0]
batch_normalization_94 (BatchNo)	(None, 74, 74, 32)	96	conv2d_94[0][0]
activation_94 (Activation)	(None, 74, 74, 32)	0	batch_normalization_94[0][0]
conv2d_95 (Conv2D)	(None, 72, 72, 32)	9216	activation_94[0][0]
batch_normalization_95 (BatchNo)	(None, 72, 72, 32)	96	conv2d_95[0][0]
activation_95 (Activation)	(None, 72, 72, 32)	0	batch_normalization_95[0][0]
conv2d_96 (Conv2D)	(None, 72, 72, 64)	18432	activation_95[0][0]
batch_normalization_96 (BatchNo)	(None, 72, 72, 64)	192	conv2d_96[0][0]
activation_96 (Activation)	(None, 72, 72, 64)	0	batch_normalization_96[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 35, 35, 64)	0	activation_96[0][0]
conv2d_97 (Conv2D)	(None, 35, 35, 80)	5120	max_pooling2d_4[0][0]
batch_normalization_97 (BatchNo)	(None, 35, 35, 80)	240	conv2d_97[0][0]

activation_97 (Activation) _97[0][0]	(None, 35, 35, 80)	0	batch_normalization
conv2d_98 (Conv2D)	(None, 33, 33, 192)	138240	activation_97[0][0]
batch_normalization_98 (BatchNo)	(None, 33, 33, 192)	576	conv2d_98[0][0]
activation_98 (Activation) _98[0][0]	(None, 33, 33, 192)	0	batch_normalization
max_pooling2d_5 (MaxPooling2D)	(None, 16, 16, 192)	0	activation_98[0][0]
conv2d_102 (Conv2D) [0]	(None, 16, 16, 64)	12288	max_pooling2d_5[0]
batch_normalization_102 (BatchN)	(None, 16, 16, 64)	192	conv2d_102[0][0]
activation_102 (Activation) _102[0][0]	(None, 16, 16, 64)	0	batch_normalization
conv2d_100 (Conv2D) [0]	(None, 16, 16, 48)	9216	max_pooling2d_5[0]
conv2d_103 (Conv2D) [0]	(None, 16, 16, 96)	55296	activation_102[0]
batch_normalization_100 (BatchN)	(None, 16, 16, 48)	144	conv2d_100[0][0]
batch_normalization_103 (BatchN)	(None, 16, 16, 96)	288	conv2d_103[0]
activation_100 (Activation) _100[0][0]	(None, 16, 16, 48)	0	batch_normalization
activation_103 (Activation) _103[0][0]	(None, 16, 16, 96)	0	batch_normalization
average_pooling2d_12 (AveragePo [0]	(None, 16, 16, 192)	0	max_pooling2d_5[0]
conv2d_99 (Conv2D) [0]	(None, 16, 16, 64)	12288	max_pooling2d_5[0]
conv2d_101 (Conv2D) [0]	(None, 16, 16, 64)	76800	activation_100[0]
conv2d_104 (Conv2D) [0]	(None, 16, 16, 96)	82944	activation_103[0]
conv2d_105 (Conv2D) 2[0][0]	(None, 16, 16, 32)	6144	average_pooling2d_1

batch_normalization_99 (BatchNorm (None, 16, 16, 64))	192	conv2d_99[0][0]
batch_normalization_101 (BatchNorm (None, 16, 16, 64))	192	conv2d_101[0][0]
batch_normalization_104 (BatchNorm (None, 16, 16, 96))	288	conv2d_104[0][0]
batch_normalization_105 (BatchNorm (None, 16, 16, 32))	96	conv2d_105[0][0]
activation_99 (Activation) (None, 16, 16, 64)	0	batch_normalization_99[0][0]
activation_101 (Activation) (None, 16, 16, 64)	0	batch_normalization_101[0][0]
activation_104 (Activation) (None, 16, 16, 96)	0	batch_normalization_104[0][0]
activation_105 (Activation) (None, 16, 16, 32)	0	batch_normalization_105[0][0]
mixed0 (Concatenate) (None, 16, 16, 256)	0	activation_99[0][0] activation_101[0][0] activation_104[0][0] activation_105[0][0]
conv2d_109 (Conv2D) (None, 16, 16, 64)	16384	mixed0[0][0]
batch_normalization_109 (BatchNorm (None, 16, 16, 64))	192	conv2d_109[0][0]
activation_109 (Activation) (None, 16, 16, 64)	0	batch_normalization_109[0][0]
conv2d_107 (Conv2D) (None, 16, 16, 48)	12288	mixed0[0][0]
conv2d_110 (Conv2D) (None, 16, 16, 96)	55296	activation_109[0][0]
batch_normalization_107 (BatchNorm (None, 16, 16, 48))	144	conv2d_107[0][0]
batch_normalization_110 (BatchNorm (None, 16, 16, 96))	288	conv2d_110[0][0]
activation_107 (Activation) (None, 16, 16, 48)	0	batch_normalization_107[0][0]
activation_110 (Activation) (None, 16, 16, 96)	0	batch_normalization_110[0][0]

average_pooling2d_13 (AveragePo [None, 16, 16, 256])	0		mixed0[0][0]
conv2d_106 (Conv2D)	(None, 16, 16, 64)	16384	mixed0[0][0]
conv2d_108 (Conv2D) [0]	(None, 16, 16, 64)	76800	activation_107[0]
conv2d_111 (Conv2D) [0]	(None, 16, 16, 96)	82944	activation_110[0]
conv2d_112 (Conv2D) 3[0][0]	(None, 16, 16, 64)	16384	average_pooling2d_1
batch_normalization_106 (BatchN [None, 16, 16, 64])	192		conv2d_106[0][0]
batch_normalization_108 (BatchN [None, 16, 16, 64])	192		conv2d_108[0][0]
batch_normalization_111 (BatchN [None, 16, 16, 96])	288		conv2d_111[0][0]
batch_normalization_112 (BatchN [None, 16, 16, 64])	192		conv2d_112[0][0]
activation_106 (Activation) _106[0][0]	(None, 16, 16, 64)	0	batch_normalization
activation_108 (Activation) _108[0][0]	(None, 16, 16, 64)	0	batch_normalization
activation_111 (Activation) _111[0][0]	(None, 16, 16, 96)	0	batch_normalization
activation_112 (Activation) _112[0][0]	(None, 16, 16, 64)	0	batch_normalization
mixed1 (Concatenate) [0]	(None, 16, 16, 288)	0	activation_106[0]
[0]			activation_108[0]
[0]			activation_111[0]
[0]			activation_112[0]
conv2d_116 (Conv2D)	(None, 16, 16, 64)	18432	mixed1[0][0]
batch_normalization_116 (BatchN [None, 16, 16, 64])	192		conv2d_116[0][0]
activation_116 (Activation) _116[0][0]	(None, 16, 16, 64)	0	batch_normalization
conv2d_114 (Conv2D)	(None, 16, 16, 48)	13824	mixed1[0][0]

conv2d_117 (Conv2D) [0]	(None, 16, 16, 96)	55296	activation_116[0]
batch_normalization_114 (BatchN) [None, 16, 16, 48]	144	conv2d_114[0][0]	
batch_normalization_117 (BatchN) [None, 16, 16, 96]	288	conv2d_117[0][0]	
activation_114 (Activation) [None][0]	(None, 16, 16, 48)	0	batch_normalization_114[0][0]
activation_117 (Activation) [None][0]	(None, 16, 16, 96)	0	batch_normalization_117[0][0]
average_pooling2d_14 (AveragePo [None, 16, 16, 288])	0	mixed1[0]	[0]
conv2d_113 (Conv2D) [0]	(None, 16, 16, 64)	18432	mixed1[0][0]
conv2d_115 (Conv2D) [0]	(None, 16, 16, 64)	76800	activation_114[0]
conv2d_118 (Conv2D) [0]	(None, 16, 16, 96)	82944	activation_117[0]
conv2d_119 (Conv2D) [0][0]	(None, 16, 16, 64)	18432	average_pooling2d_1
batch_normalization_113 (BatchN) [None, 16, 16, 64]	192	conv2d_113[0]	[0]
batch_normalization_115 (BatchN) [None, 16, 16, 64]	192	conv2d_115[0]	[0]
batch_normalization_118 (BatchN) [None, 16, 16, 96]	288	conv2d_118[0]	[0]
batch_normalization_119 (BatchN) [None, 16, 16, 64]	192	conv2d_119[0]	[0]
activation_113 (Activation) [None][0]	(None, 16, 16, 64)	0	batch_normalization_113[0][0]
activation_115 (Activation) [None][0]	(None, 16, 16, 64)	0	batch_normalization_115[0][0]
activation_118 (Activation) [None][0]	(None, 16, 16, 96)	0	batch_normalization_118[0][0]
activation_119 (Activation) [None][0]	(None, 16, 16, 64)	0	batch_normalization_119[0][0]
mixed2 (Concatenate) [0]	(None, 16, 16, 288)	0	activation_113[0]
			activation_115[0]
			activation_118[0]

[0]				activation_119[0]
conv2d_121 (Conv2D)	(None, 16, 16, 64)	18432	mixed2[0][0]	
batch_normalization_121 (BatchN)	(None, 16, 16, 64)	192	conv2d_121[0][0]	
activation_121 (Activation)	(None, 16, 16, 64)	0	batch_normalization_121[0][0]	
conv2d_122 (Conv2D)	(None, 16, 16, 96)	55296	activation_121[0]	
batch_normalization_122 (BatchN)	(None, 16, 16, 96)	288	conv2d_122[0][0]	
activation_122 (Activation)	(None, 16, 16, 96)	0	batch_normalization_122[0][0]	
conv2d_120 (Conv2D)	(None, 7, 7, 384)	995328	mixed2[0][0]	
conv2d_123 (Conv2D)	(None, 7, 7, 96)	82944	activation_122[0]	
batch_normalization_120 (BatchN)	(None, 7, 7, 384)	1152	conv2d_120[0][0]	
batch_normalization_123 (BatchN)	(None, 7, 7, 96)	288	conv2d_123[0][0]	
activation_120 (Activation)	(None, 7, 7, 384)	0	batch_normalization_120[0][0]	
activation_123 (Activation)	(None, 7, 7, 96)	0	batch_normalization_123[0][0]	
max_pooling2d_6 (MaxPooling2D)	(None, 7, 7, 288)	0	mixed2[0][0]	
mixed3 (Concatenate)	(None, 7, 7, 768)	0	activation_120[0]	
[0]			activation_123[0]	
[0]			max_pooling2d_6[0]	
conv2d_128 (Conv2D)	(None, 7, 7, 128)	98304	mixed3[0][0]	
batch_normalization_128 (BatchN)	(None, 7, 7, 128)	384	conv2d_128[0][0]	
activation_128 (Activation)	(None, 7, 7, 128)	0	batch_normalization_128[0][0]	
conv2d_129 (Conv2D)	(None, 7, 7, 128)	114688	activation_128[0]	

[0]

batch_normalization_129 (BatchN (None, 7, 7, 128))	(None, 7, 7, 128)	384	conv2d_129[0][0]
activation_129 (Activation)	(None, 7, 7, 128)	0	batch_normalization_129[0][0]
conv2d_125 (Conv2D)	(None, 7, 7, 128)	98304	mixed3[0][0]
conv2d_130 (Conv2D)	(None, 7, 7, 128)	114688	activation_129[0][0]
batch_normalization_125 (BatchN (None, 7, 7, 128))	(None, 7, 7, 128)	384	conv2d_125[0][0]
batch_normalization_130 (BatchN (None, 7, 7, 128))	(None, 7, 7, 128)	384	conv2d_130[0][0]
activation_125 (Activation)	(None, 7, 7, 128)	0	batch_normalization_125[0][0]
activation_130 (Activation)	(None, 7, 7, 128)	0	batch_normalization_130[0][0]
conv2d_126 (Conv2D)	(None, 7, 7, 128)	114688	activation_125[0][0]
conv2d_131 (Conv2D)	(None, 7, 7, 128)	114688	activation_130[0][0]
batch_normalization_126 (BatchN (None, 7, 7, 128))	(None, 7, 7, 128)	384	conv2d_126[0][0]
batch_normalization_131 (BatchN (None, 7, 7, 128))	(None, 7, 7, 128)	384	conv2d_131[0][0]
activation_126 (Activation)	(None, 7, 7, 128)	0	batch_normalization_126[0][0]
activation_131 (Activation)	(None, 7, 7, 128)	0	batch_normalization_131[0][0]
average_pooling2d_15 (AveragePo	(None, 7, 7, 768)	0	mixed3[0][0]
conv2d_124 (Conv2D)	(None, 7, 7, 192)	147456	mixed3[0][0]
conv2d_127 (Conv2D)	(None, 7, 7, 192)	172032	activation_126[0][0]
conv2d_132 (Conv2D)	(None, 7, 7, 192)	172032	activation_131[0][0]
conv2d_133 (Conv2D)	(None, 7, 7, 192)	147456	average_pooling2d_15[0][0]

batch_normalization_124 (BatchN (None, 7, 7, 192)	576	conv2d_124[0][0]
batch_normalization_127 (BatchN (None, 7, 7, 192)	576	conv2d_127[0][0]
batch_normalization_132 (BatchN (None, 7, 7, 192)	576	conv2d_132[0][0]
batch_normalization_133 (BatchN (None, 7, 7, 192)	576	conv2d_133[0][0]
activation_124 (Activation) (None, 7, 7, 192)	0	batch_normalization_124[0][0]
activation_127 (Activation) (None, 7, 7, 192)	0	batch_normalization_127[0][0]
activation_132 (Activation) (None, 7, 7, 192)	0	batch_normalization_132[0][0]
activation_133 (Activation) (None, 7, 7, 192)	0	batch_normalization_133[0][0]
mixed4 (Concatenate) [0] [0] [0] [0]	0	activation_124[0] activation_127[0] activation_132[0] activation_133[0]
conv2d_138 (Conv2D)	122880	mixed4[0][0]
batch_normalization_138 (BatchN (None, 7, 7, 160)	480	conv2d_138[0][0]
activation_138 (Activation) (None, 7, 7, 160)	0	batch_normalization_138[0][0]
conv2d_139 (Conv2D) [0]	179200	activation_138[0]
batch_normalization_139 (BatchN (None, 7, 7, 160)	480	conv2d_139[0][0]
activation_139 (Activation) (None, 7, 7, 160)	0	batch_normalization_139[0][0]
conv2d_135 (Conv2D)	122880	mixed4[0][0]
conv2d_140 (Conv2D) [0]	179200	activation_139[0]
batch_normalization_135 (BatchN (None, 7, 7, 160)	480	conv2d_135[0][0]

batch_normalization_140 (BatchN (None, 7, 7, 160))	480	conv2d_140[0][0]
activation_135 (Activation) (None, 7, 7, 160)	0	batch_normalization_135[0][0]
activation_140 (Activation) (None, 7, 7, 160)	0	batch_normalization_140[0][0]
conv2d_136 (Conv2D) (None, 7, 7, 160)	179200	activation_135[0]
conv2d_141 (Conv2D) (None, 7, 7, 160)	179200	activation_140[0]
batch_normalization_136 (BatchN (None, 7, 7, 160))	480	conv2d_136[0][0]
batch_normalization_141 (BatchN (None, 7, 7, 160))	480	conv2d_141[0][0]
activation_136 (Activation) (None, 7, 7, 160)	0	batch_normalization_136[0][0]
activation_141 (Activation) (None, 7, 7, 160)	0	batch_normalization_141[0][0]
average_pooling2d_16 (AveragePo (None, 7, 7, 768))	0	mixed4[0][0]
conv2d_134 (Conv2D) (None, 7, 7, 192)	147456	mixed4[0][0]
conv2d_137 (Conv2D) (None, 7, 7, 192)	215040	activation_136[0]
conv2d_142 (Conv2D) (None, 7, 7, 192)	215040	activation_141[0]
conv2d_143 (Conv2D) (None, 7, 7, 192)	147456	average_pooling2d_16[0][0]
batch_normalization_134 (BatchN (None, 7, 7, 192))	576	conv2d_134[0][0]
batch_normalization_137 (BatchN (None, 7, 7, 192))	576	conv2d_137[0][0]
batch_normalization_142 (BatchN (None, 7, 7, 192))	576	conv2d_142[0][0]
batch_normalization_143 (BatchN (None, 7, 7, 192))	576	conv2d_143[0][0]
activation_134 (Activation) (None, 7, 7, 192)	0	batch_normalization_134[0][0]

activation_137 (Activation) _137[0][0]	(None, 7, 7, 192)	0	batch_normalization
activation_142 (Activation) _142[0][0]	(None, 7, 7, 192)	0	batch_normalization
activation_143 (Activation) _143[0][0]	(None, 7, 7, 192)	0	batch_normalization
mixed5 (Concatenate) [0]	(None, 7, 7, 768)	0	activation_134[0] activation_137[0] activation_142[0] activation_143[0]
conv2d_148 (Conv2D)	(None, 7, 7, 160)	122880	mixed5[0][0]
batch_normalization_148 (BatchN (None, 7, 7, 160)	480	conv2d_148[0][0]	
activation_148 (Activation) _148[0][0]	(None, 7, 7, 160)	0	batch_normalization
conv2d_149 (Conv2D) [0]	(None, 7, 7, 160)	179200	activation_148[0]
batch_normalization_149 (BatchN (None, 7, 7, 160)	480	conv2d_149[0][0]	
activation_149 (Activation) _149[0][0]	(None, 7, 7, 160)	0	batch_normalization
conv2d_145 (Conv2D)	(None, 7, 7, 160)	122880	mixed5[0][0]
conv2d_150 (Conv2D) [0]	(None, 7, 7, 160)	179200	activation_149[0]
batch_normalization_145 (BatchN (None, 7, 7, 160)	480	conv2d_145[0][0]	
batch_normalization_150 (BatchN (None, 7, 7, 160)	480	conv2d_150[0][0]	
activation_145 (Activation) _145[0][0]	(None, 7, 7, 160)	0	batch_normalization
activation_150 (Activation) _150[0][0]	(None, 7, 7, 160)	0	batch_normalization
conv2d_146 (Conv2D) [0]	(None, 7, 7, 160)	179200	activation_145[0]
conv2d_151 (Conv2D)	(None, 7, 7, 160)	179200	activation_150[0]

[0]

batch_normalization_146 (BatchN (None, 7, 7, 160)	480	conv2d_146[0][0]
batch_normalization_151 (BatchN (None, 7, 7, 160)	480	conv2d_151[0][0]
activation_146 (Activation) (None, 7, 7, 160)	0	batch_normalization_146[0][0]
activation_151 (Activation) (None, 7, 7, 160)	0	batch_normalization_151[0][0]
average_pooling2d_17 (AveragePo (None, 7, 7, 768)	0	mixed5[0][0]
conv2d_144 (Conv2D) (None, 7, 7, 192)	147456	mixed5[0][0]
conv2d_147 (Conv2D) (None, 7, 7, 192)	215040	activation_146[0][0]
conv2d_152 (Conv2D) (None, 7, 7, 192)	215040	activation_151[0][0]
conv2d_153 (Conv2D) (None, 7, 7, 192)	147456	average_pooling2d_17[0][0]
batch_normalization_144 (BatchN (None, 7, 7, 192)	576	conv2d_144[0][0]
batch_normalization_147 (BatchN (None, 7, 7, 192)	576	conv2d_147[0][0]
batch_normalization_152 (BatchN (None, 7, 7, 192)	576	conv2d_152[0][0]
batch_normalization_153 (BatchN (None, 7, 7, 192)	576	conv2d_153[0][0]
activation_144 (Activation) (None, 7, 7, 192)	0	batch_normalization_144[0][0]
activation_147 (Activation) (None, 7, 7, 192)	0	batch_normalization_147[0][0]
activation_152 (Activation) (None, 7, 7, 192)	0	batch_normalization_152[0][0]
activation_153 (Activation) (None, 7, 7, 192)	0	batch_normalization_153[0][0]
mixed6 (Concatenate) [0]	(None, 7, 7, 768)	0
		activation_144[0]
		activation_147[0]
		activation_152[0]
		activation_153[0]

activation_153[0]

[0]

conv2d_158 (Conv2D)	(None, 7, 7, 192)	147456	mixed6[0][0]
batch_normalization_158 (BatchN	(None, 7, 7, 192)	576	conv2d_158[0][0]
activation_158 (Activation)	(None, 7, 7, 192)	0	batch_normalization_158[0][0]
conv2d_159 (Conv2D)	(None, 7, 7, 192)	258048	activation_158[0]
[0]			
batch_normalization_159 (BatchN	(None, 7, 7, 192)	576	conv2d_159[0][0]
activation_159 (Activation)	(None, 7, 7, 192)	0	batch_normalization_159[0][0]
conv2d_155 (Conv2D)	(None, 7, 7, 192)	147456	mixed6[0][0]
conv2d_160 (Conv2D)	(None, 7, 7, 192)	258048	activation_159[0]
[0]			
batch_normalization_155 (BatchN	(None, 7, 7, 192)	576	conv2d_155[0][0]
batch_normalization_160 (BatchN	(None, 7, 7, 192)	576	conv2d_160[0][0]
activation_155 (Activation)	(None, 7, 7, 192)	0	batch_normalization_155[0][0]
activation_160 (Activation)	(None, 7, 7, 192)	0	batch_normalization_160[0][0]
conv2d_156 (Conv2D)	(None, 7, 7, 192)	258048	activation_155[0]
[0]			
conv2d_161 (Conv2D)	(None, 7, 7, 192)	258048	activation_160[0]
[0]			
batch_normalization_156 (BatchN	(None, 7, 7, 192)	576	conv2d_156[0][0]
batch_normalization_161 (BatchN	(None, 7, 7, 192)	576	conv2d_161[0][0]
activation_156 (Activation)	(None, 7, 7, 192)	0	batch_normalization_156[0][0]
activation_161 (Activation)	(None, 7, 7, 192)	0	batch_normalization_161[0][0]
average_pooling2d_18 (AveragePo	(None, 7, 7, 768)	0	mixed6[0][0]

conv2d_154 (Conv2D)	(None, 7, 7, 192)	147456	mixed6[0][0]
conv2d_157 (Conv2D)	(None, 7, 7, 192)	258048	activation_156[0][0]
conv2d_162 (Conv2D)	(None, 7, 7, 192)	258048	activation_161[0][0]
conv2d_163 (Conv2D)	(None, 7, 7, 192)	147456	average_pooling2d_18[0][0]
batch_normalization_154 (BatchN)	(None, 7, 7, 192)	576	conv2d_154[0][0]
batch_normalization_157 (BatchN)	(None, 7, 7, 192)	576	conv2d_157[0][0]
batch_normalization_162 (BatchN)	(None, 7, 7, 192)	576	conv2d_162[0][0]
batch_normalization_163 (BatchN)	(None, 7, 7, 192)	576	conv2d_163[0][0]
activation_154 (Activation)	(None, 7, 7, 192)	0	batch_normalization_154[0][0]
activation_157 (Activation)	(None, 7, 7, 192)	0	batch_normalization_157[0][0]
activation_162 (Activation)	(None, 7, 7, 192)	0	batch_normalization_162[0][0]
activation_163 (Activation)	(None, 7, 7, 192)	0	batch_normalization_163[0][0]
mixed7 (Concatenate)	(None, 7, 7, 768)	0	activation_154[0][0]
			activation_157[0][0]
			activation_162[0][0]
			activation_163[0][0]
flatten_7 (Flatten)	(None, 37632)	0	mixed7[0][0]
dense_10 (Dense)	(None, 512)	19268096	flatten_7[0][0]
dense_11 (Dense)	(None, 1024)	525312	dense_10[0][0]
dense_12 (Dense)	(None, 1024)	1049600	dense_11[0][0]
dropout_2 (Dropout)	(None, 1024)	0	dense_12[0][0]

```

dense_13 (Dense)           (None, 2)      2050      dropout_2[0][0]
=====
=====
Total params: 29,820,322
Trainable params: 20,845,058
Non-trainable params: 8,975,264
=====

('Failed to import pydot. You must `pip install pydot` and install graphviz (http://graphviz.gitlab.io/download/), ', 'for `pydotprint` to work.')

