Apparent Age and Gender Prediction in Keras

February 13, 2019 / Machine Learning

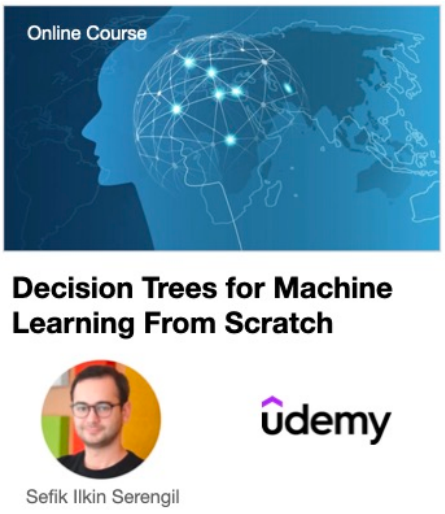
Computer vision researchers of ETH Zurich University (Switzerland) announced a very successful apparent age and gender prediction models. They both shared how they designed the machine learning model and pre-trained weights for transfer learning. Their implementation was based on Caffe framework. Even though I tried to convert Caffe model and weights to Keras / TensorFlow, I couldn't handle this. That's why, I intend to adopt this research from scratch in Keras.

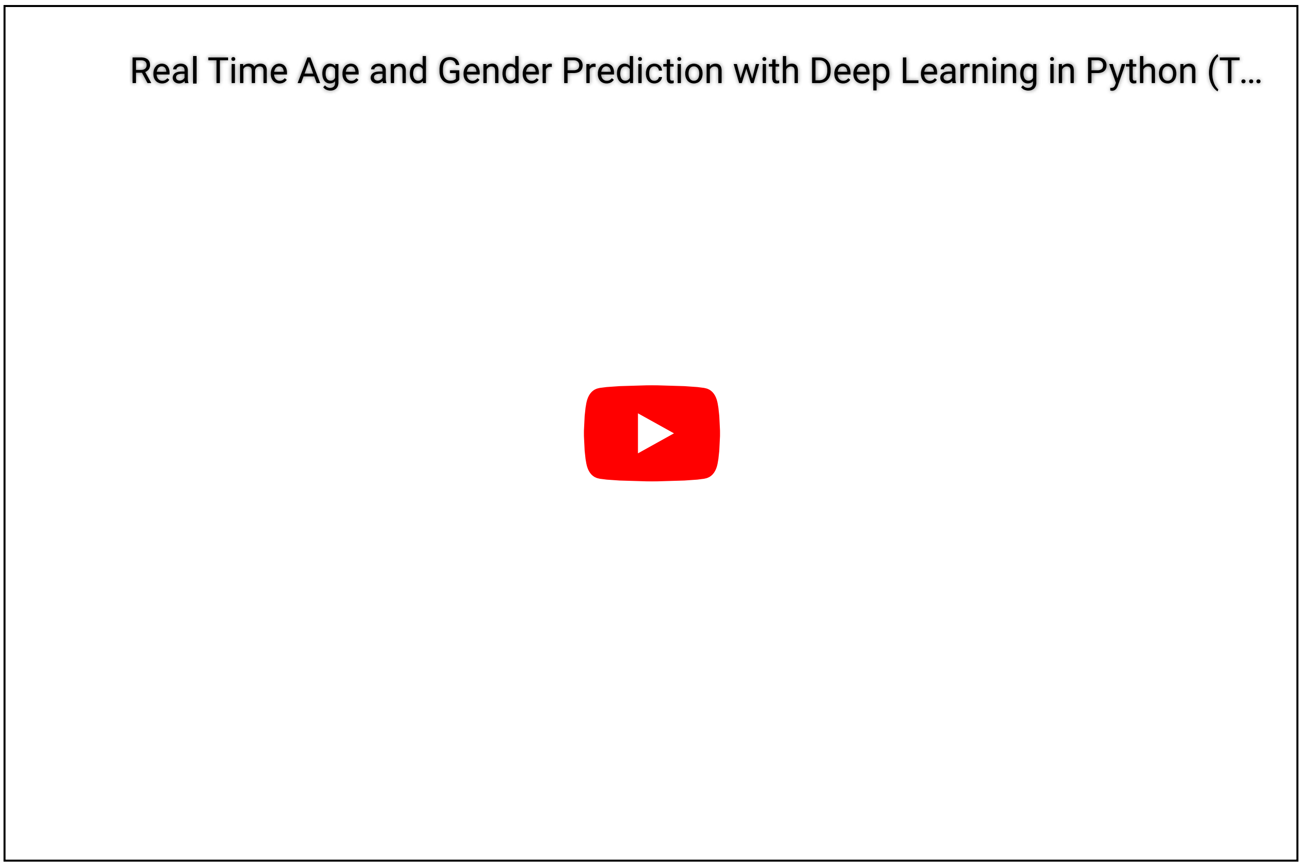
Katy Perry Transformation

What this post offers?