```

Fitting model

```
In [51]: step_size_train = train_gender_generator.n//train_gender_generator.batch_size

history = model.fit_generator(generator=train_gender_generator,
                               validation_data=val_gender_generator,
                               steps_per_epoch=step_size_train,
                               epochs=100,
                               validation_steps=None,
                               verbose=2,
                               callbacks=[earlystop, plateau]) # this will take some
```

```
c:\Miniconda\envs\d2l\lib\site-packages\tensorflow\python\keras\engine\training.py:1
844: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future
e version. Please use `Model.fit`, which supports generators.
    warnings.warn(`Model.fit_generator` is deprecated and '
Epoch 1/100
678/678 - 145s - loss: 0.3606 - accuracy: 0.8489 - val_loss: 0.2661 - val_accuracy:
0.9001
Epoch 2/100
678/678 - 139s - loss: 0.2992 - accuracy: 0.8767 - val_loss: 0.2204 - val_accuracy:
0.9167
Epoch 3/100
678/678 - 139s - loss: 0.2838 - accuracy: 0.8861 - val_loss: 0.2620 - val_accuracy:
0.8956
Epoch 4/100
678/678 - 139s - loss: 0.2697 - accuracy: 0.8902 - val_loss: 0.2052 - val_accuracy:
0.9204
Epoch 5/100
678/678 - 138s - loss: 0.2661 - accuracy: 0.8927 - val_loss: 0.2050 - val_accuracy:
0.9196
Epoch 6/100
678/678 - 138s - loss: 0.2604 - accuracy: 0.8946 - val_loss: 0.2155 - val_accuracy:
0.9119
Epoch 7/100
678/678 - 139s - loss: 0.2536 - accuracy: 0.8965 - val_loss: 0.1968 - val_accuracy:
0.9222
Epoch 8/100
678/678 - 139s - loss: 0.2544 - accuracy: 0.8990 - val_loss: 0.1936 - val_accuracy:
0.9270
Epoch 9/100
678/678 - 139s - loss: 0.2482 - accuracy: 0.8990 - val_loss: 0.1900 - val_accuracy:
0.9259
Epoch 10/100
678/678 - 139s - loss: 0.2468 - accuracy: 0.9005 - val_loss: 0.1835 - val_accuracy:
0.9266
Epoch 11/100
678/678 - 139s - loss: 0.2361 - accuracy: 0.9040 - val_loss: 0.1866 - val_accuracy:
0.9263
Epoch 12/100
678/678 - 139s - loss: 0.2341 - accuracy: 0.9046 - val_loss: 0.1988 - val_accuracy:
0.9274
Epoch 13/100
678/678 - 139s - loss: 0.2294 - accuracy: 0.9080 - val_loss: 0.1827 - val_accuracy:
```

0.9281
Epoch 14/100
678/678 - 139s - loss: 0.2311 - accuracy: 0.9073 - val_loss: 0.1838 - val_accuracy: 0.9270
Epoch 15/100
678/678 - 139s - loss: 0.2265 - accuracy: 0.9075 - val_loss: 0.1878 - val_accuracy: 0.9259
Epoch 16/100
678/678 - 139s - loss: 0.2269 - accuracy: 0.9096 - val_loss: 0.1896 - val_accuracy: 0.9270
Epoch 17/100
678/678 - 139s - loss: 0.2246 - accuracy: 0.9094 - val_loss: 0.1956 - val_accuracy: 0.9281
Epoch 18/100
678/678 - 139s - loss: 0.2194 - accuracy: 0.9151 - val_loss: 0.1829 - val_accuracy: 0.9314
Epoch 19/100
678/678 - 139s - loss: 0.2165 - accuracy: 0.9139 - val_loss: 0.2342 - val_accuracy: 0.9071
Epoch 20/100
678/678 - 139s - loss: 0.2168 - accuracy: 0.9121 - val_loss: 0.2089 - val_accuracy: 0.9207
Epoch 21/100
678/678 - 139s - loss: 0.2127 - accuracy: 0.9150 - val_loss: 0.1836 - val_accuracy: 0.9307
Epoch 22/100
678/678 - 139s - loss: 0.2136 - accuracy: 0.9134 - val_loss: 0.1780 - val_accuracy: 0.9336
Epoch 23/100
678/678 - 139s - loss: 0.2131 - accuracy: 0.9124 - val_loss: 0.2067 - val_accuracy: 0.9237
Epoch 24/100
678/678 - 139s - loss: 0.2125 - accuracy: 0.9140 - val_loss: 0.1751 - val_accuracy: 0.9340
Epoch 25/100
678/678 - 139s - loss: 0.2095 - accuracy: 0.9173 - val_loss: 0.1750 - val_accuracy: 0.9369
Epoch 26/100
678/678 - 139s - loss: 0.2067 - accuracy: 0.9161 - val_loss: 0.1704 - val_accuracy: 0.9366
Epoch 27/100
678/678 - 139s - loss: 0.2052 - accuracy: 0.9180 - val_loss: 0.1708 - val_accuracy: 0.9355
Epoch 28/100
678/678 - 139s - loss: 0.1998 - accuracy: 0.9203 - val_loss: 0.1808 - val_accuracy: 0.9347
Epoch 29/100
678/678 - 139s - loss: 0.2018 - accuracy: 0.9188 - val_loss: 0.1709 - val_accuracy: 0.9336
Epoch 30/100
678/678 - 138s - loss: 0.2015 - accuracy: 0.9185 - val_loss: 0.1677 - val_accuracy: 0.9340

Epoch 00030: ReduceLROnPlateau reducing learning rate to 9.999999747378752e-06.
Epoch 31/100
678/678 - 139s - loss: 0.1885 - accuracy: 0.9267 - val_loss: 0.1655 - val_accuracy: 0.9369
Epoch 32/100
678/678 - 139s - loss: 0.1811 - accuracy: 0.9274 - val_loss: 0.1641 - val_accuracy: 0.9373
Epoch 33/100
678/678 - 138s - loss: 0.1880 - accuracy: 0.9269 - val_loss: 0.1651 - val_accuracy: 0.9384
Epoch 34/100
678/678 - 139s - loss: 0.1843 - accuracy: 0.9256 - val_loss: 0.1653 - val_accuracy: 0.9392
Epoch 35/100
678/678 - 139s - loss: 0.1807 - accuracy: 0.9286 - val_loss: 0.1662 - val_accuracy: 0.9384

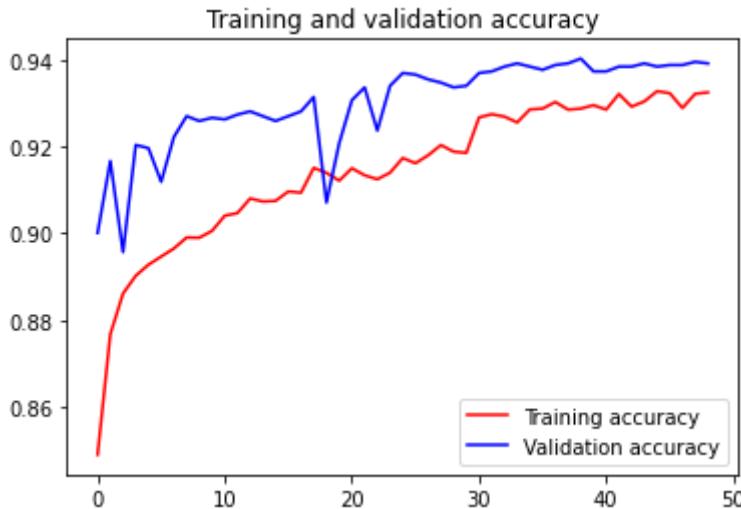
```
Epoch 36/100
678/678 - 139s - loss: 0.1801 - accuracy: 0.9287 - val_loss: 0.1665 - val_accuracy:
0.9377
Epoch 37/100
678/678 - 139s - loss: 0.1788 - accuracy: 0.9303 - val_loss: 0.1639 - val_accuracy:
0.9388
Epoch 38/100
678/678 - 139s - loss: 0.1807 - accuracy: 0.9285 - val_loss: 0.1667 - val_accuracy:
0.9392
Epoch 39/100
678/678 - 138s - loss: 0.1808 - accuracy: 0.9287 - val_loss: 0.1656 - val_accuracy:
0.9403
Epoch 40/100
678/678 - 139s - loss: 0.1757 - accuracy: 0.9295 - val_loss: 0.1675 - val_accuracy:
0.9373
Epoch 41/100
678/678 - 139s - loss: 0.1768 - accuracy: 0.9285 - val_loss: 0.1654 - val_accuracy:
0.9373
Epoch 42/100
678/678 - 139s - loss: 0.1728 - accuracy: 0.9321 - val_loss: 0.1693 - val_accuracy:
0.9384
Epoch 43/100
678/678 - 139s - loss: 0.1801 - accuracy: 0.9292 - val_loss: 0.1661 - val_accuracy:
0.9384
Epoch 44/100
678/678 - 139s - loss: 0.1762 - accuracy: 0.9304 - val_loss: 0.1662 - val_accuracy:
0.9392

Epoch 00044: ReduceLROnPlateau reducing learning rate to 9.999999747378752e-07.
Epoch 45/100
678/678 - 139s - loss: 0.1723 - accuracy: 0.9328 - val_loss: 0.1650 - val_accuracy:
0.9384
Epoch 46/100
678/678 - 139s - loss: 0.1764 - accuracy: 0.9323 - val_loss: 0.1647 - val_accuracy:
0.9388
Epoch 47/100
678/678 - 139s - loss: 0.1751 - accuracy: 0.9289 - val_loss: 0.1642 - val_accuracy:
0.9388
Epoch 48/100
678/678 - 139s - loss: 0.1734 - accuracy: 0.9322 - val_loss: 0.1647 - val_accuracy:
0.9395
Epoch 49/100
678/678 - 139s - loss: 0.1732 - accuracy: 0.9325 - val_loss: 0.1647 - val_accuracy:
0.9392
Restoring model weights from the end of the best epoch.

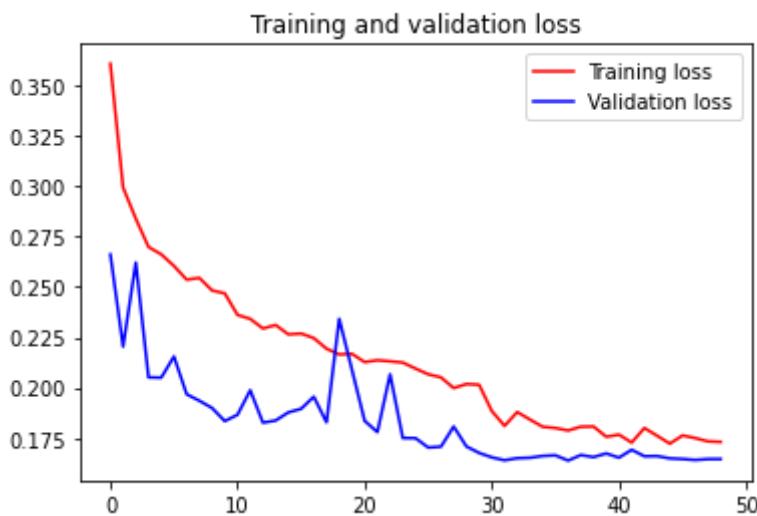
Epoch 00049: ReduceLROnPlateau reducing learning rate to 9.99999974752428e-08.
Epoch 00049: early stopping
```

Model result

```
In [52]: plot_acc(history)
```



<Figure size 432x288 with 0 Axes>

In [53]: `plot_loss(history)`

<Figure size 432x288 with 0 Axes>

```
# Evaluate the model on the test data using `evaluate`
print("Evaluate on test data")
results = model.evaluate(test_gender_generator, batch_size=32)
print("test loss, test acc:", results)

# Generate predictions (probabilities -- the output of the last layer)
# on new data using `predict`
print("Generate predictions for 3 samples")
predictions = model.predict(test_gender_generator)
print("predictions shape:", predictions.shape)
```

```
Evaluate on test data
85/85 [=====] - 3s 40ms/step - loss: 0.1612 - accuracy: 0.9370
test loss, test acc: [0.16123680770397186, 0.9369701147079468]
Generate predictions for 3 samples
predictions shape: (2713, 2)
```

```
In [60]: import numpy as np

labels =[ "Female", # index 0
          "Male"      # index 1
        ]

x, y = test_gender_generator.next()
```

```
# Plot a random sample of 10 test images, their predicted Labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks=[])
    # Display each image
    ax.imshow(np.squeeze(x[index]))
    predict_index = np.argmax(predictions[index])
    true_index = np.argmax(y[index])
    # Set the title for each image
    ax.set_title("predict: {}, true: {}".format(labels[predict_index],
                                                labels[true_index]),
                color=("green" if predict_index == true_index else "red"))
plt.show()
```



ResNet

April 29, 2021

1 age

1.1 load packages, link to GPU

```
[1]: import tensorflow as tf  
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

Num GPUs Available: 1

```
[2]: from tensorflow.python.client import device_lib  
print(device_lib.list_local_devices())
```

```
[name: "/device:CPU:0"  
device_type: "CPU"  
memory_limit: 268435456  
locality {  
}  
incarnation: 5224171634087788538  
, name: "/device:GPU:0"  
device_type: "GPU"  
memory_limit: 7843092896  
locality {  
    bus_id: 1  
    links {  
    }  
}  
incarnation: 14087842961676633213  
physical_device_desc: "device: 0, name: NVIDIA Tesla M60, pci bus id:  
0001:00:00.0, compute capability: 5.2"  
]
```

```
[1]: import pandas as pd  
import os  
from PIL import Image  
import numpy as np  
from keras.callbacks import ModelCheckpoint, LearningRateScheduler,  
    EarlyStopping, ReduceLROnPlateau, TensorBoard  
from keras import optimizers, losses, activations, models
```

```

from keras.layers import Convolution2D, Dense, Input, Flatten, Dropout, \
    MaxPooling2D, BatchNormalization, \
    GlobalMaxPool2D, Concatenate, GlobalMaxPooling2D, GlobalAveragePooling2D, \
    Lambda
from keras.applications.resnet50 import ResNet50
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from keras import backend as K

from tqdm import tqdm
from collections import Counter

```

1.2 Define functions for model

[2]:

```

# resize the photos
def read_and_resize(filepath, input_shape=(128, 128)):
    im = Image.open(str(filepath)).convert('RGB')
    im = im.resize(input_shape)
    im_array = np.array(im, dtype="uint8")#[..., ::-1]
    return np.array(im_array / (np.max(im_array)+ 0.001), dtype="float32")

```

[3]:

```

# data augment
datagen = ImageDataGenerator(
    rotation_range=6,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    zoom_range=0.1)

```

[4]:

```

def augment(im_array):
    im_array = datagen.random_transform(im_array)
    return im_array

```

[5]:

```

# generate images and labels for training data
def gen(df, batch_size=32, aug=False):
    df = df.sample(frac=1)

    dict_age = {'(0,18)' : 0,
                '(18,30)' : 1,
                '(30,40)' : 2,
                '(40,60)' : 3,
                '(60,100+)' : 4}

    while True:

```

```

        for i, batch in enumerate([df[i:i+batch_size] for i in range(0,df.
→shape[0],batch_size)]):
            if aug:
                images = np.array([augment(read_and_resize(file_path)) for
→file_path in batch.path.values])
            else:
                images = np.array([read_and_resize(file_path) for file_path in
→batch.path.values])

            labels = np.array([dict_age[g] for g in batch.AGE.values])
            # labels = labels[..., np.newaxis]

            yield images, labels

```

```
[6]: # get the structure of model
def get_model(n_classes=1):

    base_model = ResNet50(weights='imagenet', include_top=False)

    #for layer in base_model.layers:
    #    layer.trainable = False

    x = base_model.output
    x = GlobalMaxPooling2D()(x)
    x = Dropout(0.5)(x)
    x = Dense(100, activation="relu")(x)
    x = Dense(75, activation="relu")(x)
    x = Dense(50, activation="relu")(x)
    x = Dropout(0.5)(x)
    if n_classes == 1:
        x = Dense(n_classes, activation="sigmoid")(x)
        # x = Dense(n_classes, activation="relu")(x)
    else:
        x = Dense(n_classes, activation="softmax")(x)

    base_model = Model(base_model.input, x, name="base_model")
    if n_classes == 1:
        base_model.compile(loss="binary_crossentropy", metrics=['acc'],
→optimizer="adam")
    else:
        base_model.compile(loss="sparse_categorical_crossentropy",
→metrics=['acc'], optimizer="adam")

    return base_model

```

```
[7]: # get the file path of figures
def create_path(df, base_path):

    df['path'] = df.apply(lambda x: base_path+x['PATH'], axis=1)

    return df
```

```
[9]: import warnings
warnings.filterwarnings("ignore")
```

1.3 Load data for the model

```
[10]: csvfile = pd.read_csv('wikifinal.csv', header = 0)
for i in range(csvfile.shape[0]):
    if csvfile.AGE[i] <= 18:
        csvfile.AGE[i] = '(0,18)'
    elif csvfile.AGE[i] <= 30:
        csvfile.AGE[i] = '(18,30)'
    elif csvfile.AGE[i] <= 40:
        csvfile.AGE[i] = '(30,40)'
    elif csvfile.AGE[i] <= 60:
        csvfile.AGE[i] = '(40,60)'
    else:
        csvfile.AGE[i] = '(60,100+)'
```

1.4 Fit the model

```
[73]: # train the model
base_path = "E:/HKU BA/MSBA 7011/group project/final/crop/"

dict_age = {'(0,18)' : 0,
            '(18,30)' : 1,
            '(30,40)' : 2,
            '(40,60)' : 3,
            '(60,100+)' : 4}

bag = 3

accuracies = []

train_df, test_df = train_test_split(csvfile, test_size=0.1)
train_df = train_df.reset_index(drop=True)
test_df = test_df.reset_index(drop=True)

# insert a path to lead to aligned photos
train_df = create_path(train_df, base_path=base_path)
test_df = create_path(test_df, base_path=base_path)
```

```

cnt_ave = 0
predictions = 0

# change the test image to a array
test_images = np.array([read_and_resize(file_path) for file_path in test_df.
    ↪path.values])
test_labels = np.array([dict_age[a] for a in test_df.AGE.values])

for k in tqdm(range(bag)):

    tr_tr, tr_val = train_test_split(train_df, test_size=0.11)

    # file_path: the place to save the model
    file_path = "baseline_age"+str(cnt_ave)+".h5"

    checkpoint = ModelCheckpoint(file_path, monitor='val_acc', verbose=1,
        ↪save_best_only=True, mode='max')

    early = EarlyStopping(monitor="val_acc", mode="max", patience=20)

    reduce_on_plateau = ReduceLROnPlateau(monitor="val_acc", mode="max",
        ↪factor=0.1, patience=3)

    callbacks_list = [checkpoint, early, reduce_on_plateau] # early

    model = get_model(n_classes=len(dict_age))
    # `aug` determines whether or not to augment the images
    model.fit_generator(gen(tr_tr, aug=True), validation_data=gen(tr_val),
        ↪epochs=250, verbose=2, workers=1,
                    callbacks=callbacks_list, steps_per_epoch=100,
        ↪validation_steps = 50,use_multiprocessing=False)

    model.load_weights(file_path)

    predictions += model.predict(test_images)
    cnt_ave += 1

    K.clear_session()

predictions = predictions/cnt_ave

predictions = predictions.argmax(axis=-1)

acc = accuracy_score(test_labels, predictions)

print("accuracy : %s " %acc)

```

```
accuracies.append(acc)

0%|          | 0/3 [00:00<?, ?it/s]

Epoch 1/250
100/100 - 30s - loss: 2.0421 - acc: 0.3181 - val_loss: 1.7333 - val_acc: 0.2425

Epoch 00001: val_acc improved from -inf to 0.24250, saving model to
baseline_age0.h5
Epoch 2/250
100/100 - 22s - loss: 1.5810 - acc: 0.3534 - val_loss: 4.5723 - val_acc: 0.3880

Epoch 00002: val_acc improved from 0.24250 to 0.38798, saving model to
baseline_age0.h5
Epoch 3/250
100/100 - 22s - loss: 1.4446 - acc: 0.4084 - val_loss: 1.4857 - val_acc: 0.4087

Epoch 00003: val_acc improved from 0.38798 to 0.40875, saving model to
baseline_age0.h5
Epoch 4/250
100/100 - 22s - loss: 1.4149 - acc: 0.4206 - val_loss: 3.0242 - val_acc: 0.3980

Epoch 00004: val_acc did not improve from 0.40875
Epoch 5/250
100/100 - 22s - loss: 1.3561 - acc: 0.4531 - val_loss: 1.4972 - val_acc: 0.3886

Epoch 00005: val_acc did not improve from 0.40875
Epoch 6/250
100/100 - 22s - loss: 1.3314 - acc: 0.4663 - val_loss: 1.4714 - val_acc: 0.3938

Epoch 00006: val_acc did not improve from 0.40875
Epoch 7/250
100/100 - 22s - loss: 1.3135 - acc: 0.4700 - val_loss: 1.4544 - val_acc: 0.3930

Epoch 00007: val_acc did not improve from 0.40875
Epoch 8/250
100/100 - 22s - loss: 1.2898 - acc: 0.4806 - val_loss: 1.4483 - val_acc: 0.4006

Epoch 00008: val_acc did not improve from 0.40875
Epoch 9/250
100/100 - 22s - loss: 1.2849 - acc: 0.4734 - val_loss: 1.3838 - val_acc: 0.4312

Epoch 00009: val_acc improved from 0.40875 to 0.43116, saving model to
baseline_age0.h5
Epoch 10/250
100/100 - 22s - loss: 1.2811 - acc: 0.4791 - val_loss: 1.2895 - val_acc: 0.4593
```

```
Epoch 00010: val_acc improved from 0.43116 to 0.45932, saving model to
baseline_age0.h5
Epoch 11/250
100/100 - 22s - loss: 1.2560 - acc: 0.4981 - val_loss: 1.2319 - val_acc: 0.5019

Epoch 00011: val_acc improved from 0.45932 to 0.50187, saving model to
baseline_age0.h5
Epoch 12/250
100/100 - 22s - loss: 1.2384 - acc: 0.4972 - val_loss: 1.1817 - val_acc: 0.5275

Epoch 00012: val_acc improved from 0.50187 to 0.52753, saving model to
baseline_age0.h5
Epoch 13/250
100/100 - 22s - loss: 1.2307 - acc: 0.4991 - val_loss: 1.1519 - val_acc: 0.5362

Epoch 00013: val_acc improved from 0.52753 to 0.53625, saving model to
baseline_age0.h5
Epoch 14/250
100/100 - 22s - loss: 1.2175 - acc: 0.5131 - val_loss: 1.1447 - val_acc: 0.5463

Epoch 00014: val_acc improved from 0.53625 to 0.54631, saving model to
baseline_age0.h5
Epoch 15/250
100/100 - 22s - loss: 1.2360 - acc: 0.5044 - val_loss: 1.1781 - val_acc: 0.5175

Epoch 00015: val_acc did not improve from 0.54631
Epoch 16/250
100/100 - 22s - loss: 1.2241 - acc: 0.4969 - val_loss: 1.1334 - val_acc: 0.5494

Epoch 00016: val_acc improved from 0.54631 to 0.54937, saving model to
baseline_age0.h5
Epoch 17/250
100/100 - 22s - loss: 1.2239 - acc: 0.5044 - val_loss: 1.1520 - val_acc: 0.5300

Epoch 00017: val_acc did not improve from 0.54937
Epoch 18/250
100/100 - 22s - loss: 1.2020 - acc: 0.5250 - val_loss: 1.1303 - val_acc: 0.5350

Epoch 00018: val_acc did not improve from 0.54937
Epoch 19/250
100/100 - 22s - loss: 1.1910 - acc: 0.5184 - val_loss: 1.1431 - val_acc: 0.5401

Epoch 00019: val_acc did not improve from 0.54937
Epoch 20/250
100/100 - 22s - loss: 1.1844 - acc: 0.5300 - val_loss: 1.1178 - val_acc: 0.5413

Epoch 00020: val_acc did not improve from 0.54937
Epoch 21/250
```

```
100/100 - 22s - loss: 1.1570 - acc: 0.5382 - val_loss: 1.0969 - val_acc: 0.5669

Epoch 00021: val_acc improved from 0.54937 to 0.56687, saving model to
baseline_age0.h5
Epoch 22/250
100/100 - 22s - loss: 1.1667 - acc: 0.5281 - val_loss: 1.0884 - val_acc: 0.5726

Epoch 00022: val_acc improved from 0.56687 to 0.57259, saving model to
baseline_age0.h5
Epoch 23/250
100/100 - 22s - loss: 1.1867 - acc: 0.5281 - val_loss: 1.1012 - val_acc: 0.5537

Epoch 00023: val_acc did not improve from 0.57259
Epoch 24/250
100/100 - 22s - loss: 1.1912 - acc: 0.5188 - val_loss: 1.0801 - val_acc: 0.5751

Epoch 00024: val_acc improved from 0.57259 to 0.57509, saving model to
baseline_age0.h5
Epoch 25/250
100/100 - 22s - loss: 1.1507 - acc: 0.5497 - val_loss: 1.0991 - val_acc: 0.5576

Epoch 00025: val_acc did not improve from 0.57509
Epoch 26/250
100/100 - 22s - loss: 1.1404 - acc: 0.5362 - val_loss: 1.0789 - val_acc: 0.5669

Epoch 00026: val_acc did not improve from 0.57509
Epoch 27/250
100/100 - 22s - loss: 1.1436 - acc: 0.5431 - val_loss: 1.0732 - val_acc: 0.5713

Epoch 00027: val_acc did not improve from 0.57509
Epoch 28/250
100/100 - 22s - loss: 1.1354 - acc: 0.5469 - val_loss: 1.0894 - val_acc: 0.5606

Epoch 00028: val_acc did not improve from 0.57509
Epoch 29/250
100/100 - 22s - loss: 1.1596 - acc: 0.5312 - val_loss: 1.0692 - val_acc: 0.5719

Epoch 00029: val_acc did not improve from 0.57509
Epoch 30/250
100/100 - 22s - loss: 1.1623 - acc: 0.5319 - val_loss: 1.0850 - val_acc: 0.5613

Epoch 00030: val_acc did not improve from 0.57509
Epoch 31/250
100/100 - 22s - loss: 1.1613 - acc: 0.5322 - val_loss: 1.0631 - val_acc: 0.5750

Epoch 00031: val_acc did not improve from 0.57509
Epoch 32/250
100/100 - 22s - loss: 1.1301 - acc: 0.5491 - val_loss: 1.0694 - val_acc: 0.5776
```

```
Epoch 00032: val_acc improved from 0.57509 to 0.57760, saving model to
baseline_age0.h5
Epoch 33/250
100/100 - 22s - loss: 1.1371 - acc: 0.5450 - val_loss: 1.0891 - val_acc: 0.5569

Epoch 00033: val_acc did not improve from 0.57760
Epoch 34/250
100/100 - 22s - loss: 1.1643 - acc: 0.5463 - val_loss: 1.0689 - val_acc: 0.5719

Epoch 00034: val_acc did not improve from 0.57760
Epoch 35/250
100/100 - 22s - loss: 1.1428 - acc: 0.5422 - val_loss: 1.0811 - val_acc: 0.5632

Epoch 00035: val_acc did not improve from 0.57760
Epoch 36/250
100/100 - 22s - loss: 1.1850 - acc: 0.5209 - val_loss: 1.0707 - val_acc: 0.5650

Epoch 00036: val_acc did not improve from 0.57760
Epoch 37/250
100/100 - 22s - loss: 1.1494 - acc: 0.5375 - val_loss: 1.0689 - val_acc: 0.5782

Epoch 00037: val_acc improved from 0.57760 to 0.57822, saving model to
baseline_age0.h5
Epoch 38/250
100/100 - 22s - loss: 1.1699 - acc: 0.5422 - val_loss: 1.0860 - val_acc: 0.5544

Epoch 00038: val_acc did not improve from 0.57822
Epoch 39/250
100/100 - 22s - loss: 1.1436 - acc: 0.5450 - val_loss: 1.0572 - val_acc: 0.5856

Epoch 00039: val_acc improved from 0.57822 to 0.58562, saving model to
baseline_age0.h5
Epoch 40/250
100/100 - 22s - loss: 1.1324 - acc: 0.5619 - val_loss: 1.0820 - val_acc: 0.5663

Epoch 00040: val_acc did not improve from 0.58562
Epoch 41/250
100/100 - 22s - loss: 1.1527 - acc: 0.5394 - val_loss: 1.0761 - val_acc: 0.5612

Epoch 00041: val_acc did not improve from 0.58562
Epoch 42/250
100/100 - 22s - loss: 1.1507 - acc: 0.5381 - val_loss: 1.0726 - val_acc: 0.5770

Epoch 00042: val_acc did not improve from 0.58562
Epoch 43/250
100/100 - 22s - loss: 1.1602 - acc: 0.5325 - val_loss: 1.0815 - val_acc: 0.5601
```

```
Epoch 00043: val_acc did not improve from 0.58562
Epoch 44/250
100/100 - 22s - loss: 1.1377 - acc: 0.5422 - val_loss: 1.0654 - val_acc: 0.5763

Epoch 00044: val_acc did not improve from 0.58562
Epoch 45/250
100/100 - 22s - loss: 1.1501 - acc: 0.5516 - val_loss: 1.0735 - val_acc: 0.5763

Epoch 00045: val_acc did not improve from 0.58562
Epoch 46/250
100/100 - 22s - loss: 1.1366 - acc: 0.5362 - val_loss: 1.0761 - val_acc: 0.5638

Epoch 00046: val_acc did not improve from 0.58562
Epoch 47/250
100/100 - 22s - loss: 1.1521 - acc: 0.5428 - val_loss: 1.0721 - val_acc: 0.5688

Epoch 00047: val_acc did not improve from 0.58562
Epoch 48/250
100/100 - 22s - loss: 1.1302 - acc: 0.5497 - val_loss: 1.0846 - val_acc: 0.5607

Epoch 00048: val_acc did not improve from 0.58562
Epoch 49/250
100/100 - 22s - loss: 1.1686 - acc: 0.5269 - val_loss: 1.0738 - val_acc: 0.5700

Epoch 00049: val_acc did not improve from 0.58562
Epoch 50/250
100/100 - 22s - loss: 1.1571 - acc: 0.5375 - val_loss: 1.0676 - val_acc: 0.5776

Epoch 00050: val_acc did not improve from 0.58562
Epoch 51/250
100/100 - 22s - loss: 1.1621 - acc: 0.5309 - val_loss: 1.0868 - val_acc: 0.5625

Epoch 00051: val_acc did not improve from 0.58562
Epoch 52/250
100/100 - 22s - loss: 1.1472 - acc: 0.5472 - val_loss: 1.0627 - val_acc: 0.5776

Epoch 00052: val_acc did not improve from 0.58562
Epoch 53/250
100/100 - 22s - loss: 1.1348 - acc: 0.5425 - val_loss: 1.0822 - val_acc: 0.5645

Epoch 00053: val_acc did not improve from 0.58562
Epoch 54/250
100/100 - 22s - loss: 1.1619 - acc: 0.5469 - val_loss: 1.0745 - val_acc: 0.5663

Epoch 00054: val_acc did not improve from 0.58562
Epoch 55/250
100/100 - 22s - loss: 1.1282 - acc: 0.5476 - val_loss: 1.0627 - val_acc: 0.5782
```

```
Epoch 00055: val_acc did not improve from 0.58562
Epoch 56/250
100/100 - 22s - loss: 1.1607 - acc: 0.5353 - val_loss: 1.0866 - val_acc: 0.5619

Epoch 00056: val_acc did not improve from 0.58562
Epoch 57/250
100/100 - 22s - loss: 1.1640 - acc: 0.5241 - val_loss: 1.0686 - val_acc: 0.5719

Epoch 00057: val_acc did not improve from 0.58562
Epoch 58/250
100/100 - 22s - loss: 1.1579 - acc: 0.5312 - val_loss: 1.0837 - val_acc: 0.5620

Epoch 00058: val_acc did not improve from 0.58562
Epoch 59/250
100/100 - 22s - loss: 1.1238 - acc: 0.5653 - val_loss: 1.0626 - val_acc: 0.5750

Epoch 00059: val_acc did not improve from 0.58562
33% | 1/3 [21:51<43:42, 1311.24s/it]

Epoch 1/250
100/100 - 31s - loss: 1.8359 - acc: 0.3141 - val_loss: 3.9228 - val_acc: 0.1919

Epoch 00001: val_acc improved from -inf to 0.19187, saving model to
baseline_age1.h5
Epoch 2/250
100/100 - 22s - loss: 1.4301 - acc: 0.4041 - val_loss: 13.0788 - val_acc: 0.3942

Epoch 00002: val_acc improved from 0.19187 to 0.39424, saving model to
baseline_age1.h5
Epoch 3/250
100/100 - 22s - loss: 1.4010 - acc: 0.4384 - val_loss: 1.4725 - val_acc: 0.3956

Epoch 00003: val_acc improved from 0.39424 to 0.39562, saving model to
baseline_age1.h5
Epoch 4/250
100/100 - 22s - loss: 1.3176 - acc: 0.4694 - val_loss: 1.4842 - val_acc: 0.4011

Epoch 00004: val_acc improved from 0.39562 to 0.40113, saving model to
baseline_age1.h5
Epoch 5/250
100/100 - 22s - loss: 1.2782 - acc: 0.4881 - val_loss: 1.5691 - val_acc: 0.3886

Epoch 00005: val_acc did not improve from 0.40113
Epoch 6/250
100/100 - 22s - loss: 1.2651 - acc: 0.4872 - val_loss: 1.4248 - val_acc: 0.4106

Epoch 00006: val_acc improved from 0.40113 to 0.41063, saving model to
baseline_age1.h5
```

```
Epoch 7/250
100/100 - 22s - loss: 1.2330 - acc: 0.5041 - val_loss: 1.4529 - val_acc: 0.3930

Epoch 00007: val_acc did not improve from 0.41063
Epoch 8/250
100/100 - 22s - loss: 1.2250 - acc: 0.4997 - val_loss: 1.4693 - val_acc: 0.4162

Epoch 00008: val_acc improved from 0.41063 to 0.41625, saving model to
baseline_age1.h5
Epoch 9/250
100/100 - 22s - loss: 1.1968 - acc: 0.5256 - val_loss: 1.6980 - val_acc: 0.4255

Epoch 00009: val_acc improved from 0.41625 to 0.42553, saving model to
baseline_age1.h5
Epoch 10/250
100/100 - 22s - loss: 1.2095 - acc: 0.5113 - val_loss: 1.5371 - val_acc: 0.2847

Epoch 00010: val_acc did not improve from 0.42553
Epoch 11/250
100/100 - 22s - loss: 1.2369 - acc: 0.5038 - val_loss: 1.3213 - val_acc: 0.4363

Epoch 00011: val_acc improved from 0.42553 to 0.43625, saving model to
baseline_age1.h5
Epoch 12/250
100/100 - 22s - loss: 1.1958 - acc: 0.5184 - val_loss: 1.1963 - val_acc: 0.5106

Epoch 00012: val_acc improved from 0.43625 to 0.51064, saving model to
baseline_age1.h5
Epoch 13/250
100/100 - 22s - loss: 1.2513 - acc: 0.4944 - val_loss: 1.4392 - val_acc: 0.4075

Epoch 00013: val_acc did not improve from 0.51064
Epoch 14/250
100/100 - 22s - loss: 1.2309 - acc: 0.5031 - val_loss: 1.8978 - val_acc: 0.4180

Epoch 00014: val_acc did not improve from 0.51064
Epoch 15/250
100/100 - 22s - loss: 1.2035 - acc: 0.5194 - val_loss: 1.2190 - val_acc: 0.4937

Epoch 00015: val_acc did not improve from 0.51064
Epoch 16/250
100/100 - 22s - loss: 1.1552 - acc: 0.5434 - val_loss: 1.0846 - val_acc: 0.5612

Epoch 00016: val_acc improved from 0.51064 to 0.56125, saving model to
baseline_age1.h5
Epoch 17/250
100/100 - 22s - loss: 1.1354 - acc: 0.5366 - val_loss: 1.0819 - val_acc: 0.5482
```

```
Epoch 00017: val_acc did not improve from 0.56125
Epoch 18/250
100/100 - 22s - loss: 1.1187 - acc: 0.5456 - val_loss: 1.0703 - val_acc: 0.5644

Epoch 00018: val_acc improved from 0.56125 to 0.56437, saving model to
baseline_age1.h5
Epoch 19/250
100/100 - 22s - loss: 1.0981 - acc: 0.5566 - val_loss: 1.0141 - val_acc: 0.5832

Epoch 00019: val_acc improved from 0.56437 to 0.58323, saving model to
baseline_age1.h5
Epoch 20/250
100/100 - 22s - loss: 1.1309 - acc: 0.5409 - val_loss: 1.0516 - val_acc: 0.5820

Epoch 00020: val_acc did not improve from 0.58323
Epoch 21/250
100/100 - 22s - loss: 1.0952 - acc: 0.5497 - val_loss: 1.0289 - val_acc: 0.5756

Epoch 00021: val_acc did not improve from 0.58323
Epoch 22/250
100/100 - 22s - loss: 1.0613 - acc: 0.5769 - val_loss: 1.0149 - val_acc: 0.5857

Epoch 00022: val_acc improved from 0.58323 to 0.58573, saving model to
baseline_age1.h5
Epoch 23/250
100/100 - 22s - loss: 1.0846 - acc: 0.5666 - val_loss: 1.0447 - val_acc: 0.5738

Epoch 00023: val_acc did not improve from 0.58573
Epoch 24/250
100/100 - 22s - loss: 1.0610 - acc: 0.5728 - val_loss: 0.9844 - val_acc: 0.5864

Epoch 00024: val_acc improved from 0.58573 to 0.58636, saving model to
baseline_age1.h5
Epoch 25/250
100/100 - 22s - loss: 1.0370 - acc: 0.5838 - val_loss: 1.0035 - val_acc: 0.6014

Epoch 00025: val_acc improved from 0.58636 to 0.60138, saving model to
baseline_age1.h5
Epoch 26/250
100/100 - 22s - loss: 1.0683 - acc: 0.5688 - val_loss: 1.0004 - val_acc: 0.5875

Epoch 00026: val_acc did not improve from 0.60138
Epoch 27/250
100/100 - 22s - loss: 1.0594 - acc: 0.5684 - val_loss: 0.9975 - val_acc: 0.5901

Epoch 00027: val_acc did not improve from 0.60138
Epoch 28/250
100/100 - 22s - loss: 1.0513 - acc: 0.5779 - val_loss: 1.0198 - val_acc: 0.5838
```

```
Epoch 00028: val_acc did not improve from 0.60138
Epoch 29/250
100/100 - 22s - loss: 1.0301 - acc: 0.6047 - val_loss: 0.9580 - val_acc: 0.5938

Epoch 00029: val_acc did not improve from 0.60138
Epoch 30/250
100/100 - 22s - loss: 1.0486 - acc: 0.5706 - val_loss: 0.9907 - val_acc: 0.6033

Epoch 00030: val_acc improved from 0.60138 to 0.60325, saving model to
baseline_age1.h5
Epoch 31/250
100/100 - 22s - loss: 1.0461 - acc: 0.5797 - val_loss: 0.9747 - val_acc: 0.5950

Epoch 00031: val_acc did not improve from 0.60325
Epoch 32/250
100/100 - 22s - loss: 1.0210 - acc: 0.5969 - val_loss: 0.9803 - val_acc: 0.5951

Epoch 00032: val_acc did not improve from 0.60325
Epoch 33/250
100/100 - 22s - loss: 1.0554 - acc: 0.5766 - val_loss: 0.9763 - val_acc: 0.6020

Epoch 00033: val_acc did not improve from 0.60325
Epoch 34/250
100/100 - 22s - loss: 1.0246 - acc: 0.5936 - val_loss: 0.9545 - val_acc: 0.5962

Epoch 00034: val_acc did not improve from 0.60325
Epoch 35/250
100/100 - 22s - loss: 1.0088 - acc: 0.5922 - val_loss: 0.9855 - val_acc: 0.6008

Epoch 00035: val_acc did not improve from 0.60325
Epoch 36/250
100/100 - 22s - loss: 1.0184 - acc: 0.5919 - val_loss: 0.9700 - val_acc: 0.5962

Epoch 00036: val_acc did not improve from 0.60325
Epoch 37/250
100/100 - 22s - loss: 1.0383 - acc: 0.5803 - val_loss: 0.9814 - val_acc: 0.5895

Epoch 00037: val_acc did not improve from 0.60325
Epoch 38/250
100/100 - 22s - loss: 1.0345 - acc: 0.5888 - val_loss: 0.9768 - val_acc: 0.6058

Epoch 00038: val_acc improved from 0.60325 to 0.60576, saving model to
baseline_age1.h5
Epoch 39/250
100/100 - 22s - loss: 1.0274 - acc: 0.5872 - val_loss: 0.9585 - val_acc: 0.5975

Epoch 00039: val_acc did not improve from 0.60576
```

```
Epoch 40/250
100/100 - 22s - loss: 1.0286 - acc: 0.5778 - val_loss: 0.9874 - val_acc: 0.6001

Epoch 00040: val_acc did not improve from 0.60576
Epoch 41/250
100/100 - 22s - loss: 1.0240 - acc: 0.5857 - val_loss: 0.9720 - val_acc: 0.5938

Epoch 00041: val_acc did not improve from 0.60576
Epoch 42/250
100/100 - 22s - loss: 1.0192 - acc: 0.5863 - val_loss: 0.9587 - val_acc: 0.5970

Epoch 00042: val_acc did not improve from 0.60576
Epoch 43/250
100/100 - 22s - loss: 1.0124 - acc: 0.5944 - val_loss: 0.9844 - val_acc: 0.6008

Epoch 00043: val_acc did not improve from 0.60576
Epoch 44/250
100/100 - 22s - loss: 1.0315 - acc: 0.5809 - val_loss: 0.9617 - val_acc: 0.5975

Epoch 00044: val_acc did not improve from 0.60576
Epoch 45/250
100/100 - 22s - loss: 1.0395 - acc: 0.5838 - val_loss: 0.9798 - val_acc: 0.6014

Epoch 00045: val_acc did not improve from 0.60576
Epoch 46/250
100/100 - 22s - loss: 1.0169 - acc: 0.5919 - val_loss: 0.9897 - val_acc: 0.5869

Epoch 00046: val_acc did not improve from 0.60576
Epoch 47/250
100/100 - 23s - loss: 1.0420 - acc: 0.5788 - val_loss: 0.9490 - val_acc: 0.6026

Epoch 00047: val_acc did not improve from 0.60576
Epoch 48/250
100/100 - 22s - loss: 1.0282 - acc: 0.5851 - val_loss: 0.9848 - val_acc: 0.6033

Epoch 00048: val_acc did not improve from 0.60576
Epoch 49/250
100/100 - 22s - loss: 1.0123 - acc: 0.5878 - val_loss: 0.9659 - val_acc: 0.5969

Epoch 00049: val_acc did not improve from 0.60576
Epoch 50/250
100/100 - 22s - loss: 1.0127 - acc: 0.5934 - val_loss: 0.9739 - val_acc: 0.6026

Epoch 00050: val_acc did not improve from 0.60576
Epoch 51/250
100/100 - 22s - loss: 1.0464 - acc: 0.5844 - val_loss: 0.9827 - val_acc: 0.5950

Epoch 00051: val_acc did not improve from 0.60576
```

```
Epoch 52/250
100/100 - 22s - loss: 1.0216 - acc: 0.5888 - val_loss: 0.9445 - val_acc: 0.5995

Epoch 00052: val_acc did not improve from 0.60576
Epoch 53/250
100/100 - 22s - loss: 1.0298 - acc: 0.5878 - val_loss: 0.9823 - val_acc: 0.6058

Epoch 00053: val_acc did not improve from 0.60576
Epoch 54/250
100/100 - 22s - loss: 1.0366 - acc: 0.5816 - val_loss: 0.9642 - val_acc: 0.6000

Epoch 00054: val_acc did not improve from 0.60576
Epoch 55/250
100/100 - 22s - loss: 1.0248 - acc: 0.5845 - val_loss: 0.9765 - val_acc: 0.5982

Epoch 00055: val_acc did not improve from 0.60576
Epoch 56/250
100/100 - 22s - loss: 1.0108 - acc: 0.5969 - val_loss: 0.9875 - val_acc: 0.5931

Epoch 00056: val_acc did not improve from 0.60576
Epoch 57/250
100/100 - 22s - loss: 1.0273 - acc: 0.5909 - val_loss: 0.9444 - val_acc: 0.6019

Epoch 00057: val_acc did not improve from 0.60576
Epoch 58/250
100/100 - 22s - loss: 1.0280 - acc: 0.5922 - val_loss: 0.9842 - val_acc: 0.6039

Epoch 00058: val_acc did not improve from 0.60576
67%|          | 2/3 [43:49<21:55, 1315.42s/it]

Epoch 1/250
100/100 - 31s - loss: 1.9728 - acc: 0.3359 - val_loss: 1.4491 - val_acc: 0.4075

Epoch 00001: val_acc improved from -inf to 0.40750, saving model to
baseline_age2.h5
Epoch 2/250
100/100 - 22s - loss: 1.4855 - acc: 0.3794 - val_loss: 1.4545 - val_acc: 0.3911

Epoch 00002: val_acc did not improve from 0.40750
Epoch 3/250
100/100 - 22s - loss: 1.4484 - acc: 0.4009 - val_loss: 1.4307 - val_acc: 0.4100

Epoch 00003: val_acc improved from 0.40750 to 0.41000, saving model to
baseline_age2.h5
Epoch 4/250
100/100 - 22s - loss: 1.4189 - acc: 0.4078 - val_loss: 1.4793 - val_acc: 0.3874

Epoch 00004: val_acc did not improve from 0.41000
```

```
Epoch 5/250
100/100 - 22s - loss: 1.4204 - acc: 0.4234 - val_loss: 1.4552 - val_acc: 0.3974

Epoch 00005: val_acc did not improve from 0.41000
Epoch 6/250
100/100 - 22s - loss: 1.3802 - acc: 0.4353 - val_loss: 1.4350 - val_acc: 0.4062

Epoch 00006: val_acc did not improve from 0.41000
Epoch 7/250
100/100 - 22s - loss: 1.3371 - acc: 0.4584 - val_loss: 1.4386 - val_acc: 0.3880

Epoch 00007: val_acc did not improve from 0.41000
Epoch 8/250
100/100 - 22s - loss: 1.2962 - acc: 0.4719 - val_loss: 1.4075 - val_acc: 0.4119

Epoch 00008: val_acc improved from 0.41000 to 0.41188, saving model to
baseline_age2.h5
Epoch 9/250
100/100 - 22s - loss: 1.3081 - acc: 0.4716 - val_loss: 1.3640 - val_acc: 0.4212

Epoch 00009: val_acc improved from 0.41188 to 0.42115, saving model to
baseline_age2.h5
Epoch 10/250
100/100 - 22s - loss: 1.2763 - acc: 0.4975 - val_loss: 1.2494 - val_acc: 0.5069

Epoch 00010: val_acc improved from 0.42115 to 0.50688, saving model to
baseline_age2.h5
Epoch 11/250
100/100 - 22s - loss: 1.2710 - acc: 0.4772 - val_loss: 1.2330 - val_acc: 0.5200

Epoch 00011: val_acc improved from 0.50688 to 0.52000, saving model to
baseline_age2.h5
Epoch 12/250
100/100 - 22s - loss: 1.2773 - acc: 0.4959 - val_loss: 1.2209 - val_acc: 0.5175

Epoch 00012: val_acc did not improve from 0.52000
Epoch 13/250
100/100 - 22s - loss: 1.2596 - acc: 0.4888 - val_loss: 1.1877 - val_acc: 0.5288