We can apply age and gender predictions in real time.

🙋 You may consider to enroll my top-rated machine learning course on Udemy

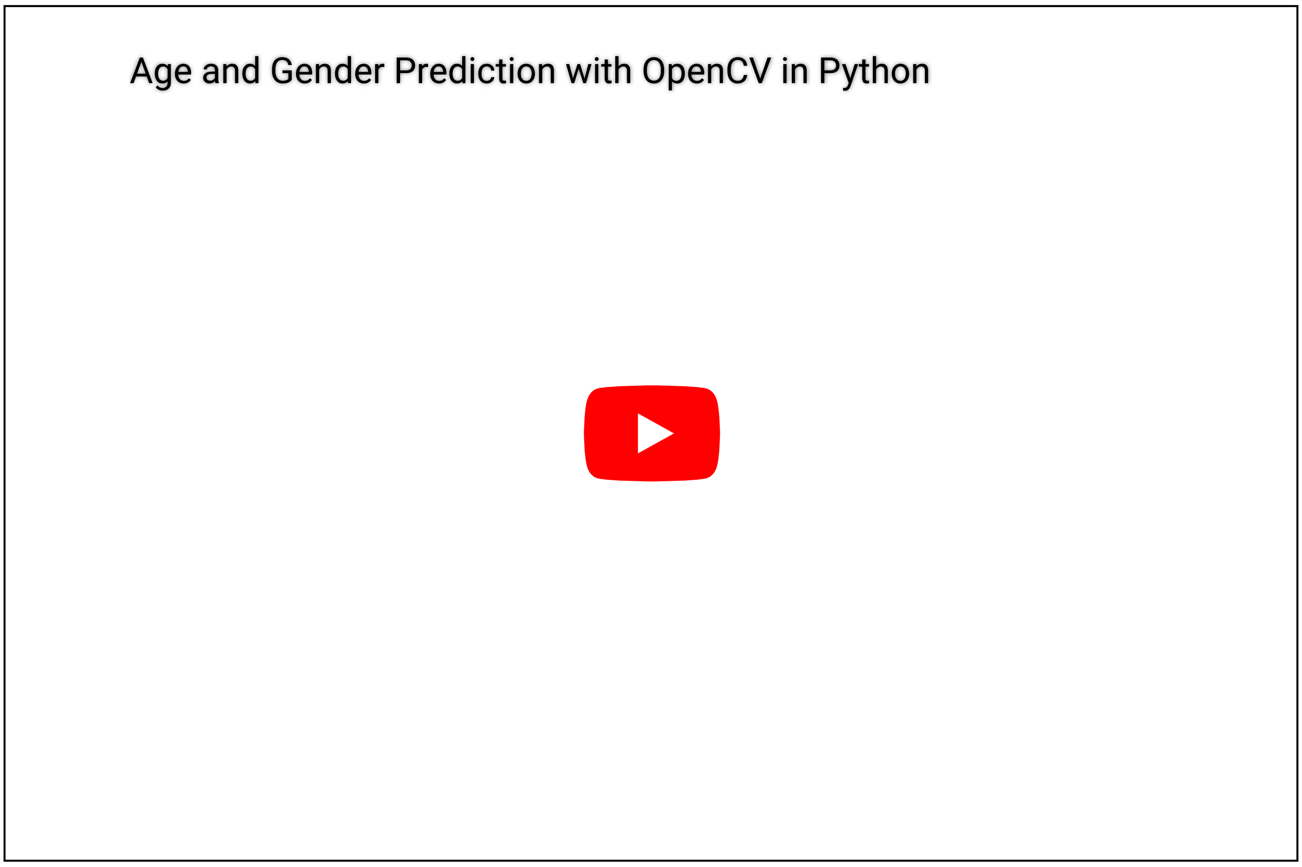



Real Time Age and Gender Prediction with Deep Learning in Python (T...