Epoch 00013: val_acc improved from 0.52000 to 0.52875, saving model to
baseline_age2.h5
Epoch 14/250
100/100 - 22s - loss: 1.2367 - acc: 0.4947 - val_loss: 1.1812 - val_acc: 0.5238

Epoch 00014: val_acc did not improve from 0.52875
Epoch 15/250
100/100 - 22s - loss: 1.2367 - acc: 0.5009 - val_loss: 1.1739 - val_acc: 0.5407
```

```
Epoch 00015: val_acc improved from 0.52875 to 0.54068, saving model to
baseline_age2.h5
Epoch 16/250
100/100 - 22s - loss: 1.2188 - acc: 0.5144 - val_loss: 1.1370 - val_acc: 0.5462

Epoch 00016: val_acc improved from 0.54068 to 0.54625, saving model to
baseline_age2.h5
Epoch 17/250
100/100 - 22s - loss: 1.2268 - acc: 0.4997 - val_loss: 1.1659 - val_acc: 0.5200

Epoch 00017: val_acc did not improve from 0.54625
Epoch 18/250
100/100 - 22s - loss: 1.2136 - acc: 0.5106 - val_loss: 1.1022 - val_acc: 0.5713

Epoch 00018: val_acc improved from 0.54625 to 0.57125, saving model to
baseline_age2.h5
Epoch 19/250
100/100 - 22s - loss: 1.1943 - acc: 0.5184 - val_loss: 1.1347 - val_acc: 0.5338

Epoch 00019: val_acc did not improve from 0.57125
Epoch 20/250
100/100 - 22s - loss: 1.2018 - acc: 0.5219 - val_loss: 1.4751 - val_acc: 0.4068

Epoch 00020: val_acc did not improve from 0.57125
Epoch 21/250
100/100 - 22s - loss: 1.2138 - acc: 0.5275 - val_loss: 1.1181 - val_acc: 0.5487

Epoch 00021: val_acc did not improve from 0.57125
Epoch 22/250
100/100 - 22s - loss: 1.2209 - acc: 0.5234 - val_loss: 1.1234 - val_acc: 0.5701

Epoch 00022: val_acc did not improve from 0.57125
Epoch 23/250
100/100 - 22s - loss: 1.1960 - acc: 0.5291 - val_loss: 1.0833 - val_acc: 0.5919

Epoch 00023: val_acc improved from 0.57125 to 0.59188, saving model to
baseline_age2.h5
Epoch 24/250
100/100 - 22s - loss: 1.1791 - acc: 0.5331 - val_loss: 1.1066 - val_acc: 0.5651

Epoch 00024: val_acc did not improve from 0.59188
Epoch 25/250
100/100 - 22s - loss: 1.1751 - acc: 0.5472 - val_loss: 1.0862 - val_acc: 0.5982

Epoch 00025: val_acc improved from 0.59188 to 0.59825, saving model to
baseline_age2.h5
Epoch 26/250
100/100 - 22s - loss: 1.1784 - acc: 0.5222 - val_loss: 1.0823 - val_acc: 0.5688
```

```
Epoch 00026: val_acc did not improve from 0.59825
Epoch 27/250
100/100 - 22s - loss: 1.1887 - acc: 0.5316 - val_loss: 1.0947 - val_acc: 0.5776

Epoch 00027: val_acc did not improve from 0.59825
Epoch 28/250
100/100 - 22s - loss: 1.1463 - acc: 0.5535 - val_loss: 1.0590 - val_acc: 0.5831

Epoch 00028: val_acc did not improve from 0.59825
Epoch 29/250
100/100 - 22s - loss: 1.1815 - acc: 0.5372 - val_loss: 1.0867 - val_acc: 0.5788

Epoch 00029: val_acc did not improve from 0.59825
Epoch 30/250
100/100 - 22s - loss: 1.1634 - acc: 0.5416 - val_loss: 1.0732 - val_acc: 0.5845

Epoch 00030: val_acc did not improve from 0.59825
Epoch 31/250
100/100 - 22s - loss: 1.1735 - acc: 0.5309 - val_loss: 1.0616 - val_acc: 0.5856

Epoch 00031: val_acc did not improve from 0.59825
Epoch 32/250
100/100 - 22s - loss: 1.1687 - acc: 0.5422 - val_loss: 1.0913 - val_acc: 0.5763

Epoch 00032: val_acc did not improve from 0.59825
Epoch 33/250
100/100 - 22s - loss: 1.1738 - acc: 0.5400 - val_loss: 1.0577 - val_acc: 0.5920

Epoch 00033: val_acc did not improve from 0.59825
Epoch 34/250
100/100 - 22s - loss: 1.1821 - acc: 0.5297 - val_loss: 1.0880 - val_acc: 0.5744

Epoch 00034: val_acc did not improve from 0.59825
Epoch 35/250
100/100 - 22s - loss: 1.1560 - acc: 0.5466 - val_loss: 1.0828 - val_acc: 0.5801

Epoch 00035: val_acc did not improve from 0.59825
Epoch 36/250
100/100 - 22s - loss: 1.1745 - acc: 0.5469 - val_loss: 1.0546 - val_acc: 0.5900

Epoch 00036: val_acc did not improve from 0.59825
Epoch 37/250
100/100 - 22s - loss: 1.1619 - acc: 0.5512 - val_loss: 1.0942 - val_acc: 0.5713

Epoch 00037: val_acc did not improve from 0.59825
Epoch 38/250
100/100 - 22s - loss: 1.1727 - acc: 0.5328 - val_loss: 1.0612 - val_acc: 0.5957
```

```
Epoch 00038: val_acc did not improve from 0.59825
Epoch 39/250
100/100 - 22s - loss: 1.1569 - acc: 0.5537 - val_loss: 1.0831 - val_acc: 0.5781

Epoch 00039: val_acc did not improve from 0.59825
Epoch 40/250
100/100 - 22s - loss: 1.1752 - acc: 0.5325 - val_loss: 1.0866 - val_acc: 0.5776

Epoch 00040: val_acc did not improve from 0.59825
Epoch 41/250
100/100 - 22s - loss: 1.1694 - acc: 0.5347 - val_loss: 1.0474 - val_acc: 0.5987

Epoch 00041: val_acc improved from 0.59825 to 0.59875, saving model to
baseline_age2.h5
Epoch 42/250
100/100 - 22s - loss: 1.1505 - acc: 0.5469 - val_loss: 1.1008 - val_acc: 0.5620

Epoch 00042: val_acc did not improve from 0.59875
Epoch 43/250
100/100 - 22s - loss: 1.1772 - acc: 0.5484 - val_loss: 1.0641 - val_acc: 0.5957

Epoch 00043: val_acc did not improve from 0.59875
Epoch 44/250
100/100 - 22s - loss: 1.1585 - acc: 0.5478 - val_loss: 1.0759 - val_acc: 0.5756

Epoch 00044: val_acc did not improve from 0.59875
Epoch 45/250
100/100 - 22s - loss: 1.1646 - acc: 0.5306 - val_loss: 1.0847 - val_acc: 0.5801

Epoch 00045: val_acc did not improve from 0.59875
Epoch 46/250
100/100 - 22s - loss: 1.1637 - acc: 0.5447 - val_loss: 1.0477 - val_acc: 0.5994

Epoch 00046: val_acc improved from 0.59875 to 0.59938, saving model to
baseline_age2.h5
Epoch 47/250
100/100 - 22s - loss: 1.1779 - acc: 0.5353 - val_loss: 1.0941 - val_acc: 0.5682

Epoch 00047: val_acc did not improve from 0.59938
Epoch 48/250
100/100 - 22s - loss: 1.1554 - acc: 0.5447 - val_loss: 1.0756 - val_acc: 0.5914

Epoch 00048: val_acc did not improve from 0.59938
Epoch 49/250
100/100 - 22s - loss: 1.1727 - acc: 0.5384 - val_loss: 1.0675 - val_acc: 0.5769

Epoch 00049: val_acc did not improve from 0.59938
```

```
Epoch 50/250
100/100 - 22s - loss: 1.1723 - acc: 0.5481 - val_loss: 1.0889 - val_acc: 0.5807

Epoch 00050: val_acc did not improve from 0.59938
Epoch 51/250
100/100 - 22s - loss: 1.1801 - acc: 0.5384 - val_loss: 1.0563 - val_acc: 0.5881

Epoch 00051: val_acc did not improve from 0.59938
Epoch 52/250
100/100 - 22s - loss: 1.1675 - acc: 0.5403 - val_loss: 1.0867 - val_acc: 0.5751

Epoch 00052: val_acc did not improve from 0.59938
Epoch 53/250
100/100 - 22s - loss: 1.1514 - acc: 0.5375 - val_loss: 1.0709 - val_acc: 0.5907

Epoch 00053: val_acc did not improve from 0.59938
Epoch 54/250
100/100 - 22s - loss: 1.1692 - acc: 0.5369 - val_loss: 1.0648 - val_acc: 0.5788

Epoch 00054: val_acc did not improve from 0.59938
Epoch 55/250
100/100 - 23s - loss: 1.1553 - acc: 0.5488 - val_loss: 1.0852 - val_acc: 0.5801

Epoch 00055: val_acc did not improve from 0.59938
Epoch 56/250
100/100 - 22s - loss: 1.1701 - acc: 0.5434 - val_loss: 1.0563 - val_acc: 0.5906

Epoch 00056: val_acc did not improve from 0.59938
Epoch 57/250
100/100 - 22s - loss: 1.1727 - acc: 0.5437 - val_loss: 1.0858 - val_acc: 0.5775

Epoch 00057: val_acc did not improve from 0.59938
Epoch 58/250
100/100 - 22s - loss: 1.1541 - acc: 0.5387 - val_loss: 1.0734 - val_acc: 0.5857

Epoch 00058: val_acc did not improve from 0.59938
Epoch 59/250
100/100 - 22s - loss: 1.1802 - acc: 0.5328 - val_loss: 1.0616 - val_acc: 0.5856

Epoch 00059: val_acc did not improve from 0.59938
Epoch 60/250
100/100 - 22s - loss: 1.1561 - acc: 0.5337 - val_loss: 1.0911 - val_acc: 0.5795

Epoch 00060: val_acc did not improve from 0.59938
Epoch 61/250
100/100 - 22s - loss: 1.1675 - acc: 0.5381 - val_loss: 1.0602 - val_acc: 0.5901

Epoch 00061: val_acc did not improve from 0.59938
```

```

Epoch 62/250
100/100 - 22s - loss: 1.1485 - acc: 0.5538 - val_loss: 1.0872 - val_acc: 0.5744

Epoch 00062: val_acc did not improve from 0.59938
Epoch 63/250
100/100 - 22s - loss: 1.1902 - acc: 0.5306 - val_loss: 1.0824 - val_acc: 0.5807

Epoch 00063: val_acc did not improve from 0.59938
Epoch 64/250
100/100 - 22s - loss: 1.1686 - acc: 0.5466 - val_loss: 1.0534 - val_acc: 0.5906

Epoch 00064: val_acc did not improve from 0.59938
Epoch 65/250
100/100 - 22s - loss: 1.1669 - acc: 0.5362 - val_loss: 1.0937 - val_acc: 0.5713

Epoch 00065: val_acc did not improve from 0.59938
Epoch 66/250
100/100 - 23s - loss: 1.1708 - acc: 0.5356 - val_loss: 1.0623 - val_acc: 0.5976

Epoch 00066: val_acc did not improve from 0.59938
100%|     | 3/3 [1:08:54<00:00, 1378.12s/it]
accuracy : 0.5915960191669738

```

1.5 Number of trainable parameters

[34]: model.summary()

```

Model: "base_model"
-----
Layer (type)          Output Shape         Param #  Connected to
=====
input_3 (InputLayer) [(None, None, None, 0
-----
conv1_pad (ZeroPadding2D)      (None, None, None, 3 0           input_3[0] [0]
-----
conv1_conv (Conv2D)          (None, None, None, 6 9472        conv1_pad[0] [0]
-----
conv1_bn (BatchNormalization) (None, None, None, 6 256
conv1_conv[0] [0]
-----
```

```
-----  
conv1_relu (Activation)      (None, None, None, 6 0)      conv1_bn[0] [0]  
-----  
-----  
pool1_pad (ZeroPadding2D)    (None, None, None, 6 0)  
conv1_relu[0] [0]  
-----  
-----  
pool1_pool (MaxPooling2D)    (None, None, None, 6 0)      pool1_pad[0] [0]  
-----  
-----  
conv2_block1_1_conv (Conv2D)  (None, None, None, 6 4160)  
pool1_pool[0] [0]  
-----  
-----  
conv2_block1_1_bn (BatchNormali (None, None, None, 6 256  
conv2_block1_1_conv[0] [0]  
-----  
-----  
conv2_block1_1_relu (Activation (None, None, None, 6 0  
conv2_block1_1_bn[0] [0]  
-----  
-----  
conv2_block1_2_conv (Conv2D)  (None, None, None, 6 36928  
conv2_block1_1_relu[0] [0]  
-----  
-----  
conv2_block1_2_bn (BatchNormali (None, None, None, 6 256  
conv2_block1_2_conv[0] [0]  
-----  
-----  
conv2_block1_2_relu (Activation (None, None, None, 6 0  
conv2_block1_2_bn[0] [0]  
-----  
-----  
conv2_block1_0_conv (Conv2D)  (None, None, None, 2 16640  
pool1_pool[0] [0]  
-----  
-----  
conv2_block1_3_conv (Conv2D)  (None, None, None, 2 16640  
conv2_block1_2_relu[0] [0]  
-----  
-----  
conv2_block1_0_bn (BatchNormali (None, None, None, 2 1024  
conv2_block1_0_conv[0] [0]  
-----  
-----  
conv2_block1_3_bn (BatchNormali (None, None, None, 2 1024
```

```
conv2_block1_3_conv[0] [0]
-----
conv2_block1_add (Add)          (None, None, None, 2 0
conv2_block1_0_bn[0] [0]
conv2_block1_3_bn[0] [0]
-----
conv2_block1_out (Activation)  (None, None, None, 2 0
conv2_block1_add[0] [0]
-----
conv2_block2_1_conv (Conv2D)    (None, None, None, 6 16448
conv2_block1_out[0] [0]
-----
conv2_block2_1_bn (BatchNormali (None, None, None, 6 256
conv2_block2_1_conv[0] [0]
-----
conv2_block2_1_relu (Activation (None, None, None, 6 0
conv2_block2_1_bn[0] [0]
-----
conv2_block2_2_conv (Conv2D)    (None, None, None, 6 36928
conv2_block2_1_relu[0] [0]
-----
conv2_block2_2_bn (BatchNormali (None, None, None, 6 256
conv2_block2_2_conv[0] [0]
-----
conv2_block2_2_relu (Activation (None, None, None, 6 0
conv2_block2_2_bn[0] [0]
-----
conv2_block2_3_conv (Conv2D)    (None, None, None, 2 16640
conv2_block2_2_relu[0] [0]
-----
conv2_block2_3_bn (BatchNormali (None, None, None, 2 1024
conv2_block2_3_conv[0] [0]
-----
conv2_block2_add (Add)          (None, None, None, 2 0
conv2_block1_out[0] [0]
conv2_block2_3_bn[0] [0]
```

```
-----  
conv2_block2_out (Activation)      (None, None, None, 2 0  
conv2_block2_add[0] [0]  
  
-----  
conv2_block3_1_conv (Conv2D)      (None, None, None, 6 16448  
conv2_block2_out[0] [0]  
  
-----  
conv2_block3_1_bn (BatchNormali (None, None, None, 6 256  
conv2_block3_1_conv[0] [0]  
  
-----  
conv2_block3_1_relu (Activation (None, None, None, 6 0  
conv2_block3_1_bn[0] [0]  
  
-----  
conv2_block3_2_conv (Conv2D)      (None, None, None, 6 36928  
conv2_block3_1_relu[0] [0]  
  
-----  
conv2_block3_2_bn (BatchNormali (None, None, None, 6 256  
conv2_block3_2_conv[0] [0]  
  
-----  
conv2_block3_2_relu (Activation (None, None, None, 6 0  
conv2_block3_2_bn[0] [0]  
  
-----  
conv2_block3_3_conv (Conv2D)      (None, None, None, 2 16640  
conv2_block3_2_relu[0] [0]  
  
-----  
conv2_block3_3_bn (BatchNormali (None, None, None, 2 1024  
conv2_block3_3_conv[0] [0]  
  
-----  
conv2_block3_add (Add)           (None, None, None, 2 0  
conv2_block2_out[0] [0]  
conv2_block3_3_bn[0] [0]  
  
-----  
conv2_block3_out (Activation)    (None, None, None, 2 0  
conv2_block3_add[0] [0]  
  
-----  
conv3_block1_1_conv (Conv2D)     (None, None, None, 1 32896  
conv2_block3_out[0] [0]
```

```
-----  
conv3_block1_1_bn (BatchNormali (None, None, None, 1 512  
conv3_block1_1_conv[0] [0]  
  
-----  
conv3_block1_1_relu (Activation (None, None, None, 1 0  
conv3_block1_1_bn[0] [0]  
  
-----  
conv3_block1_2_conv (Conv2D)      (None, None, None, 1 147584  
conv3_block1_1_relu[0] [0]  
  
-----  
conv3_block1_2_bn (BatchNormali (None, None, None, 1 512  
conv3_block1_2_conv[0] [0]  
  
-----  
conv3_block1_2_relu (Activation (None, None, None, 1 0  
conv3_block1_2_bn[0] [0]  
  
-----  
conv3_block1_0_conv (Conv2D)      (None, None, None, 5 131584  
conv2_block3_out[0] [0]  
  
-----  
conv3_block1_3_conv (Conv2D)      (None, None, None, 5 66048  
conv3_block1_2_relu[0] [0]  
  
-----  
conv3_block1_0_bn (BatchNormali (None, None, None, 5 2048  
conv3_block1_0_conv[0] [0]  
  
-----  
conv3_block1_3_bn (BatchNormali (None, None, None, 5 2048  
conv3_block1_3_conv[0] [0]  
  
-----  
conv3_block1_add (Add)           (None, None, None, 5 0  
conv3_block1_0_bn[0] [0]  
conv3_block1_3_bn[0] [0]  
  
-----  
conv3_block1_out (Activation)   (None, None, None, 5 0  
conv3_block1_add[0] [0]  
  
-----  
conv3_block2_1_conv (Conv2D)     (None, None, None, 1 65664
```

```
conv3_block1_out[0] [0]
-----
conv3_block2_1_bn (BatchNormali (None, None, None, 1 512
conv3_block2_1_conv[0] [0]
-----
conv3_block2_1_relu (Activation (None, None, None, 1 0
conv3_block2_1_bn[0] [0]
-----
conv3_block2_2_conv (Conv2D)      (None, None, None, 1 147584
conv3_block2_1_relu[0] [0]
-----
conv3_block2_2_bn (BatchNormali (None, None, None, 1 512
conv3_block2_2_conv[0] [0]
-----
conv3_block2_2_relu (Activation (None, None, None, 1 0
conv3_block2_2_bn[0] [0]
-----
conv3_block2_3_conv (Conv2D)      (None, None, None, 5 66048
conv3_block2_2_relu[0] [0]
-----
conv3_block2_3_bn (BatchNormali (None, None, None, 5 2048
conv3_block2_3_conv[0] [0]
-----
conv3_block2_add (Add)           (None, None, None, 5 0
conv3_block1_out[0] [0]
conv3_block2_3_bn[0] [0]
-----
conv3_block2_out (Activation)    (None, None, None, 5 0
conv3_block2_add[0] [0]
-----
conv3_block3_1_conv (Conv2D)     (None, None, None, 1 65664
conv3_block2_out[0] [0]
-----
conv3_block3_1_bn (BatchNormali (None, None, None, 1 512
conv3_block3_1_conv[0] [0]
```

```
conv3_block3_1_relu (Activation (None, None, None, 1 0
conv3_block3_1_bn[0] [0]
-----
-----
conv3_block3_2_conv (Conv2D)      (None, None, None, 1 147584
conv3_block3_1_relu[0] [0]
-----
-----
conv3_block3_2_bn (BatchNormali (None, None, None, 1 512
conv3_block3_2_conv[0] [0]
-----
-----
conv3_block3_2_relu (Activation (None, None, None, 1 0
conv3_block3_2_bn[0] [0]
-----
-----
conv3_block3_3_conv (Conv2D)      (None, None, None, 5 66048
conv3_block3_2_relu[0] [0]
-----
-----
conv3_block3_3_bn (BatchNormali (None, None, None, 5 2048
conv3_block3_3_conv[0] [0]
-----
-----
conv3_block3_add (Add)          (None, None, None, 5 0
conv3_block2_out[0] [0]
conv3_block3_3_bn[0] [0]
-----
-----
conv3_block3_out (Activation)   (None, None, None, 5 0
conv3_block3_add[0] [0]
-----
-----
conv3_block4_1_conv (Conv2D)     (None, None, None, 1 65664
conv3_block3_out[0] [0]
-----
-----
conv3_block4_1_bn (BatchNormali (None, None, None, 1 512
conv3_block4_1_conv[0] [0]
-----
-----
conv3_block4_1_relu (Activation (None, None, None, 1 0
conv3_block4_1_bn[0] [0]
-----
-----
conv3_block4_2_conv (Conv2D)     (None, None, None, 1 147584
conv3_block4_1_relu[0] [0]
```

```
-----  
conv3_block4_2_bn (BatchNormali (None, None, None, 1 512  
conv3_block4_2_conv[0] [0]  
-----  
-----  
conv3_block4_2_relu (Activation (None, None, None, 1 0  
conv3_block4_2_bn[0] [0]  
-----  
-----  
conv3_block4_3_conv (Conv2D)      (None, None, None, 5 66048  
conv3_block4_2_relu[0] [0]  
-----  
-----  
conv3_block4_3_bn (BatchNormali (None, None, None, 5 2048  
conv3_block4_3_conv[0] [0]  
-----  
-----  
conv3_block4_add (Add)           (None, None, None, 5 0  
conv3_block3_out[0] [0]  
conv3_block4_3_bn[0] [0]  
-----  
-----  
conv3_block4_out (Activation)    (None, None, None, 5 0  
conv3_block4_add[0] [0]  
-----  
-----  
conv4_block1_1_conv (Conv2D)     (None, None, None, 2 131328  
conv3_block4_out[0] [0]  
-----  
-----  
conv4_block1_1_bn (BatchNormali (None, None, None, 2 1024  
conv4_block1_1_conv[0] [0]  
-----  
-----  
conv4_block1_1_relu (Activation (None, None, None, 2 0  
conv4_block1_1_bn[0] [0]  
-----  
-----  
conv4_block1_2_conv (Conv2D)     (None, None, None, 2 590080  
conv4_block1_1_relu[0] [0]  
-----  
-----  
conv4_block1_2_bn (BatchNormali (None, None, None, 2 1024  
conv4_block1_2_conv[0] [0]  
-----  
-----  
conv4_block1_2_relu (Activation (None, None, None, 2 0  
conv4_block1_2_bn[0] [0]
```

```
-----  
conv4_block1_0_conv (Conv2D)      (None, None, None, 1 525312  
conv3_block4_out[0] [0]  
  
-----  
conv4_block1_3_conv (Conv2D)      (None, None, None, 1 263168  
conv4_block1_2_relu[0] [0]  
  
-----  
conv4_block1_0_bn (BatchNormali (None, None, None, 1 4096  
conv4_block1_0_conv[0] [0]  
  
-----  
conv4_block1_3_bn (BatchNormali (None, None, None, 1 4096  
conv4_block1_3_conv[0] [0]  
  
-----  
conv4_block1_add (Add)           (None, None, None, 1 0  
conv4_block1_0_bn[0] [0]  
conv4_block1_3_bn[0] [0]  
  
-----  
conv4_block1_out (Activation)    (None, None, None, 1 0  
conv4_block1_add[0] [0]  
  
-----  
conv4_block2_1_conv (Conv2D)      (None, None, None, 2 262400  
conv4_block1_out[0] [0]  
  
-----  
conv4_block2_1_bn (BatchNormali (None, None, None, 2 1024  
conv4_block2_1_conv[0] [0]  
  
-----  
conv4_block2_1_relu (Activation (None, None, None, 2 0  
conv4_block2_1_bn[0] [0]  
  
-----  
conv4_block2_2_conv (Conv2D)      (None, None, None, 2 590080  
conv4_block2_1_relu[0] [0]  
  
-----  
conv4_block2_2_bn (BatchNormali (None, None, None, 2 1024  
conv4_block2_2_conv[0] [0]  
  
-----  
conv4_block2_2_relu (Activation (None, None, None, 2 0
```

```
conv4_block2_2_bn[0] [0]
-----
----- conv4_block2_3_conv (Conv2D)      (None, None, None, 1 263168
conv4_block2_2_relu[0] [0]
-----
----- conv4_block2_3_bn (BatchNormali (None, None, None, 1 4096
conv4_block2_3_conv[0] [0]
-----
----- conv4_block2_add (Add)          (None, None, None, 1 0
conv4_block1_out[0] [0]
conv4_block2_3_bn[0] [0]
-----
----- conv4_block2_out (Activation)  (None, None, None, 1 0
conv4_block2_add[0] [0]
-----
----- conv4_block3_1_conv (Conv2D)    (None, None, None, 2 262400
conv4_block2_out[0] [0]
-----
----- conv4_block3_1_bn (BatchNormali (None, None, None, 2 1024
conv4_block3_1_conv[0] [0]
-----
----- conv4_block3_1_relu (Activation (None, None, None, 2 0
conv4_block3_1_bn[0] [0]
-----
----- conv4_block3_2_conv (Conv2D)    (None, None, None, 2 590080
conv4_block3_1_relu[0] [0]
-----
----- conv4_block3_2_bn (BatchNormali (None, None, None, 2 1024
conv4_block3_2_conv[0] [0]
-----
----- conv4_block3_2_relu (Activation (None, None, None, 2 0
conv4_block3_2_bn[0] [0]
-----
----- conv4_block3_3_conv (Conv2D)    (None, None, None, 1 263168
conv4_block3_2_relu[0] [0]
```

```
conv4_block3_3_bn (BatchNormali (None, None, None, 1 4096
conv4_block3_3_conv[0] [0]
-----
----- conv4_block3_add (Add) (None, None, None, 1 0
conv4_block2_out[0] [0]
conv4_block3_3_bn[0] [0]
-----
----- conv4_block3_out (Activation) (None, None, None, 1 0
conv4_block3_add[0] [0]
-----
----- conv4_block4_1_conv (Conv2D) (None, None, None, 2 262400
conv4_block3_out[0] [0]
-----
----- conv4_block4_1_bn (BatchNormali (None, None, None, 2 1024
conv4_block4_1_conv[0] [0]
-----
----- conv4_block4_1_relu (Activation (None, None, None, 2 0
conv4_block4_1_bn[0] [0]
-----
----- conv4_block4_2_conv (Conv2D) (None, None, None, 2 590080
conv4_block4_1_relu[0] [0]
-----
----- conv4_block4_2_bn (BatchNormali (None, None, None, 2 1024
conv4_block4_2_conv[0] [0]
-----
----- conv4_block4_2_relu (Activation (None, None, None, 2 0
conv4_block4_2_bn[0] [0]
-----
----- conv4_block4_3_conv (Conv2D) (None, None, None, 1 263168
conv4_block4_2_relu[0] [0]
-----
----- conv4_block4_3_bn (BatchNormali (None, None, None, 1 4096
conv4_block4_3_conv[0] [0]
-----
----- conv4_block4_add (Add) (None, None, None, 1 0
conv4_block3_out[0] [0]
conv4_block4_3_bn[0] [0]
```

```
-----  
conv4_block4_out (Activation)      (None, None, None, 1 0  
conv4_block4_add[0] [0]  
  
-----  
conv4_block5_1_conv (Conv2D)      (None, None, None, 2 262400  
conv4_block4_out[0] [0]  
  
-----  
conv4_block5_1_bn (BatchNormali (None, None, None, 2 1024  
conv4_block5_1_conv[0] [0]  
  
-----  
conv4_block5_1_relu (Activation (None, None, None, 2 0  
conv4_block5_1_bn[0] [0]  
  
-----  
conv4_block5_2_conv (Conv2D)      (None, None, None, 2 590080  
conv4_block5_1_relu[0] [0]  
  
-----  
conv4_block5_2_bn (BatchNormali (None, None, None, 2 1024  
conv4_block5_2_conv[0] [0]  
  
-----  
conv4_block5_2_relu (Activation (None, None, None, 2 0  
conv4_block5_2_bn[0] [0]  
  
-----  
conv4_block5_3_conv (Conv2D)      (None, None, None, 1 263168  
conv4_block5_2_relu[0] [0]  
  
-----  
conv4_block5_3_bn (BatchNormali (None, None, None, 1 4096  
conv4_block5_3_conv[0] [0]  
  
-----  
conv4_block5_add (Add)           (None, None, None, 1 0  
conv4_block4_out[0] [0]  
conv4_block5_3_bn[0] [0]  
  
-----  
conv4_block5_out (Activation)    (None, None, None, 1 0  
conv4_block5_add[0] [0]  
  
-----  
conv4_block6_1_conv (Conv2D)     (None, None, None, 2 262400
```

```
conv4_block5_out[0] [0]
-----
conv4_block6_1_bn (BatchNormali (None, None, None, 2 1024
conv4_block6_1_conv[0] [0]
-----
conv4_block6_1_relu (Activation (None, None, None, 2 0
conv4_block6_1_bn[0] [0]
-----
conv4_block6_2_conv (Conv2D)      (None, None, None, 2 590080
conv4_block6_1_relu[0] [0]
-----
conv4_block6_2_bn (BatchNormali (None, None, None, 2 1024
conv4_block6_2_conv[0] [0]
-----
conv4_block6_2_relu (Activation (None, None, None, 2 0
conv4_block6_2_bn[0] [0]
-----
conv4_block6_3_conv (Conv2D)      (None, None, None, 1 263168
conv4_block6_2_relu[0] [0]
-----
conv4_block6_3_bn (BatchNormali (None, None, None, 1 4096
conv4_block6_3_conv[0] [0]
-----
conv4_block6_add (Add)           (None, None, None, 1 0
conv4_block5_out[0] [0]
conv4_block6_3_bn[0] [0]
-----
conv4_block6_out (Activation)   (None, None, None, 1 0
conv4_block6_add[0] [0]
-----
conv5_block1_1_conv (Conv2D)     (None, None, None, 5 524800
conv4_block6_out[0] [0]
-----
conv5_block1_1_bn (BatchNormali (None, None, None, 5 2048
conv5_block1_1_conv[0] [0]
```

```
conv5_block1_1_relu (Activation (None, None, None, 5 0
conv5_block1_1_bn[0] [0]
-----
-----
conv5_block1_2_conv (Conv2D)      (None, None, None, 5 2359808
conv5_block1_1_relu[0] [0]
-----
-----
conv5_block1_2_bn (BatchNormali (None, None, None, 5 2048
conv5_block1_2_conv[0] [0]
-----
-----
conv5_block1_2_relu (Activation (None, None, None, 5 0
conv5_block1_2_bn[0] [0]
-----
-----
conv5_block1_0_conv (Conv2D)      (None, None, None, 2 2099200
conv4_block6_out[0] [0]
-----
-----
conv5_block1_3_conv (Conv2D)      (None, None, None, 2 1050624
conv5_block1_2_relu[0] [0]
-----
-----
conv5_block1_0_bn (BatchNormali (None, None, None, 2 8192
conv5_block1_0_conv[0] [0]
-----
-----
conv5_block1_3_bn (BatchNormali (None, None, None, 2 8192
conv5_block1_3_conv[0] [0]
-----
-----
conv5_block1_add (Add)          (None, None, None, 2 0
conv5_block1_0_bn[0] [0]
conv5_block1_3_bn[0] [0]
-----
-----
conv5_block1_out (Activation)   (None, None, None, 2 0
conv5_block1_add[0] [0]
-----
-----
conv5_block2_1_conv (Conv2D)    (None, None, None, 5 1049088
conv5_block1_out[0] [0]
-----
-----
conv5_block2_1_bn (BatchNormali (None, None, None, 5 2048
conv5_block2_1_conv[0] [0]
```

```
-----  
conv5_block2_1_relu (Activation (None, None, None, 5 0  
conv5_block2_1_bn[0] [0]  
  
-----  
conv5_block2_2_conv (Conv2D)      (None, None, None, 5 2359808  
conv5_block2_1_relu[0] [0]  
  
-----  
conv5_block2_2_bn (BatchNormali (None, None, None, 5 2048  
conv5_block2_2_conv[0] [0]  
  
-----  
conv5_block2_2_relu (Activation (None, None, None, 5 0  
conv5_block2_2_bn[0] [0]  
  
-----  
conv5_block2_3_conv (Conv2D)      (None, None, None, 2 1050624  
conv5_block2_2_relu[0] [0]  
  
-----  
conv5_block2_3_bn (BatchNormali (None, None, None, 2 8192  
conv5_block2_3_conv[0] [0]  
  
-----  
conv5_block2_add (Add)           (None, None, None, 2 0  
conv5_block1_out[0] [0]  
conv5_block2_3_bn[0] [0]  
  
-----  
conv5_block2_out (Activation)    (None, None, None, 2 0  
conv5_block2_add[0] [0]  
  
-----  
conv5_block3_1_conv (Conv2D)     (None, None, None, 5 1049088  
conv5_block2_out[0] [0]  
  
-----  
conv5_block3_1_bn (BatchNormali (None, None, None, 5 2048  
conv5_block3_1_conv[0] [0]  
  
-----  
conv5_block3_1_relu (Activation (None, None, None, 5 0  
conv5_block3_1_bn[0] [0]  
  
-----  
conv5_block3_2_conv (Conv2D)     (None, None, None, 5 2359808  
conv5_block3_1_relu[0] [0]
```

```

-----  

conv5_block3_2_bn (BatchNormali (None, None, None, 5 2048  

conv5_block3_2_conv[0] [0]  

-----  

conv5_block3_2_relu (Activation (None, None, None, 5 0  

conv5_block3_2_bn[0] [0]  

-----  

conv5_block3_3_conv (Conv2D)      (None, None, None, 2 1050624  

conv5_block3_2_relu[0] [0]  

-----  

conv5_block3_3_bn (BatchNormali (None, None, None, 2 8192  

conv5_block3_3_conv[0] [0]  

-----  

conv5_block3_add (Add)           (None, None, None, 2 0  

conv5_block2_out[0] [0]  

conv5_block3_3_bn[0] [0]  

-----  

conv5_block3_out (Activation)   (None, None, None, 2 0  

conv5_block3_add[0] [0]  

-----  

global_max_pooling2d_2 (GlobalM (None, 2048)          0  

conv5_block3_out[0] [0]  

-----  

dropout_4 (Dropout)             (None, 2048)          0  

global_max_pooling2d_2[0] [0]  

-----  

dense_8 (Dense)                (None, 100)          204900    dropout_4[0] [0]  

-----  

dense_9 (Dense)                (None, 75)           7575      dense_8[0] [0]  

-----  

dense_10 (Dense)               (None, 50)           3800      dense_9[0] [0]  

-----  

dropout_5 (Dropout)             (None, 50)           0         dense_10[0] [0]  

-----  

dense_11 (Dense)               (None, 5)            255       dropout_5[0] [0]

```

```
=====
=====
Total params: 23,804,242
Trainable params: 23,751,122
Non-trainable params: 53,120
```

1.6 Plot the result

```
[13]: predictions = model.predict(test_images)
```

```
[14]: import matplotlib.pyplot as plt
```

```
[15]: labels = {
    0: '(0,18)',
    1: '(18,30)',
    2: '(30,40)',
    3: '(40,60)',
    4: '(60,100+)'
}
```

```
[20]: # Plot a random sample of 12 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(9, 12))
for i, index in enumerate(np.random.choice(test_images.shape[0], size=12, replace=False)):
    ax = figure.add_subplot(4, 3, i + 1, xticks=[], yticks[])
    # Display each image
    ax.imshow(np.squeeze(test_images[index]))
    predict_index = np.argmax(predictions[index])
    true_index = test_labels[index]
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                color=("green" if predict_index == true_index
                else "red"))
plt.show()
```



```
[16]: from sklearn.metrics import confusion_matrix  
import itertools
```

```
[17]: # Look at confusion matrix

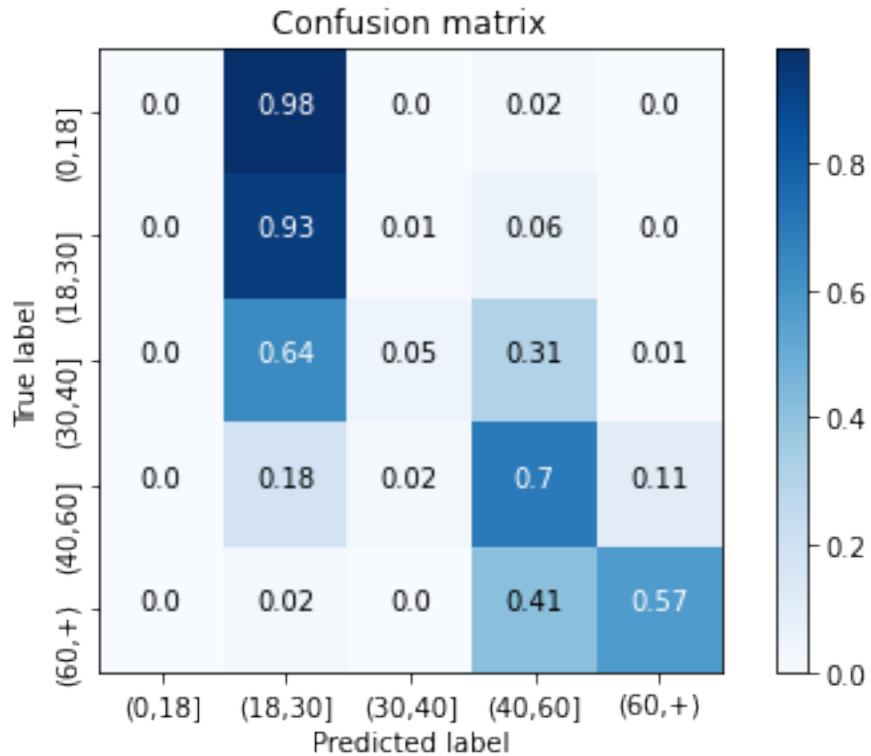
def plot_confusion_matrix(cm, classes,
                         normalize=False,
                         title='Confusion matrix',
                         cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, ['(0,18]', '(18,30]', '(30,40]', '(40,60]', '(60,+]'])
    plt.yticks(tick_marks, ['(0,18]', '(18,30]', '(30,40]', '(40,60]', '(60,+]'],
               rotation=90)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, round(cm[i, j],2),
                 horizontalalignment="center",
                 color="white" if round(cm[i, j],2) > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

# Predict the values from the validation dataset
Y_pred = predictions
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred, axis = 1)
# Convert validation observations to one hot vectors
Y_true = test_labels
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(5),normalize=True)
```



2 gender

2.1 Define functions for model

```
[22]: def read_and_resize(filepath, input_shape=(128, 128)):
    im = Image.open(str(filepath)).convert('RGB')
    im = im.resize(input_shape)
    im_array = np.array(im, dtype="uint8")#[..., ::-1]
    return np.array(im_array / (np.max(im_array)+ 0.001), dtype="float32")
```

```
[23]: datagen = ImageDataGenerator(
    rotation_range=6,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    zoom_range=0.1)
```

```
[24]: def augment(im_array):
    im_array = datagen.random_transform(im_array)
    return im_array
```

```
[25]: def gen(df, batch_size=32, aug=False):
    df = df.sample(frac=1)

    while True:
        for i, batch in enumerate([df[i:i+batch_size] for i in range(0,df.
        ↪shape[0],batch_size)]):
            if aug:
                images = np.array([augment(read_and_resize(file_path)) for
                ↪file_path in batch.path.values])
            else:
                images = np.array([read_and_resize(file_path) for file_path in
                ↪batch.path.values])

            labels = np.array(batch.GENDER.values)
            # labels = labels[..., np.newaxis]

            yield images, labels
```

```
[26]: def get_model(n_classes=1):

    base_model = ResNet50(weights='imagenet', include_top=False)

    #for layer in base_model.layers:
    #    layer.trainable = False

    x = base_model.output
    x = GlobalMaxPooling2D()(x)
    x = Dropout(0.5)(x)
    x = Dense(100, activation="relu")(x)
    x = Dense(75, activation="relu")(x)
    x = Dense(50, activation="relu")(x)
    x = Dropout(0.5)(x)
    if n_classes == 1:
        x = Dense(n_classes, activation="sigmoid")(x)
        # x = Dense(n_classes, activation="relu")(x)
    else:
        x = Dense(n_classes, activation="softmax")(x)

    base_model = Model(base_model.input, x, name="base_model")
    if n_classes == 1:
        base_model.compile(loss="binary_crossentropy", metrics=['acc'],
        ↪optimizer="adam")
    else:
        base_model.compile(loss="sparse_categorical_crossentropy",
        ↪metrics=['acc'], optimizer="adam")
```

```
    return base_model
```

```
[27]: def create_path(df, base_path):  
    df['path'] = df.apply(lambda x: base_path+x['PATH'], axis=1)  
    return df
```

2.2 Fit the model

```
[48]: base_path = "C:/Users/msba/7011/group project/crop/"  
  
bag = 3  
  
accuracies = []  
  
cnt_ave = 0  
predictions_gender = 0  
  
# change the test image to a array  
test_images = np.array([read_and_resize(file_path) for file_path in test_df.  
                      ↪path.values])  
test_labels = np.array(test_df.GENDER.values)  
  
for k in tqdm(range(bag)):  
  
    tr_tr, tr_val = train_test_split(train_df, test_size=0.11)  
  
    # file_path: the place to save the model  
    file_path = "baseline_gender_"+str(cnt_ave)+".h5"  
  
    checkpoint = ModelCheckpoint(file_path, monitor='val_acc', verbose=1,  
                                 ↪save_best_only=True, mode='max')  
  
    early = EarlyStopping(monitor="val_acc", mode="max", patience=20)  
  
    reduce_on_plateau = ReduceLROnPlateau(monitor="val_acc", mode="max",  
                                         ↪factor=0.1, patience=3)  
  
    callbacks_list = [checkpoint, early, reduce_on_plateau] # early  
  
    model_gender = get_model(n_classes=2)  
    # `aug` determines whether or not to augment the images  
    model_gender.fit_generator(gen(tr_tr, aug=True),  
                               ↪validation_data=gen(tr_val), epochs=250, verbose=2, workers=1,
```

```

        callbacks=callbacks_list, steps_per_epoch=100, u
→validation_steps = 50,use_multiprocessing=False)

model_gender.load_weights(file_path)

predictions_gender += model_gender.predict(test_images)
cnt_ave += 1

# test_images = test_images[:, :, ::-1, :]

# predictions += model.predict(test_images)
# cnt_ave += 1

K.clear_session()

predictions_gender = predictions_gender/cnt_ave

predictions_gender = predictions_gender.argmax(axis=-1)

acc = accuracy_score(test_labels, predictions_gender)

print("accuracy : %s " %acc)

accuracies.append(acc)

```

0% | 0/3 [00:00<?, ?it/s]

Epoch 1/250
100/100 - 31s - loss: 1.0043 - acc: 0.6953 - val_loss: 0.6095 - val_acc: 0.7138

Epoch 00001: val_acc improved from -inf to 0.71375, saving model to
baseline_gender_0.h5

Epoch 2/250
100/100 - 22s - loss: 0.5137 - acc: 0.7738 - val_loss: 0.5902 - val_acc: 0.7234

Epoch 00002: val_acc improved from 0.71375 to 0.72340, saving model to
baseline_gender_0.h5

Epoch 3/250
100/100 - 22s - loss: 0.3695 - acc: 0.8594 - val_loss: 0.6552 - val_acc: 0.7081

Epoch 00003: val_acc did not improve from 0.72340

Epoch 4/250
100/100 - 22s - loss: 0.3300 - acc: 0.8775 - val_loss: 0.6260 - val_acc: 0.7190

Epoch 00004: val_acc did not improve from 0.72340

Epoch 5/250
100/100 - 22s - loss: 0.2658 - acc: 0.9013 - val_loss: 0.6929 - val_acc: 0.5282

```
Epoch 00005: val_acc did not improve from 0.72340
Epoch 6/250
100/100 - 21s - loss: 0.2302 - acc: 0.9166 - val_loss: 0.6454 - val_acc: 0.6662

Epoch 00006: val_acc did not improve from 0.72340
Epoch 7/250
100/100 - 21s - loss: 0.2095 - acc: 0.9249 - val_loss: 0.5009 - val_acc: 0.7509

Epoch 00007: val_acc improved from 0.72340 to 0.75094, saving model to
baseline_gender_0.h5
Epoch 8/250
100/100 - 21s - loss: 0.1770 - acc: 0.9350 - val_loss: 0.3989 - val_acc: 0.8019

Epoch 00008: val_acc improved from 0.75094 to 0.80187, saving model to
baseline_gender_0.h5
Epoch 9/250
100/100 - 21s - loss: 0.1748 - acc: 0.9353 - val_loss: 0.2438 - val_acc: 0.8955

Epoch 00009: val_acc improved from 0.80187 to 0.89549, saving model to
baseline_gender_0.h5
Epoch 10/250
100/100 - 21s - loss: 0.1683 - acc: 0.9400 - val_loss: 0.1562 - val_acc: 0.9324

Epoch 00010: val_acc improved from 0.89549 to 0.93242, saving model to
baseline_gender_0.h5
Epoch 11/250
100/100 - 22s - loss: 0.1644 - acc: 0.9375 - val_loss: 0.1735 - val_acc: 0.9306

Epoch 00011: val_acc did not improve from 0.93242
Epoch 12/250
100/100 - 21s - loss: 0.1495 - acc: 0.9478 - val_loss: 0.1413 - val_acc: 0.9437

Epoch 00012: val_acc improved from 0.93242 to 0.94368, saving model to
baseline_gender_0.h5
Epoch 13/250
100/100 - 21s - loss: 0.1605 - acc: 0.9428 - val_loss: 0.1230 - val_acc: 0.9544

Epoch 00013: val_acc improved from 0.94368 to 0.95438, saving model to
baseline_gender_0.h5
Epoch 14/250
100/100 - 21s - loss: 0.1550 - acc: 0.9468 - val_loss: 0.1509 - val_acc: 0.9456

Epoch 00014: val_acc did not improve from 0.95438
Epoch 15/250
100/100 - 21s - loss: 0.1624 - acc: 0.9394 - val_loss: 0.1131 - val_acc: 0.9549

Epoch 00015: val_acc improved from 0.95438 to 0.95494, saving model to
baseline_gender_0.h5
```

```
Epoch 16/250
100/100 - 21s - loss: 0.1321 - acc: 0.9544 - val_loss: 0.1360 - val_acc: 0.9588

Epoch 00016: val_acc improved from 0.95494 to 0.95875, saving model to
baseline_gender_0.h5
Epoch 17/250
100/100 - 22s - loss: 0.1464 - acc: 0.9491 - val_loss: 0.1288 - val_acc: 0.9531

Epoch 00017: val_acc did not improve from 0.95875
Epoch 18/250
100/100 - 21s - loss: 0.1328 - acc: 0.9572 - val_loss: 0.1063 - val_acc: 0.9575

Epoch 00018: val_acc did not improve from 0.95875
Epoch 19/250
100/100 - 21s - loss: 0.1404 - acc: 0.9513 - val_loss: 0.1449 - val_acc: 0.9506

Epoch 00019: val_acc did not improve from 0.95875
Epoch 20/250
100/100 - 21s - loss: 0.1327 - acc: 0.9531 - val_loss: 0.1067 - val_acc: 0.9574

Epoch 00020: val_acc did not improve from 0.95875
Epoch 21/250
100/100 - 21s - loss: 0.1235 - acc: 0.9556 - val_loss: 0.1054 - val_acc: 0.9650

Epoch 00021: val_acc improved from 0.95875 to 0.96500, saving model to
baseline_gender_0.h5
Epoch 22/250
100/100 - 21s - loss: 0.1271 - acc: 0.9550 - val_loss: 0.1187 - val_acc: 0.9574