DeepFace library for Python covers age prediction. You can run age estimation with a few lines of code.

Pre-trained model

In this post, we are going to re-train the age and gender prediction models from scratch. If you focus on just prediction stage, then the following video might attract your attention. This subject is covered in a dedicated blog post actually: **Age and Gender Prediction with Deep Learning in OpenCV**. We will use pre-trained models for Caffe within OpenCV in this case. Besides, you don't have to have Caffe on your environment. OpenCV handles to build Caffe models with its dnn module.



On the other hand, if training stage attracts your attention, then you should continue to read this blog post.

Dataset

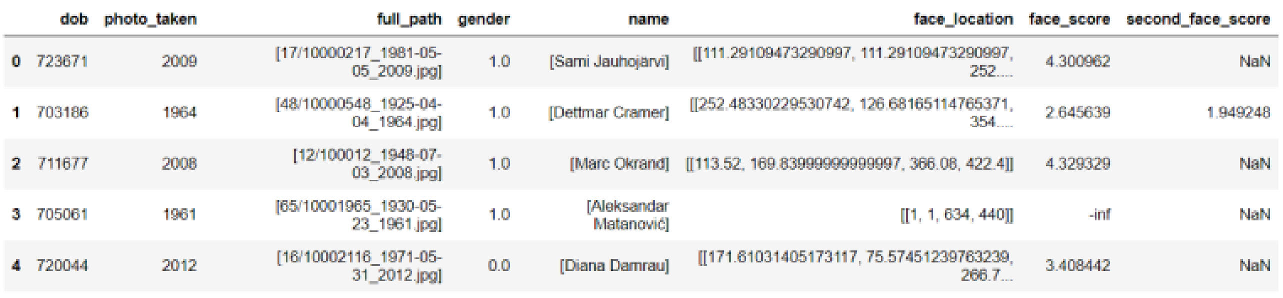
The original work consumed face pictures collected from IMDB (7 GB) and Wikipedia (1 GB). You can find these data sets here. In this post, I will just consume wiki data source to develop solution fast. You should **download faces only** files.

Extracting wiki_crop.tar creates 100 folders and an index file (wiki.mat). The index file [illegible] as Matlab format. We can read Matlab files in python with SciPy.

```
import scipy.io
mat = scipy.io.loadmat(&amp;#039;wiki_crop/wiki.mat&amp;#039;)
```

Converting pandas dataframe will make transformations easier.

```
instances = mat[&amp;#039;wiki&amp;#039;][0][0][0].shape[1]

columns = [&amp;amp;quot;dob&amp;amp;quot;, &amp;amp;quot;photo

import pandas as pd
df = pd.DataFrame(index = range(0,instances), columns = column

for i in mat:
if i == &amp;amp;quot;wiki&amp;amp;quot;:
current_array = mat[i][0][0]
for j in range(len(current_array)):
df[columns[j]] = pd.DataFrame(current_array[j][0])
```