Epoch 00022: val_acc did not improve from 0.96500
Epoch 23/250
100/100 - 22s - loss: 0.1164 - acc: 0.9591 - val_loss: 0.0901 - val_acc: 0.9663

Epoch 00023: val_acc improved from 0.96500 to 0.96625, saving model to
baseline_gender_0.h5
Epoch 24/250
100/100 - 21s - loss: 0.1126 - acc: 0.9625 - val_loss: 0.1218 - val_acc: 0.9599

Epoch 00024: val_acc did not improve from 0.96625
Epoch 25/250
100/100 - 21s - loss: 0.0979 - acc: 0.9672 - val_loss: 0.1033 - val_acc: 0.9612

Epoch 00025: val_acc did not improve from 0.96625
Epoch 26/250
100/100 - 22s - loss: 0.1135 - acc: 0.9616 - val_loss: 0.1010 - val_acc: 0.9675

Epoch 00026: val_acc improved from 0.96625 to 0.96750, saving model to
baseline_gender_0.h5
```

```
Epoch 27/250
100/100 - 22s - loss: 0.1167 - acc: 0.9584 - val_loss: 0.1194 - val_acc: 0.9593

Epoch 00027: val_acc did not improve from 0.96750
Epoch 28/250
100/100 - 21s - loss: 0.1076 - acc: 0.9584 - val_loss: 0.0874 - val_acc: 0.9669

Epoch 00028: val_acc did not improve from 0.96750
Epoch 29/250
100/100 - 22s - loss: 0.1158 - acc: 0.9591 - val_loss: 0.1173 - val_acc: 0.9650

Epoch 00029: val_acc did not improve from 0.96750
Epoch 30/250
100/100 - 22s - loss: 0.0967 - acc: 0.9659 - val_loss: 0.1027 - val_acc: 0.9637

Epoch 00030: val_acc did not improve from 0.96750
Epoch 31/250
100/100 - 22s - loss: 0.1155 - acc: 0.9609 - val_loss: 0.0936 - val_acc: 0.9712

Epoch 00031: val_acc improved from 0.96750 to 0.97125, saving model to
baseline_gender_0.h5
Epoch 32/250
100/100 - 22s - loss: 0.0939 - acc: 0.9684 - val_loss: 0.1221 - val_acc: 0.9625

Epoch 00032: val_acc did not improve from 0.97125
Epoch 33/250
100/100 - 21s - loss: 0.0996 - acc: 0.9641 - val_loss: 0.0909 - val_acc: 0.9687

Epoch 00033: val_acc did not improve from 0.97125
Epoch 34/250
100/100 - 22s - loss: 0.1170 - acc: 0.9556 - val_loss: 0.1148 - val_acc: 0.9650

Epoch 00034: val_acc did not improve from 0.97125
Epoch 35/250
100/100 - 21s - loss: 0.0986 - acc: 0.9609 - val_loss: 0.1090 - val_acc: 0.9631

Epoch 00035: val_acc did not improve from 0.97125
Epoch 36/250
100/100 - 21s - loss: 0.1152 - acc: 0.9638 - val_loss: 0.0971 - val_acc: 0.9681

Epoch 00036: val_acc did not improve from 0.97125
Epoch 37/250
100/100 - 22s - loss: 0.0991 - acc: 0.9656 - val_loss: 0.1244 - val_acc: 0.9599

Epoch 00037: val_acc did not improve from 0.97125
Epoch 38/250
100/100 - 21s - loss: 0.1047 - acc: 0.9616 - val_loss: 0.0853 - val_acc: 0.9706
```

```
Epoch 00038: val_acc did not improve from 0.97125
Epoch 39/250
100/100 - 21s - loss: 0.0920 - acc: 0.9709 - val_loss: 0.1123 - val_acc: 0.9669

Epoch 00039: val_acc did not improve from 0.97125
Epoch 40/250
100/100 - 21s - loss: 0.1096 - acc: 0.9675 - val_loss: 0.1086 - val_acc: 0.9637

Epoch 00040: val_acc did not improve from 0.97125
Epoch 41/250
100/100 - 21s - loss: 0.1111 - acc: 0.9596 - val_loss: 0.0960 - val_acc: 0.9688

Epoch 00041: val_acc did not improve from 0.97125
Epoch 42/250
100/100 - 21s - loss: 0.1087 - acc: 0.9625 - val_loss: 0.1334 - val_acc: 0.9587

Epoch 00042: val_acc did not improve from 0.97125
Epoch 43/250
100/100 - 21s - loss: 0.1067 - acc: 0.9631 - val_loss: 0.0865 - val_acc: 0.9687

Epoch 00043: val_acc did not improve from 0.97125
Epoch 44/250
100/100 - 21s - loss: 0.1042 - acc: 0.9647 - val_loss: 0.1112 - val_acc: 0.9675

Epoch 00044: val_acc did not improve from 0.97125
Epoch 45/250
100/100 - 21s - loss: 0.1106 - acc: 0.9650 - val_loss: 0.1145 - val_acc: 0.9650

Epoch 00045: val_acc did not improve from 0.97125
Epoch 46/250
100/100 - 21s - loss: 0.0973 - acc: 0.9703 - val_loss: 0.0848 - val_acc: 0.9706

Epoch 00046: val_acc did not improve from 0.97125
Epoch 47/250
100/100 - 21s - loss: 0.1157 - acc: 0.9591 - val_loss: 0.1315 - val_acc: 0.9587

Epoch 00047: val_acc did not improve from 0.97125
Epoch 48/250
100/100 - 21s - loss: 0.1105 - acc: 0.9596 - val_loss: 0.0930 - val_acc: 0.9650

Epoch 00048: val_acc did not improve from 0.97125
Epoch 49/250
100/100 - 21s - loss: 0.1090 - acc: 0.9653 - val_loss: 0.1012 - val_acc: 0.9706

Epoch 00049: val_acc did not improve from 0.97125
Epoch 50/250
100/100 - 21s - loss: 0.1009 - acc: 0.9638 - val_loss: 0.1168 - val_acc: 0.9631
```

```
Epoch 00050: val_acc did not improve from 0.97125
Epoch 51/250
100/100 - 21s - loss: 0.1010 - acc: 0.9681 - val_loss: 0.0888 - val_acc: 0.9700

Epoch 00051: val_acc did not improve from 0.97125
33%|           | 1/3 [18:48<37:37, 1128.75s/it]

Epoch 1/250
100/100 - 30s - loss: 1.1153 - acc: 0.6834 - val_loss: 77.0444 - val_acc: 0.7231

Epoch 00001: val_acc improved from -inf to 0.72312, saving model to
baseline_gender_1.h5
Epoch 2/250
100/100 - 22s - loss: 0.6045 - acc: 0.7266 - val_loss: 0.6136 - val_acc: 0.7265

Epoch 00002: val_acc improved from 0.72312 to 0.72653, saving model to
baseline_gender_1.h5
Epoch 3/250
100/100 - 22s - loss: 0.5007 - acc: 0.8025 - val_loss: 0.5942 - val_acc: 0.7306

Epoch 00003: val_acc improved from 0.72653 to 0.73062, saving model to
baseline_gender_1.h5
Epoch 4/250
100/100 - 22s - loss: 0.4311 - acc: 0.8456 - val_loss: 0.6159 - val_acc: 0.7128

Epoch 00004: val_acc did not improve from 0.73062
Epoch 5/250
100/100 - 22s - loss: 0.3772 - acc: 0.8550 - val_loss: 0.7441 - val_acc: 0.3411

Epoch 00005: val_acc did not improve from 0.73062
Epoch 6/250
100/100 - 22s - loss: 0.3901 - acc: 0.8453 - val_loss: 0.5922 - val_acc: 0.7331

Epoch 00006: val_acc improved from 0.73062 to 0.73312, saving model to
baseline_gender_1.h5
Epoch 7/250
100/100 - 22s - loss: 0.3271 - acc: 0.8755 - val_loss: 0.5966 - val_acc: 0.7203

Epoch 00007: val_acc did not improve from 0.73312
Epoch 8/250
100/100 - 22s - loss: 0.3163 - acc: 0.8772 - val_loss: 0.5418 - val_acc: 0.7337

Epoch 00008: val_acc improved from 0.73312 to 0.73375, saving model to
baseline_gender_1.h5
Epoch 9/250
100/100 - 22s - loss: 0.3394 - acc: 0.8741 - val_loss: 0.4335 - val_acc: 0.8198

Epoch 00009: val_acc improved from 0.73375 to 0.81977, saving model to
```

```
baseline_gender_1.h5
Epoch 10/250
100/100 - 22s - loss: 0.2817 - acc: 0.9034 - val_loss: 0.3504 - val_acc: 0.8554

Epoch 00010: val_acc improved from 0.81977 to 0.85544, saving model to
baseline_gender_1.h5
Epoch 11/250
100/100 - 22s - loss: 0.2619 - acc: 0.9009 - val_loss: 0.2181 - val_acc: 0.9112

Epoch 00011: val_acc improved from 0.85544 to 0.91125, saving model to
baseline_gender_1.h5
Epoch 12/250
100/100 - 22s - loss: 0.2480 - acc: 0.9106 - val_loss: 0.3889 - val_acc: 0.8473

Epoch 00012: val_acc did not improve from 0.91125
Epoch 13/250
100/100 - 22s - loss: 0.2387 - acc: 0.9134 - val_loss: 0.2013 - val_acc: 0.9219

Epoch 00013: val_acc improved from 0.91125 to 0.92188, saving model to
baseline_gender_1.h5
Epoch 14/250
100/100 - 22s - loss: 0.2338 - acc: 0.9177 - val_loss: 0.1810 - val_acc: 0.9305

Epoch 00014: val_acc improved from 0.92188 to 0.93054, saving model to
baseline_gender_1.h5
Epoch 15/250
100/100 - 22s - loss: 0.2231 - acc: 0.9150 - val_loss: 0.2292 - val_acc: 0.9161

Epoch 00015: val_acc did not improve from 0.93054
Epoch 16/250
100/100 - 22s - loss: 0.2248 - acc: 0.9178 - val_loss: 0.3983 - val_acc: 0.8944

Epoch 00016: val_acc did not improve from 0.93054
Epoch 17/250
100/100 - 21s - loss: 0.2030 - acc: 0.9272 - val_loss: 0.1752 - val_acc: 0.9324

Epoch 00017: val_acc improved from 0.93054 to 0.93242, saving model to
baseline_gender_1.h5
Epoch 18/250
100/100 - 22s - loss: 0.2002 - acc: 0.9287 - val_loss: 0.1788 - val_acc: 0.9275

Epoch 00018: val_acc did not improve from 0.93242
Epoch 19/250
100/100 - 22s - loss: 0.2152 - acc: 0.9237 - val_loss: 0.2862 - val_acc: 0.8917

Epoch 00019: val_acc did not improve from 0.93242
Epoch 20/250
100/100 - 22s - loss: 0.1937 - acc: 0.9337 - val_loss: 0.2953 - val_acc: 0.8967
```

```
Epoch 00020: val_acc did not improve from 0.93242
Epoch 21/250
100/100 - 22s - loss: 0.1847 - acc: 0.9337 - val_loss: 0.1276 - val_acc: 0.9475

Epoch 00021: val_acc improved from 0.93242 to 0.94750, saving model to
baseline_gender_1.h5
Epoch 22/250
100/100 - 22s - loss: 0.1841 - acc: 0.9391 - val_loss: 0.1240 - val_acc: 0.9574

Epoch 00022: val_acc improved from 0.94750 to 0.95745, saving model to
baseline_gender_1.h5
Epoch 23/250
100/100 - 22s - loss: 0.1370 - acc: 0.9513 - val_loss: 0.1026 - val_acc: 0.9613

Epoch 00023: val_acc improved from 0.95745 to 0.96125, saving model to
baseline_gender_1.h5
Epoch 24/250
100/100 - 22s - loss: 0.1311 - acc: 0.9559 - val_loss: 0.1174 - val_acc: 0.9531

Epoch 00024: val_acc did not improve from 0.96125
Epoch 25/250
100/100 - 22s - loss: 0.1443 - acc: 0.9550 - val_loss: 0.1199 - val_acc: 0.9631

Epoch 00025: val_acc improved from 0.96125 to 0.96308, saving model to
baseline_gender_1.h5
Epoch 26/250
100/100 - 22s - loss: 0.1312 - acc: 0.9559 - val_loss: 0.0967 - val_acc: 0.9563

Epoch 00026: val_acc did not improve from 0.96308
Epoch 27/250
100/100 - 22s - loss: 0.1212 - acc: 0.9547 - val_loss: 0.1081 - val_acc: 0.9587

Epoch 00027: val_acc did not improve from 0.96308
Epoch 28/250
100/100 - 22s - loss: 0.1421 - acc: 0.9499 - val_loss: 0.1010 - val_acc: 0.9644

Epoch 00028: val_acc improved from 0.96308 to 0.96438, saving model to
baseline_gender_1.h5
Epoch 29/250
100/100 - 22s - loss: 0.1372 - acc: 0.9522 - val_loss: 0.1026 - val_acc: 0.9575

Epoch 00029: val_acc did not improve from 0.96438
Epoch 30/250
100/100 - 22s - loss: 0.1189 - acc: 0.9584 - val_loss: 0.0948 - val_acc: 0.9643

Epoch 00030: val_acc did not improve from 0.96438
Epoch 31/250
```

```
100/100 - 22s - loss: 0.1197 - acc: 0.9566 - val_loss: 0.0866 - val_acc: 0.9669

Epoch 00031: val_acc improved from 0.96438 to 0.96688, saving model to
baseline_gender_1.h5
Epoch 32/250
100/100 - 22s - loss: 0.1218 - acc: 0.9541 - val_loss: 0.1078 - val_acc: 0.9650

Epoch 00032: val_acc did not improve from 0.96688
Epoch 33/250
100/100 - 22s - loss: 0.1062 - acc: 0.9659 - val_loss: 0.1082 - val_acc: 0.9612

Epoch 00033: val_acc did not improve from 0.96688
Epoch 34/250
100/100 - 22s - loss: 0.1078 - acc: 0.9650 - val_loss: 0.0931 - val_acc: 0.9613

Epoch 00034: val_acc did not improve from 0.96688
Epoch 35/250
100/100 - 22s - loss: 0.1432 - acc: 0.9488 - val_loss: 0.0963 - val_acc: 0.9643

Epoch 00035: val_acc did not improve from 0.96688
Epoch 36/250
100/100 - 22s - loss: 0.1129 - acc: 0.9628 - val_loss: 0.0859 - val_acc: 0.9638

Epoch 00036: val_acc did not improve from 0.96688
Epoch 37/250
100/100 - 22s - loss: 0.1126 - acc: 0.9625 - val_loss: 0.1005 - val_acc: 0.9656

Epoch 00037: val_acc did not improve from 0.96688
Epoch 38/250
100/100 - 22s - loss: 0.1040 - acc: 0.9641 - val_loss: 0.0950 - val_acc: 0.9662

Epoch 00038: val_acc did not improve from 0.96688
Epoch 39/250
100/100 - 22s - loss: 0.1059 - acc: 0.9653 - val_loss: 0.0782 - val_acc: 0.9675

Epoch 00039: val_acc improved from 0.96688 to 0.96750, saving model to
baseline_gender_1.h5
Epoch 40/250
100/100 - 22s - loss: 0.0918 - acc: 0.9681 - val_loss: 0.0957 - val_acc: 0.9650

Epoch 00040: val_acc did not improve from 0.96750
Epoch 41/250
100/100 - 22s - loss: 0.1081 - acc: 0.9653 - val_loss: 0.0875 - val_acc: 0.9644

Epoch 00041: val_acc did not improve from 0.96750
Epoch 42/250
100/100 - 22s - loss: 0.1277 - acc: 0.9575 - val_loss: 0.0936 - val_acc: 0.9662
```

```
Epoch 00042: val_acc did not improve from 0.96750
Epoch 43/250
100/100 - 22s - loss: 0.1107 - acc: 0.9609 - val_loss: 0.0987 - val_acc: 0.9656

Epoch 00043: val_acc did not improve from 0.96750
Epoch 44/250
100/100 - 22s - loss: 0.1109 - acc: 0.9628 - val_loss: 0.0827 - val_acc: 0.9663

Epoch 00044: val_acc did not improve from 0.96750
Epoch 45/250
100/100 - 22s - loss: 0.0988 - acc: 0.9666 - val_loss: 0.0945 - val_acc: 0.9662

Epoch 00045: val_acc did not improve from 0.96750
Epoch 46/250
100/100 - 22s - loss: 0.0938 - acc: 0.9653 - val_loss: 0.0869 - val_acc: 0.9650

Epoch 00046: val_acc did not improve from 0.96750
Epoch 47/250
100/100 - 22s - loss: 0.0946 - acc: 0.9716 - val_loss: 0.0970 - val_acc: 0.9643

Epoch 00047: val_acc did not improve from 0.96750
Epoch 48/250
100/100 - 22s - loss: 0.1129 - acc: 0.9615 - val_loss: 0.0928 - val_acc: 0.9681

Epoch 00048: val_acc improved from 0.96750 to 0.96809, saving model to
baseline_gender_1.h5
Epoch 49/250
100/100 - 22s - loss: 0.1226 - acc: 0.9584 - val_loss: 0.0811 - val_acc: 0.9669

Epoch 00049: val_acc did not improve from 0.96809
Epoch 50/250
100/100 - 22s - loss: 0.1124 - acc: 0.9641 - val_loss: 0.0949 - val_acc: 0.9656

Epoch 00050: val_acc did not improve from 0.96809
Epoch 51/250
100/100 - 22s - loss: 0.0939 - acc: 0.9663 - val_loss: 0.0850 - val_acc: 0.9663

Epoch 00051: val_acc did not improve from 0.96809
Epoch 52/250
100/100 - 22s - loss: 0.1084 - acc: 0.9644 - val_loss: 0.0918 - val_acc: 0.9650

Epoch 00052: val_acc did not improve from 0.96809
Epoch 53/250
100/100 - 22s - loss: 0.1099 - acc: 0.9638 - val_loss: 0.0925 - val_acc: 0.9662

Epoch 00053: val_acc did not improve from 0.96809
Epoch 54/250
100/100 - 22s - loss: 0.0950 - acc: 0.9716 - val_loss: 0.0818 - val_acc: 0.9650
```

```
Epoch 00054: val_acc did not improve from 0.96809
Epoch 55/250
100/100 - 22s - loss: 0.1184 - acc: 0.9568 - val_loss: 0.0986 - val_acc: 0.9650

Epoch 00055: val_acc did not improve from 0.96809
Epoch 56/250
100/100 - 22s - loss: 0.1243 - acc: 0.9638 - val_loss: 0.0910 - val_acc: 0.9669

Epoch 00056: val_acc did not improve from 0.96809
Epoch 57/250
100/100 - 22s - loss: 0.1024 - acc: 0.9653 - val_loss: 0.0918 - val_acc: 0.9650

Epoch 00057: val_acc did not improve from 0.96809
Epoch 58/250
100/100 - 22s - loss: 0.0960 - acc: 0.9672 - val_loss: 0.0932 - val_acc: 0.9675

Epoch 00058: val_acc did not improve from 0.96809
Epoch 59/250
100/100 - 22s - loss: 0.1122 - acc: 0.9641 - val_loss: 0.0779 - val_acc: 0.9675

Epoch 00059: val_acc did not improve from 0.96809
Epoch 60/250
100/100 - 22s - loss: 0.0994 - acc: 0.9656 - val_loss: 0.1025 - val_acc: 0.9643

Epoch 00060: val_acc did not improve from 0.96809
Epoch 61/250
100/100 - 22s - loss: 0.0904 - acc: 0.9663 - val_loss: 0.0955 - val_acc: 0.9656

Epoch 00061: val_acc did not improve from 0.96809
Epoch 62/250
100/100 - 22s - loss: 0.1288 - acc: 0.9556 - val_loss: 0.0821 - val_acc: 0.9669

Epoch 00062: val_acc did not improve from 0.96809
Epoch 63/250
100/100 - 22s - loss: 0.1045 - acc: 0.9625 - val_loss: 0.0960 - val_acc: 0.9650

Epoch 00063: val_acc did not improve from 0.96809
Epoch 64/250
100/100 - 22s - loss: 0.1043 - acc: 0.9631 - val_loss: 0.0849 - val_acc: 0.9638

Epoch 00064: val_acc did not improve from 0.96809
Epoch 65/250
100/100 - 22s - loss: 0.0947 - acc: 0.9641 - val_loss: 0.1004 - val_acc: 0.9643

Epoch 00065: val_acc did not improve from 0.96809
Epoch 66/250
100/100 - 22s - loss: 0.1041 - acc: 0.9653 - val_loss: 0.0948 - val_acc: 0.9656
```

```
Epoch 00066: val_acc did not improve from 0.96809
Epoch 67/250
100/100 - 22s - loss: 0.0894 - acc: 0.9716 - val_loss: 0.0779 - val_acc: 0.9675

Epoch 00067: val_acc did not improve from 0.96809
Epoch 68/250
100/100 - 22s - loss: 0.0987 - acc: 0.9696 - val_loss: 0.0956 - val_acc: 0.9656

Epoch 00068: val_acc did not improve from 0.96809
  67%|          | 2/3 [43:55<22:31, 1351.38s/it]

Epoch 1/250
100/100 - 30s - loss: 0.9647 - acc: 0.6675 - val_loss: 364.6895 - val_acc:
0.7156

Epoch 00001: val_acc improved from -inf to 0.71562, saving model to
baseline_gender_2.h5
Epoch 2/250
100/100 - 23s - loss: 0.6139 - acc: 0.7403 - val_loss: 0.6077 - val_acc: 0.7159

Epoch 00002: val_acc improved from 0.71562 to 0.71589, saving model to
baseline_gender_2.h5
Epoch 3/250
100/100 - 22s - loss: 0.4532 - acc: 0.8166 - val_loss: 0.6077 - val_acc: 0.7144

Epoch 00003: val_acc did not improve from 0.71589
Epoch 4/250
100/100 - 22s - loss: 0.3550 - acc: 0.8556 - val_loss: 0.5937 - val_acc: 0.7190

Epoch 00004: val_acc improved from 0.71589 to 0.71902, saving model to
baseline_gender_2.h5
Epoch 5/250
100/100 - 22s - loss: 0.3145 - acc: 0.8875 - val_loss: 0.5807 - val_acc: 0.7284

Epoch 00005: val_acc improved from 0.71902 to 0.72841, saving model to
baseline_gender_2.h5
Epoch 6/250
100/100 - 22s - loss: 0.2560 - acc: 0.9072 - val_loss: 0.6051 - val_acc: 0.7125

Epoch 00006: val_acc did not improve from 0.72841
Epoch 7/250
100/100 - 22s - loss: 0.2616 - acc: 0.8961 - val_loss: 0.6070 - val_acc: 0.7171

Epoch 00007: val_acc did not improve from 0.72841
Epoch 8/250
100/100 - 22s - loss: 0.2489 - acc: 0.9075 - val_loss: 0.5046 - val_acc: 0.7812
```

```
Epoch 00008: val_acc improved from 0.72841 to 0.78125, saving model to
baseline_gender_2.h5
Epoch 9/250
100/100 - 22s - loss: 0.2701 - acc: 0.9003 - val_loss: 1.2305 - val_acc: 0.8930

Epoch 00009: val_acc improved from 0.78125 to 0.89299, saving model to
baseline_gender_2.h5
Epoch 10/250
100/100 - 22s - loss: 0.2360 - acc: 0.9159 - val_loss: 0.3525 - val_acc: 0.8705

Epoch 00010: val_acc did not improve from 0.89299
Epoch 11/250
100/100 - 22s - loss: 0.2040 - acc: 0.9253 - val_loss: 0.3741 - val_acc: 0.8700

Epoch 00011: val_acc did not improve from 0.89299
Epoch 12/250
100/100 - 22s - loss: 0.2331 - acc: 0.9200 - val_loss: 0.6583 - val_acc: 0.8304

Epoch 00012: val_acc did not improve from 0.89299
Epoch 13/250
100/100 - 22s - loss: 0.1979 - acc: 0.9372 - val_loss: 0.2142 - val_acc: 0.9312

Epoch 00013: val_acc improved from 0.89299 to 0.93125, saving model to
baseline_gender_2.h5
Epoch 14/250
100/100 - 22s - loss: 0.1722 - acc: 0.9371 - val_loss: 0.1601 - val_acc: 0.9406

Epoch 00014: val_acc improved from 0.93125 to 0.94055, saving model to
baseline_gender_2.h5
Epoch 15/250
100/100 - 22s - loss: 0.1614 - acc: 0.9444 - val_loss: 0.1438 - val_acc: 0.9431

Epoch 00015: val_acc improved from 0.94055 to 0.94305, saving model to
baseline_gender_2.h5
Epoch 16/250
100/100 - 22s - loss: 0.1691 - acc: 0.9416 - val_loss: 0.1471 - val_acc: 0.9488

Epoch 00016: val_acc improved from 0.94305 to 0.94875, saving model to
baseline_gender_2.h5
Epoch 17/250
100/100 - 22s - loss: 0.1624 - acc: 0.9441 - val_loss: 0.1309 - val_acc: 0.9468

Epoch 00017: val_acc did not improve from 0.94875
Epoch 18/250
100/100 - 22s - loss: 0.1373 - acc: 0.9550 - val_loss: 0.1549 - val_acc: 0.9538

Epoch 00018: val_acc improved from 0.94875 to 0.95375, saving model to
baseline_gender_2.h5
```

```
Epoch 19/250
100/100 - 22s - loss: 0.1317 - acc: 0.9519 - val_loss: 0.1471 - val_acc: 0.9487

Epoch 00019: val_acc did not improve from 0.95375
Epoch 20/250
100/100 - 22s - loss: 0.1324 - acc: 0.9591 - val_loss: 0.1253 - val_acc: 0.9518

Epoch 00020: val_acc did not improve from 0.95375
Epoch 21/250
100/100 - 22s - loss: 0.1451 - acc: 0.9468 - val_loss: 0.1551 - val_acc: 0.9481

Epoch 00021: val_acc did not improve from 0.95375
Epoch 22/250
100/100 - 22s - loss: 0.1541 - acc: 0.9519 - val_loss: 0.1261 - val_acc: 0.9518

Epoch 00022: val_acc did not improve from 0.95375
Epoch 23/250
100/100 - 22s - loss: 0.1273 - acc: 0.9509 - val_loss: 0.1451 - val_acc: 0.9556

Epoch 00023: val_acc improved from 0.95375 to 0.95562, saving model to
baseline_gender_2.h5
Epoch 24/250
100/100 - 22s - loss: 0.1291 - acc: 0.9528 - val_loss: 0.1387 - val_acc: 0.9524

Epoch 00024: val_acc did not improve from 0.95562
Epoch 25/250
100/100 - 22s - loss: 0.1183 - acc: 0.9550 - val_loss: 0.1181 - val_acc: 0.9531

Epoch 00025: val_acc did not improve from 0.95562
Epoch 26/250
100/100 - 22s - loss: 0.1307 - acc: 0.9531 - val_loss: 0.1379 - val_acc: 0.9563

Epoch 00026: val_acc improved from 0.95562 to 0.95625, saving model to
baseline_gender_2.h5
Epoch 27/250
100/100 - 22s - loss: 0.1143 - acc: 0.9597 - val_loss: 0.1185 - val_acc: 0.9537

Epoch 00027: val_acc did not improve from 0.95625
Epoch 28/250
100/100 - 22s - loss: 0.1284 - acc: 0.9528 - val_loss: 0.1442 - val_acc: 0.9550

Epoch 00028: val_acc did not improve from 0.95625
Epoch 29/250
100/100 - 22s - loss: 0.1375 - acc: 0.9506 - val_loss: 0.1393 - val_acc: 0.9513

Epoch 00029: val_acc did not improve from 0.95625
Epoch 30/250
100/100 - 22s - loss: 0.1267 - acc: 0.9550 - val_loss: 0.1147 - val_acc: 0.9543
```

```
Epoch 00030: val_acc did not improve from 0.95625
Epoch 31/250
100/100 - 22s - loss: 0.1219 - acc: 0.9584 - val_loss: 0.1375 - val_acc: 0.9563

Epoch 00031: val_acc did not improve from 0.95625
Epoch 32/250
100/100 - 22s - loss: 0.1135 - acc: 0.9603 - val_loss: 0.1159 - val_acc: 0.9543

Epoch 00032: val_acc did not improve from 0.95625
Epoch 33/250
100/100 - 22s - loss: 0.1234 - acc: 0.9581 - val_loss: 0.1419 - val_acc: 0.9518

Epoch 00033: val_acc did not improve from 0.95625
Epoch 34/250
100/100 - 22s - loss: 0.1196 - acc: 0.9599 - val_loss: 0.1410 - val_acc: 0.9544

Epoch 00034: val_acc did not improve from 0.95625
Epoch 35/250
100/100 - 22s - loss: 0.1233 - acc: 0.9566 - val_loss: 0.1099 - val_acc: 0.9543

Epoch 00035: val_acc did not improve from 0.95625
Epoch 36/250
100/100 - 22s - loss: 0.1433 - acc: 0.9525 - val_loss: 0.1464 - val_acc: 0.9550

Epoch 00036: val_acc did not improve from 0.95625
Epoch 37/250
100/100 - 22s - loss: 0.1357 - acc: 0.9503 - val_loss: 0.1157 - val_acc: 0.9568

Epoch 00037: val_acc improved from 0.95625 to 0.95682, saving model to
baseline_gender_2.h5
Epoch 38/250
100/100 - 22s - loss: 0.1207 - acc: 0.9569 - val_loss: 0.1268 - val_acc: 0.9531

Epoch 00038: val_acc did not improve from 0.95682
Epoch 39/250
100/100 - 22s - loss: 0.1094 - acc: 0.9613 - val_loss: 0.1374 - val_acc: 0.9556

Epoch 00039: val_acc did not improve from 0.95682
Epoch 40/250
100/100 - 22s - loss: 0.1263 - acc: 0.9616 - val_loss: 0.1144 - val_acc: 0.9512

Epoch 00040: val_acc did not improve from 0.95682
Epoch 41/250
100/100 - 22s - loss: 0.1256 - acc: 0.9574 - val_loss: 0.1469 - val_acc: 0.9544

Epoch 00041: val_acc did not improve from 0.95682
Epoch 42/250
```

```
100/100 - 22s - loss: 0.1152 - acc: 0.9547 - val_loss: 0.1296 - val_acc: 0.9543

Epoch 00042: val_acc did not improve from 0.95682
Epoch 43/250
100/100 - 22s - loss: 0.1257 - acc: 0.9556 - val_loss: 0.1226 - val_acc: 0.9531

Epoch 00043: val_acc did not improve from 0.95682
Epoch 44/250
100/100 - 22s - loss: 0.1315 - acc: 0.9547 - val_loss: 0.1376 - val_acc: 0.9550

Epoch 00044: val_acc did not improve from 0.95682
Epoch 45/250
100/100 - 22s - loss: 0.1064 - acc: 0.9609 - val_loss: 0.1125 - val_acc: 0.9556

Epoch 00045: val_acc did not improve from 0.95682
Epoch 46/250
100/100 - 22s - loss: 0.1215 - acc: 0.9556 - val_loss: 0.1426 - val_acc: 0.9550

Epoch 00046: val_acc did not improve from 0.95682
Epoch 47/250
100/100 - 22s - loss: 0.1173 - acc: 0.9619 - val_loss: 0.1345 - val_acc: 0.9518

Epoch 00047: val_acc did not improve from 0.95682
Epoch 48/250
100/100 - 22s - loss: 0.1202 - acc: 0.9565 - val_loss: 0.1192 - val_acc: 0.9543

Epoch 00048: val_acc did not improve from 0.95682
Epoch 49/250
100/100 - 22s - loss: 0.1218 - acc: 0.9591 - val_loss: 0.1363 - val_acc: 0.9556

Epoch 00049: val_acc did not improve from 0.95682
Epoch 50/250
100/100 - 22s - loss: 0.1291 - acc: 0.9563 - val_loss: 0.1227 - val_acc: 0.9512

Epoch 00050: val_acc did not improve from 0.95682
Epoch 51/250
100/100 - 22s - loss: 0.1260 - acc: 0.9544 - val_loss: 0.1411 - val_acc: 0.9569

Epoch 00051: val_acc improved from 0.95682 to 0.95688, saving model to
baseline_gender_2.h5
Epoch 52/250
100/100 - 22s - loss: 0.1217 - acc: 0.9572 - val_loss: 0.1357 - val_acc: 0.9531

Epoch 00052: val_acc did not improve from 0.95688
Epoch 53/250
100/100 - 22s - loss: 0.1164 - acc: 0.9600 - val_loss: 0.1148 - val_acc: 0.9537

Epoch 00053: val_acc did not improve from 0.95688
```

```
Epoch 54/250
100/100 - 22s - loss: 0.1170 - acc: 0.9588 - val_loss: 0.1357 - val_acc: 0.9563

Epoch 00054: val_acc did not improve from 0.95688
Epoch 55/250
100/100 - 22s - loss: 0.1237 - acc: 0.9587 - val_loss: 0.1180 - val_acc: 0.9537

Epoch 00055: val_acc did not improve from 0.95688
Epoch 56/250
100/100 - 22s - loss: 0.1367 - acc: 0.9519 - val_loss: 0.1420 - val_acc: 0.9538

Epoch 00056: val_acc did not improve from 0.95688
Epoch 57/250
100/100 - 22s - loss: 0.1194 - acc: 0.9566 - val_loss: 0.1399 - val_acc: 0.9513

Epoch 00057: val_acc did not improve from 0.95688
Epoch 58/250
100/100 - 22s - loss: 0.1280 - acc: 0.9575 - val_loss: 0.1145 - val_acc: 0.9543

Epoch 00058: val_acc did not improve from 0.95688
Epoch 59/250
100/100 - 22s - loss: 0.1154 - acc: 0.9556 - val_loss: 0.1379 - val_acc: 0.9563

Epoch 00059: val_acc did not improve from 0.95688
Epoch 60/250
100/100 - 22s - loss: 0.1217 - acc: 0.9541 - val_loss: 0.1158 - val_acc: 0.9549

Epoch 00060: val_acc did not improve from 0.95688
Epoch 61/250
100/100 - 22s - loss: 0.1172 - acc: 0.9597 - val_loss: 0.1417 - val_acc: 0.9518

Epoch 00061: val_acc did not improve from 0.95688
Epoch 62/250
100/100 - 22s - loss: 0.1200 - acc: 0.9556 - val_loss: 0.1409 - val_acc: 0.9538

Epoch 00062: val_acc did not improve from 0.95688
Epoch 63/250
100/100 - 22s - loss: 0.1317 - acc: 0.9569 - val_loss: 0.1101 - val_acc: 0.9549

Epoch 00063: val_acc did not improve from 0.95688
Epoch 64/250
100/100 - 22s - loss: 0.1320 - acc: 0.9572 - val_loss: 0.1465 - val_acc: 0.9550

Epoch 00064: val_acc did not improve from 0.95688
Epoch 65/250
100/100 - 22s - loss: 0.1194 - acc: 0.9584 - val_loss: 0.1154 - val_acc: 0.9568

Epoch 00065: val_acc did not improve from 0.95688
```

```

Epoch 66/250
100/100 - 22s - loss: 0.1149 - acc: 0.9572 - val_loss: 0.1270 - val_acc: 0.9531

Epoch 00066: val_acc did not improve from 0.95688
Epoch 67/250
100/100 - 22s - loss: 0.1201 - acc: 0.9628 - val_loss: 0.1374 - val_acc: 0.9556

Epoch 00067: val_acc did not improve from 0.95688
Epoch 68/250
100/100 - 22s - loss: 0.1123 - acc: 0.9581 - val_loss: 0.1143 - val_acc: 0.9512

Epoch 00068: val_acc did not improve from 0.95688
Epoch 69/250
100/100 - 22s - loss: 0.1263 - acc: 0.9544 - val_loss: 0.1460 - val_acc: 0.9544

Epoch 00069: val_acc did not improve from 0.95688
Epoch 70/250
100/100 - 22s - loss: 0.1339 - acc: 0.9538 - val_loss: 0.1298 - val_acc: 0.9543

Epoch 00070: val_acc did not improve from 0.95688
Epoch 71/250
100/100 - 22s - loss: 0.1392 - acc: 0.9525 - val_loss: 0.1225 - val_acc: 0.9543

Epoch 00071: val_acc did not improve from 0.95688
100%|     | 3/3 [1:10:09<00:00, 1403.24s/it]
accuracy : 0.9660892001474383

```

2.3 Number of trainable parameters

[42]: model_gender.summary()

```

Model: "base_model"
-----
Layer (type)          Output Shape         Param #  Connected to
=====
input_4 (InputLayer)   [(None, None, None, 0
-----
conv1_pad (ZeroPadding2D)    (None, None, None, 3 0      input_4[0] [0]
-----
conv1_conv (Conv2D)        (None, None, None, 6 9472      conv1_pad[0] [0]
-----
```

```
-----  
conv1_bn (BatchNormalization) (None, None, None, 6 256  
conv1_conv[0] [0]  
-----  
conv1_relu (Activation) (None, None, None, 6 0 conv1_bn[0] [0]  
-----  
pool1_pad (ZeroPadding2D) (None, None, None, 6 0  
conv1_relu[0] [0]  
-----  
pool1_pool (MaxPooling2D) (None, None, None, 6 0 pool1_pad[0] [0]  
-----  
conv2_block1_1_conv (Conv2D) (None, None, None, 6 4160  
pool1_pool[0] [0]  
-----  
conv2_block1_1_bn (BatchNormali (None, None, None, 6 256  
conv2_block1_1_conv[0] [0]  
-----  
conv2_block1_1_relu (Activation (None, None, None, 6 0  
conv2_block1_1_bn[0] [0]  
-----  
conv2_block1_2_conv (Conv2D) (None, None, None, 6 36928  
conv2_block1_1_relu[0] [0]  
-----  
conv2_block1_2_bn (BatchNormali (None, None, None, 6 256  
conv2_block1_2_conv[0] [0]  
-----  
conv2_block1_2_relu (Activation (None, None, None, 6 0  
conv2_block1_2_bn[0] [0]  
-----  
conv2_block1_0_conv (Conv2D) (None, None, None, 2 16640  
pool1_pool[0] [0]  
-----  
conv2_block1_3_conv (Conv2D) (None, None, None, 2 16640  
conv2_block1_2_relu[0] [0]  
-----  
conv2_block1_0_bn (BatchNormali (None, None, None, 2 1024
```

```
conv2_block1_0_conv[0] [0]
-----
conv2_block1_3_bn (BatchNormali (None, None, None, 2 1024
conv2_block1_3_conv[0] [0]
-----
conv2_block1_add (Add)          (None, None, None, 2 0
conv2_block1_0_bn[0] [0]
conv2_block1_3_bn[0] [0]
-----
conv2_block1_out (Activation)  (None, None, None, 2 0
conv2_block1_add[0] [0]
-----
conv2_block2_1_conv (Conv2D)    (None, None, None, 6 16448
conv2_block1_out[0] [0]
-----
conv2_block2_1_bn (BatchNormali (None, None, None, 6 256
conv2_block2_1_conv[0] [0]
-----
conv2_block2_1_relu (Activation (None, None, None, 6 0
conv2_block2_1_bn[0] [0]
-----
conv2_block2_2_conv (Conv2D)    (None, None, None, 6 36928
conv2_block2_1_relu[0] [0]
-----
conv2_block2_2_bn (BatchNormali (None, None, None, 6 256
conv2_block2_2_conv[0] [0]
-----
conv2_block2_2_relu (Activation (None, None, None, 6 0
conv2_block2_2_bn[0] [0]
-----
conv2_block2_3_conv (Conv2D)    (None, None, None, 2 16640
conv2_block2_2_relu[0] [0]
-----
conv2_block2_3_bn (BatchNormali (None, None, None, 2 1024
conv2_block2_3_conv[0] [0]
```

```
conv2_block2_add (Add)           (None, None, None, 2 0
conv2_block1_out[0] [0]
conv2_block2_3_bn[0] [0]

-----
conv2_block2_out (Activation)   (None, None, None, 2 0
conv2_block2_add[0] [0]

-----
conv2_block3_1_conv (Conv2D)    (None, None, None, 6 16448
conv2_block2_out[0] [0]

-----
conv2_block3_1_bn (BatchNormali (None, None, None, 6 256
conv2_block3_1_conv[0] [0]

-----
conv2_block3_1_relu (Activation (None, None, None, 6 0
conv2_block3_1_bn[0] [0]

-----
conv2_block3_2_conv (Conv2D)    (None, None, None, 6 36928
conv2_block3_1_relu[0] [0]

-----
conv2_block3_2_bn (BatchNormali (None, None, None, 6 256
conv2_block3_2_conv[0] [0]

-----
conv2_block3_2_relu (Activation (None, None, None, 6 0
conv2_block3_2_bn[0] [0]

-----
conv2_block3_3_conv (Conv2D)    (None, None, None, 2 16640
conv2_block3_2_relu[0] [0]

-----
conv2_block3_3_bn (BatchNormali (None, None, None, 2 1024
conv2_block3_3_conv[0] [0]

-----
conv2_block3_add (Add)          (None, None, None, 2 0
conv2_block2_out[0] [0]
conv2_block3_3_bn[0] [0]

-----
conv2_block3_out (Activation)   (None, None, None, 2 0
conv2_block3_add[0] [0]
```

```
-----  
conv3_block1_1_conv (Conv2D)      (None, None, None, 1 32896  
conv2_block3_out[0] [0]  
  
-----  
conv3_block1_1_bn (BatchNormali (None, None, None, 1 512  
conv3_block1_1_conv[0] [0]  
  
-----  
conv3_block1_1_relu (Activation (None, None, None, 1 0  
conv3_block1_1_bn[0] [0]  
  
-----  
conv3_block1_2_conv (Conv2D)      (None, None, None, 1 147584  
conv3_block1_1_relu[0] [0]  
  
-----  
conv3_block1_2_bn (BatchNormali (None, None, None, 1 512  
conv3_block1_2_conv[0] [0]  
  
-----  
conv3_block1_2_relu (Activation (None, None, None, 1 0  
conv3_block1_2_bn[0] [0]  
  
-----  
conv3_block1_0_conv (Conv2D)      (None, None, None, 5 131584  
conv2_block3_out[0] [0]  
  
-----  
conv3_block1_3_conv (Conv2D)      (None, None, None, 5 66048  
conv3_block1_2_relu[0] [0]  
  
-----  
conv3_block1_0_bn (BatchNormali (None, None, None, 5 2048  
conv3_block1_0_conv[0] [0]  
  
-----  
conv3_block1_3_bn (BatchNormali (None, None, None, 5 2048  
conv3_block1_3_conv[0] [0]  
  
-----  
conv3_block1_add (Add)           (None, None, None, 5 0  
conv3_block1_0_bn[0] [0]  
conv3_block1_3_bn[0] [0]  
  
-----  
conv3_block1_out (Activation)    (None, None, None, 5 0
```

```
conv3_block1_add[0] [0]
-----
conv3_block2_1_conv (Conv2D)      (None, None, None, 1 65664
conv3_block1_out[0] [0]
-----
conv3_block2_1_bn (BatchNormali (None, None, None, 1 512
conv3_block2_1_conv[0] [0]
-----
conv3_block2_1_relu (Activation (None, None, None, 1 0
conv3_block2_1_bn[0] [0]
-----
conv3_block2_2_conv (Conv2D)      (None, None, None, 1 147584
conv3_block2_1_relu[0] [0]
-----
conv3_block2_2_bn (BatchNormali (None, None, None, 1 512
conv3_block2_2_conv[0] [0]
-----
conv3_block2_2_relu (Activation (None, None, None, 1 0
conv3_block2_2_bn[0] [0]
-----
conv3_block2_3_conv (Conv2D)      (None, None, None, 5 66048
conv3_block2_2_relu[0] [0]
-----
conv3_block2_3_bn (BatchNormali (None, None, None, 5 2048
conv3_block2_3_conv[0] [0]
-----
conv3_block2_add (Add)           (None, None, None, 5 0
conv3_block1_out[0] [0]
conv3_block2_3_bn[0] [0]
-----
conv3_block2_out (Activation)    (None, None, None, 5 0
conv3_block2_add[0] [0]
-----
conv3_block3_1_conv (Conv2D)      (None, None, None, 1 65664
conv3_block2_out[0] [0]
```

```
conv3_block3_1_bn (BatchNormali (None, None, None, 1 512
conv3_block3_1_conv[0] [0]

-----
-----  
conv3_block3_1_relu (Activation (None, None, None, 1 0
conv3_block3_1_bn[0] [0]

-----
-----  
conv3_block3_2_conv (Conv2D)      (None, None, None, 1 147584
conv3_block3_1_relu[0] [0]

-----
-----  
conv3_block3_2_bn (BatchNormali (None, None, None, 1 512
conv3_block3_2_conv[0] [0]

-----
-----  
conv3_block3_2_relu (Activation (None, None, None, 1 0
conv3_block3_2_bn[0] [0]

-----
-----  
conv3_block3_3_conv (Conv2D)      (None, None, None, 5 66048
conv3_block3_2_relu[0] [0]

-----
-----  
conv3_block3_3_bn (BatchNormali (None, None, None, 5 2048
conv3_block3_3_conv[0] [0]

-----
-----  
conv3_block3_add (Add)           (None, None, None, 5 0
conv3_block2_out[0] [0]
conv3_block3_3_bn[0] [0]

-----
-----  
conv3_block3_out (Activation)    (None, None, None, 5 0
conv3_block3_add[0] [0]

-----
-----  
conv3_block4_1_conv (Conv2D)      (None, None, None, 1 65664
conv3_block3_out[0] [0]

-----
-----  
conv3_block4_1_bn (BatchNormali (None, None, None, 1 512
conv3_block4_1_conv[0] [0]

-----
-----  
conv3_block4_1_relu (Activation (None, None, None, 1 0
conv3_block4_1_bn[0] [0]
```

```
-----  
conv3_block4_2_conv (Conv2D)      (None, None, None, 1 147584  
conv3_block4_1_relu[0] [0]  
-----
```

```
-----  
conv3_block4_2_bn (BatchNormali (None, None, None, 1 512  
conv3_block4_2_conv[0] [0]  
-----
```

```
-----  
conv3_block4_2_relu (Activation (None, None, None, 1 0  
conv3_block4_2_bn[0] [0]  
-----
```

```
-----  
conv3_block4_3_conv (Conv2D)      (None, None, None, 5 66048  
conv3_block4_2_relu[0] [0]  
-----
```

```
-----  
conv3_block4_3_bn (BatchNormali (None, None, None, 5 2048  
conv3_block4_3_conv[0] [0]  
-----
```

```
-----  
conv3_block4_add (Add)          (None, None, None, 5 0  
conv3_block3_out[0] [0]  
conv3_block4_3_bn[0] [0]  
-----
```

```
-----  
conv3_block4_out (Activation)   (None, None, None, 5 0  
conv3_block4_add[0] [0]  
-----
```

```
-----  
conv4_block1_1_conv (Conv2D)    (None, None, None, 2 131328  
conv3_block4_out[0] [0]  
-----
```

```
-----  
conv4_block1_1_bn (BatchNormali (None, None, None, 2 1024  
conv4_block1_1_conv[0] [0]  
-----
```

```
-----  
conv4_block1_1_relu (Activation (None, None, None, 2 0  
conv4_block1_1_bn[0] [0]  
-----
```

```
-----  
conv4_block1_2_conv (Conv2D)    (None, None, None, 2 590080  
conv4_block1_1_relu[0] [0]  
-----
```

```
-----  
conv4_block1_2_bn (BatchNormali (None, None, None, 2 1024  
conv4_block1_2_conv[0] [0]
```

```
-----  
conv4_block1_2_relu (Activation (None, None, None, 2 0  
conv4_block1_2_bn[0] [0]  
  
-----  
conv4_block1_0_conv (Conv2D)      (None, None, None, 1 525312  
conv3_block4_out[0] [0]  
  
-----  
conv4_block1_3_conv (Conv2D)      (None, None, None, 1 263168  
conv4_block1_2_relu[0] [0]  
  
-----  
conv4_block1_0_bn (BatchNormali (None, None, None, 1 4096  
conv4_block1_0_conv[0] [0]  
  
-----  
conv4_block1_3_bn (BatchNormali (None, None, None, 1 4096  
conv4_block1_3_conv[0] [0]  
  
-----  
conv4_block1_add (Add)           (None, None, None, 1 0  
conv4_block1_0_bn[0] [0]  
conv4_block1_3_bn[0] [0]  
  
-----  
conv4_block1_out (Activation)    (None, None, None, 1 0  
conv4_block1_add[0] [0]  
  
-----  
conv4_block2_1_conv (Conv2D)     (None, None, None, 2 262400  
conv4_block1_out[0] [0]  
  
-----  
conv4_block2_1_bn (BatchNormali (None, None, None, 2 1024  
conv4_block2_1_conv[0] [0]  
  
-----  
conv4_block2_1_relu (Activation (None, None, None, 2 0  
conv4_block2_1_bn[0] [0]  
  
-----  
conv4_block2_2_conv (Conv2D)     (None, None, None, 2 590080  
conv4_block2_1_relu[0] [0]  
  
-----  
conv4_block2_2_bn (BatchNormali (None, None, None, 2 1024
```

```
conv4_block2_2_conv[0] [0]
-----
conv4_block2_2_relu (Activation (None, None, None, 2 0
conv4_block2_2_bn[0] [0]
-----
conv4_block2_3_conv (Conv2D)      (None, None, None, 1 263168
conv4_block2_2_relu[0] [0]
-----
conv4_block2_3_bn (BatchNormali (None, None, None, 1 4096
conv4_block2_3_conv[0] [0]
-----
conv4_block2_add (Add)          (None, None, None, 1 0
conv4_block1_out[0] [0]
conv4_block2_3_bn[0] [0]
-----
conv4_block2_out (Activation)   (None, None, None, 1 0
conv4_block2_add[0] [0]
-----
conv4_block3_1_conv (Conv2D)    (None, None, None, 2 262400
conv4_block2_out[0] [0]
-----
conv4_block3_1_bn (BatchNormali (None, None, None, 2 1024
conv4_block3_1_conv[0] [0]
-----
conv4_block3_1_relu (Activation (None, None, None, 2 0
conv4_block3_1_bn[0] [0]
-----
conv4_block3_2_conv (Conv2D)    (None, None, None, 2 590080
conv4_block3_1_relu[0] [0]
-----
conv4_block3_2_bn (BatchNormali (None, None, None, 2 1024
conv4_block3_2_conv[0] [0]
-----
conv4_block3_2_relu (Activation (None, None, None, 2 0
conv4_block3_2_bn[0] [0]
```

```
conv4_block3_3_conv (Conv2D)      (None, None, None, 1 263168
conv4_block3_2_relu[0] [0]

-----
----- conv4_block3_3_bn (BatchNormali (None, None, None, 1 4096
conv4_block3_3_conv [0] [0]

-----
----- conv4_block3_add (Add)          (None, None, None, 1 0
conv4_block2_out [0] [0]
conv4_block3_3_bn [0] [0]

-----
----- conv4_block3_out (Activation)   (None, None, None, 1 0
conv4_block3_add [0] [0]

-----
----- conv4_block4_1_conv (Conv2D)    (None, None, None, 2 262400
conv4_block3_out [0] [0]

-----
----- conv4_block4_1_bn (BatchNormali (None, None, None, 2 1024
conv4_block4_1_conv [0] [0]

-----
----- conv4_block4_1_relu (Activation (None, None, None, 2 0
conv4_block4_1_bn [0] [0]

-----
----- conv4_block4_2_conv (Conv2D)    (None, None, None, 2 590080
conv4_block4_1_relu [0] [0]

-----
----- conv4_block4_2_bn (BatchNormali (None, None, None, 2 1024
conv4_block4_2_conv [0] [0]

-----
----- conv4_block4_2_relu (Activation (None, None, None, 2 0
conv4_block4_2_bn [0] [0]

-----
----- conv4_block4_3_conv (Conv2D)    (None, None, None, 1 263168
conv4_block4_2_relu [0] [0]

-----
----- conv4_block4_3_bn (BatchNormali (None, None, None, 1 4096
conv4_block4_3_conv [0] [0]
```

```
-----  
conv4_block4_add (Add)           (None, None, None, 1 0  
conv4_block3_out[0] [0]  
conv4_block4_3_bn[0] [0]  
  
-----  
conv4_block4_out (Activation)   (None, None, None, 1 0  
conv4_block4_add[0] [0]  
  
-----  
conv4_block5_1_conv (Conv2D)    (None, None, None, 2 262400  
conv4_block4_out[0] [0]  
  
-----  
conv4_block5_1_bn (BatchNormali (None, None, None, 2 1024  
conv4_block5_1_conv[0] [0]  
  
-----  
conv4_block5_1_relu (Activation (None, None, None, 2 0  
conv4_block5_1_bn[0] [0]  
  
-----  
conv4_block5_2_conv (Conv2D)    (None, None, None, 2 590080  
conv4_block5_1_relu[0] [0]  
  
-----  
conv4_block5_2_bn (BatchNormali (None, None, None, 2 1024  
conv4_block5_2_conv[0] [0]  
  
-----  
conv4_block5_2_relu (Activation (None, None, None, 2 0  
conv4_block5_2_bn[0] [0]  
  
-----  
conv4_block5_3_conv (Conv2D)    (None, None, None, 1 263168  
conv4_block5_2_relu[0] [0]  
  
-----  
conv4_block5_3_bn (BatchNormali (None, None, None, 1 4096  
conv4_block5_3_conv[0] [0]  
  
-----  
conv4_block5_add (Add)           (None, None, None, 1 0  
conv4_block4_out[0] [0]  
conv4_block5_3_bn[0] [0]  
  
-----  
conv4_block5_out (Activation)   (None, None, None, 1 0
```

```
conv4_block5_add[0] [0]
-----
conv4_block6_1_conv (Conv2D)      (None, None, None, 2 262400
conv4_block5_out[0] [0]
-----
conv4_block6_1_bn (BatchNormali (None, None, None, 2 1024
conv4_block6_1_conv[0] [0]
-----
conv4_block6_1_relu (Activation (None, None, None, 2 0
conv4_block6_1_bn[0] [0]
-----
conv4_block6_2_conv (Conv2D)      (None, None, None, 2 590080
conv4_block6_1_relu[0] [0]
-----
conv4_block6_2_bn (BatchNormali (None, None, None, 2 1024
conv4_block6_2_conv[0] [0]
-----
conv4_block6_2_relu (Activation (None, None, None, 2 0
conv4_block6_2_bn[0] [0]
-----
conv4_block6_3_conv (Conv2D)      (None, None, None, 1 263168
conv4_block6_2_relu[0] [0]
-----
conv4_block6_3_bn (BatchNormali (None, None, None, 1 4096
conv4_block6_3_conv[0] [0]
-----
conv4_block6_add (Add)           (None, None, None, 1 0
conv4_block5_out[0] [0]
conv4_block6_3_bn[0] [0]
-----
conv4_block6_out (Activation)    (None, None, None, 1 0
conv4_block6_add[0] [0]
-----
conv5_block1_1_conv (Conv2D)      (None, None, None, 5 524800
conv4_block6_out[0] [0]
```

```
conv5_block1_1_bn (BatchNormali (None, None, None, 5 2048
conv5_block1_1_conv[0] [0]
-----
-----
conv5_block1_1_relu (Activation (None, None, None, 5 0
conv5_block1_1_bn[0] [0]
-----
-----
conv5_block1_2_conv (Conv2D)      (None, None, None, 5 2359808
conv5_block1_1_relu[0] [0]
-----
-----
conv5_block1_2_bn (BatchNormali (None, None, None, 5 2048
conv5_block1_2_conv[0] [0]
-----
-----
conv5_block1_2_relu (Activation (None, None, None, 5 0
conv5_block1_2_bn[0] [0]
-----
-----
conv5_block1_0_conv (Conv2D)      (None, None, None, 2 2099200
conv4_block6_out[0] [0]
-----
-----
conv5_block1_3_conv (Conv2D)      (None, None, None, 2 1050624
conv5_block1_2_relu[0] [0]
-----
-----
conv5_block1_0_bn (BatchNormali (None, None, None, 2 8192
conv5_block1_0_conv[0] [0]
-----
-----
conv5_block1_3_bn (BatchNormali (None, None, None, 2 8192
conv5_block1_3_conv[0] [0]
-----
-----
conv5_block1_add (Add)          (None, None, None, 2 0
conv5_block1_0_bn[0] [0]
conv5_block1_3_bn[0] [0]
-----
-----
conv5_block1_out (Activation)   (None, None, None, 2 0
conv5_block1_add[0] [0]
-----
-----
conv5_block2_1_conv (Conv2D)    (None, None, None, 5 1049088
conv5_block1_out[0] [0]
```

```
-----  
conv5_block2_1_bn (BatchNormali (None, None, None, 5 2048  
conv5_block2_1_conv[0] [0]  
  
-----  
conv5_block2_1_relu (Activation (None, None, None, 5 0  
conv5_block2_1_bn[0] [0]  
  
-----  
conv5_block2_2_conv (Conv2D)      (None, None, None, 5 2359808  
conv5_block2_1_relu[0] [0]  
  
-----  
conv5_block2_2_bn (BatchNormali (None, None, None, 5 2048  
conv5_block2_2_conv[0] [0]  
  
-----  
conv5_block2_2_relu (Activation (None, None, None, 5 0  
conv5_block2_2_bn[0] [0]  
  
-----  
conv5_block2_3_conv (Conv2D)      (None, None, None, 2 1050624  
conv5_block2_2_relu[0] [0]  
  
-----  
conv5_block2_3_bn (BatchNormali (None, None, None, 2 8192  
conv5_block2_3_conv[0] [0]  
  
-----  
conv5_block2_add (Add)           (None, None, None, 2 0  
conv5_block1_out[0] [0]  
conv5_block2_3_bn[0] [0]  
  
-----  
conv5_block2_out (Activation)    (None, None, None, 2 0  
conv5_block2_add[0] [0]  
  
-----  
conv5_block3_1_conv (Conv2D)      (None, None, None, 5 1049088  
conv5_block2_out[0] [0]  
  
-----  
conv5_block3_1_bn (BatchNormali (None, None, None, 5 2048  
conv5_block3_1_conv[0] [0]  
  
-----  
conv5_block3_1_relu (Activation (None, None, None, 5 0  
conv5_block3_1_bn[0] [0]
```

```
-----  
conv5_block3_2_conv (Conv2D)      (None, None, None, 5 2359808  
conv5_block3_1_relu[0] [0]  
  
-----  
conv5_block3_2_bn (BatchNormali (None, None, None, 5 2048  
conv5_block3_2_conv[0] [0]  
  
-----  
conv5_block3_2_relu (Activation (None, None, None, 5 0  
conv5_block3_2_bn[0] [0]  
  
-----  
conv5_block3_3_conv (Conv2D)      (None, None, None, 2 1050624  
conv5_block3_2_relu[0] [0]  
  
-----  
conv5_block3_3_bn (BatchNormali (None, None, None, 2 8192  
conv5_block3_3_conv[0] [0]  
  
-----  
conv5_block3_add (Add)           (None, None, None, 2 0  
conv5_block2_out[0] [0]  
conv5_block3_3_bn[0] [0]  
  
-----  
conv5_block3_out (Activation)    (None, None, None, 2 0  
conv5_block3_add[0] [0]  
  
-----  
global_max_pooling2d_3 (GlobalM (None, 2048)          0  
conv5_block3_out[0] [0]  
  
-----  
dropout_6 (Dropout)             (None, 2048)          0  
global_max_pooling2d_3[0] [0]  
  
-----  
dense_12 (Dense)               (None, 100)           204900      dropout_6[0] [0]  
  
-----  
dense_13 (Dense)               (None, 75)            7575        dense_12[0] [0]  
  
-----  
dense_14 (Dense)               (None, 50)            3800        dense_13[0] [0]  
-----
```

```

dropout_7 (Dropout)           (None, 50)      0          dense_14[0] [0]
-----
dense_15 (Dense)             (None, 2)       102        dropout_7[0] [0]
=====
=====
Total params: 23,804,089
Trainable params: 23,750,969
Non-trainable params: 53,120
=====
=====
```

2.4 Plot the result

```
[1]: labels = {
    0:'Female',
    1:'Male'
}

[29]: predictions

[29]: array([[2.4339561e-01, 7.5660443e-01],
           [5.0009745e-01, 4.9990255e-01],
           [7.5667405e-01, 2.4332592e-01],
           ...,
           [6.2667037e-05, 9.9993730e-01],
           [1.6162945e-03, 9.9838376e-01],
           [9.8220026e-01, 1.7799696e-02]], dtype=float32)

[33]: # Plot a random sample of 12 test images, their predicted labels and ground
      ↵truth
      figure = plt.figure(figsize=(9, 12))
      for i, index in enumerate(np.random.choice(test_images.shape[0], size=12, u
      ↵replace=False)):
          ax = figure.add_subplot(4, 3, i + 1, xticks=[], yticks[])
          # Display each image
          ax.imshow(np.squeeze(test_images[index]))
          predict_index = np.argmax(predictions[index])
          true_index = test_labels[index]
          # Set the title for each image
          ax.set_title("{} ({})".format(labels[predict_index],
                                         labels[true_index]),
                      color="green" if predict_index == true_index
          ↵else "red"))
      plt.show()
```

Female (Female)



Male (Male)



Female (Female)



Male (Male)



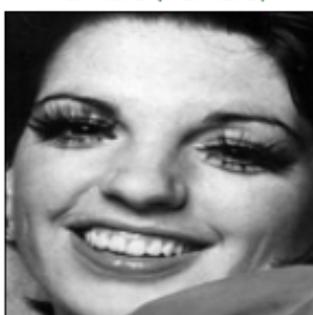
Male (Male)



Male (Male)



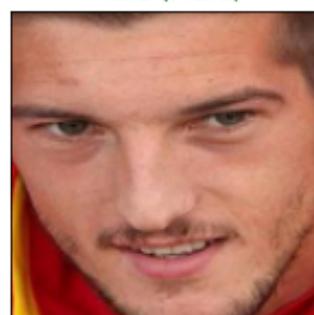
Female (Female)



Male (Male)



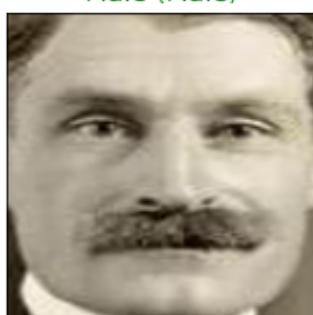
Male (Male)



Male (Male)



Male (Male)



Male (Male)



[34]: # Look at confusion matrix

```

def plot_confusion_matrix(cm, classes,
                         normalize=False,
                         title='Confusion matrix',
                         cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, ['female','male'])
    plt.yticks(tick_marks, ['female','male'], rotation=90)

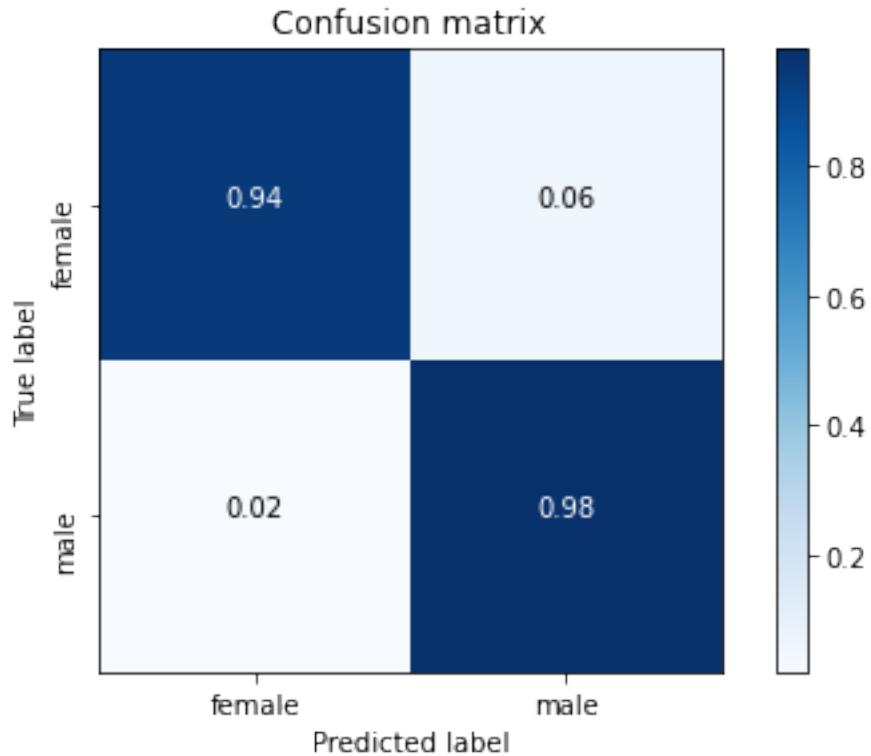
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, round(cm[i, j],2),
                 horizontalalignment="center",
                 color="white" if round(cm[i, j],2) > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

# Predict the values from the validation dataset
Y_pred = predictions
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred, axis = 1)
# Convert validation observations to one hot vectors
Y_true = test_labels
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(2),normalize=True)

```



3 Prediction when there are more than one people in the photo

3.1 Age detection

3.1.1 Load fine-tuned model for age

```
[34]: dict_age = {'(0,18)' : 0,
                 '(18,30)' : 1,
                 '(30,40)' : 2,
                 '(40,60)' : 3,
                 '(60,100+)' : 4}
```

```
[35]: # 'baseline_age1.h5' gets the highest validation accuracy, so we use it as our prediction model
      file_path = 'baseline_age1.h5'
      model = get_model(n_classes=len(dict_age))
      model.load_weights(file_path)
```

3.1.2 Load the photo

```
[114]: # use MTCNN to make face detection
import cv2
from mtcnn import MTCNN
from PIL import Image

detector = MTCNN()
image = cv2.cvtColor(cv2.imread("7.png"), cv2.COLOR_BGR2RGB)
result = detector.detect_faces(image)
im = Image.open("7.png")
```

WARNING:tensorflow:5 out of the last 19 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000024414501670> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:6 out of the last 20 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000024414501670> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

```
[115]: im
```

```
[115]:
```



3.1.3 Face detection

```
[147]: # get all faces in the photo
images_all = []
for k in range(len(result)):
    bounding_box = result[k]['box']
    crop_rectangle = (bounding_box[0], bounding_box[1], □
    ↳bounding_box[0]+bounding_box[2], bounding_box[1] + bounding_box[3])
    cropped_im = im.crop(crop_rectangle)
    images_all.append(cropped_im)
```

3.1.4 Resize the photos

```
[148]: # resize the photos
def read_and_resize_multi(i,im, input_shape=(128, 128)):
    bounding_box = result[i]['box']
    crop_rectangle = (bounding_box[0], bounding_box[1], □
    ↳bounding_box[0]+bounding_box[2], bounding_box[1] + bounding_box[3])
    cropped_im = im.crop(crop_rectangle)
    im = cropped_im.convert('RGB')
    im = im.resize(input_shape)
    im_array = np.array(im, dtype="uint8")#[..., ::-1]
    return np.array(im_array / (np.max(im_array)+ 0.001), dtype="float32")
```

```
[149]: multi_photos = np.array([read_and_resize_multi(i,im) for i in □
    ↳range(len(result))])
```

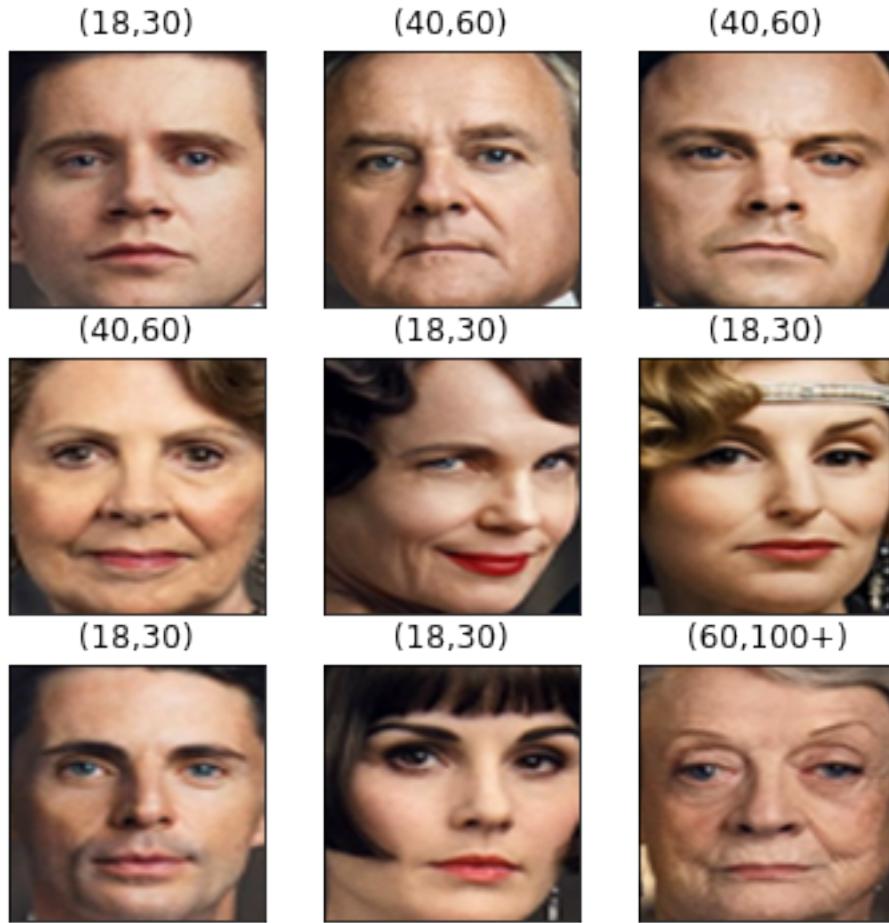
3.1.5 Predict and plot the results

```
[150]: multi_pred = model.predict(multi_photos)
```

```
[151]: labels = {  
    0: '(0,18)',  
    1: '(18,30)',  
    2: '(30,40)',  
    3: '(40,60)',  
    4: '(60,100+)'  
}
```

```
[152]: import matplotlib.pyplot as plt
```

```
[157]: # Plot the predicted labels for age  
figure = plt.figure(figsize=(6, 6))  
for i, index in enumerate(np.array(range(len(result))):  
    ax = figure.add_subplot(3, 3, i + 1, xticks=[], yticks=[])  
    # Display each image  
    ax.imshow(np.squeeze(multi_photos[index]))  
    predict_index = np.argmax(multi_pred[index])  
    # Set the title for each image  
    ax.set_title("{}{}".format(labels[predict_index]))  
plt.show()
```



3.2 Gender detection

3.2.1 Load the model

```
[158]: # load the best model for gender
file_path = "baseline_gender_0.h5"
model_gender = get_model(n_classes=2)
model_gender.load_weights(file_path)
labels = {
    0:'Female',
    1:'Male'
}
```

WARNING:tensorflow:5 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x00000243EF015280> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of

tensors. For (1), please define your `@tf.function` outside of the loop. For (2), `@tf.function` has `experimental_relax_shapes=True` option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

3.2.2 Predict and plot the results

```
[159]: # Plot the predicted labels for gender
multi_pred = model_gender.predict(multi_photos)
figure = plt.figure(figsize=(6, 6))
for i, index in enumerate(np.array(range(len(result)))):
    ax = figure.add_subplot(3, 3, i + 1, xticks=[], yticks=[])
    # Display each image
    ax.imshow(np.squeeze(multi_photos[index]))
    predict_index = np.argmax(multi_pred[index])
    # Set the title for each image
    ax.set_title("{}{}".format(labels[predict_index]))
plt.show()
```