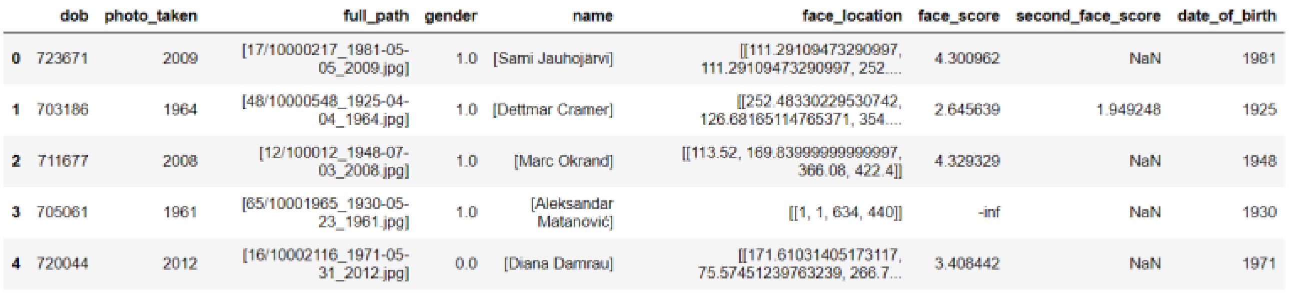


	dob	photo_taken	full_path	gender	name	face_location	face_score	second_face_score
0	723671	2009	[17/10000217_1981-05-05_2009.jpg]	1.0	[Sami Jauhojärvi]	[[111.29109473290997, 111.29109473290997, 252....	4.300962	NaN
1	703186	1964	[48/10000548_1925-04-04_1964.jpg]	1.0	[Dettmar Cramer]	[[252.48330229530742, 126.68165114765371, 354....	2.645639	1.949248
2	711677	2008	[12/100012_1948-07-03_2008.jpg]	1.0	[Marc Okrand]	[[113.52, 169.83999999999997, 366.08, 422.4]]	4.329329	NaN
3	705061	1961	[65/10001965_1930-05-23_1961.jpg]	1.0	[Aleksandar Matanović]	[[1, 1, 634, 440]]	-inf	NaN
4	720044	2012	[16/10002116_1971-05-31_2012.jpg]	0.0	[Diana Damrau]	[[171.61031405173117, 75.57451239763239, 266.7...	3.408442	NaN

Initial data set

Data set contains date of birth (dob) in Matlab datenum format. We need to convert this to Python datatime format. We just need the birth year.

```
from datetime import datetime, timedelta
def datenum_to_datetime(datenum):
days = datenum % 1
hours = days % 1 * 24
minutes = hours % 1 * 60
seconds = minutes % 1 * 60
exact_date = datetime.fromordinal(int(datenum)) \
+ timedelta(days=int(days)) + timedelta(hours=int(hours)) \
+ timedelta(minutes=int(minutes)) + timedelta(seconds=round(se
- timedelta(days=366)

return exact_date.year

df[&amp;#039;date_of_birth&amp;#039;] = df[&amp;#039;dob&amp;#0
```



	dob	photo_taken	full_path	gender	name	face_location	face_score	second_face_score	date_of_birth
0	723671	2009	[17/10000217_1981-05-05_2009.jpg]	1.0	[Sami Jauhojärvi]	[[111.29109473290997, 111.29109473290997, 252....	4.300962	NaN	1981
1	703186	1964	[48/10000548_1925-04-04_1964.jpg]	1.0	[Dettmar Cramer]	[[252.48330229530742, 126.68165114765371, 354....	2.645639	1.949248	1925
2	711677	2008	[12/100012_1948-07-03_2008.jpg]	1.0	[Marc Okrand]	[[113.52, 169.83999999999997, 366.08, 422.4]]	4.329329	NaN	1948
3	705061	1961	[65/10001965_1930-05-23_1961.jpg]	1.0	[Aleksandar Matanović]	[[1, 1, 634, 440]]	-inf	NaN	1930
4	720044	2012	[16/10002116_1971-05-31_2012.jpg]	0.0	[Diana Damrau]	[[171.61031405173117, 75.57451239763239, 266.7...	3.408442	NaN	1971

Adding exact birth date

Extracting date of birth from matlab datenum format

Now, we have both date of birth and photo taken time. Subtracting these values will give us the ages.

```
df[&amp;#039;age&amp;#039;] = df[&amp;#039;photo_taken&amp;#039;
```

Data cleaning

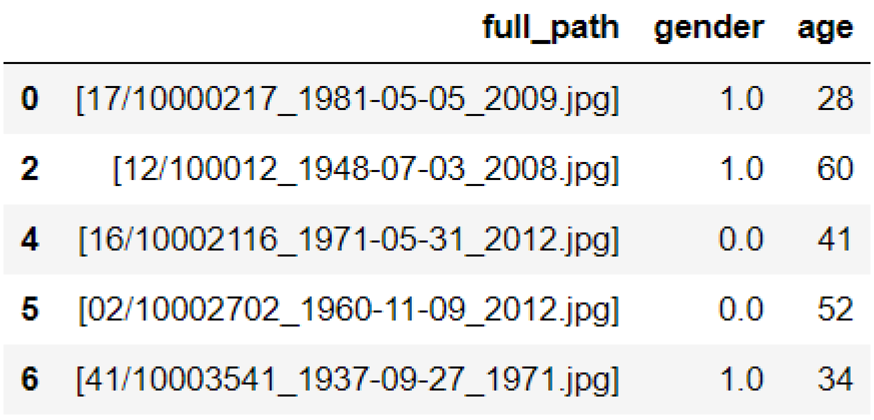
Some pictures don't include people in the wiki data set. For example, a vase picture exists in the data set. Moreover, some pictures might include two person. Furthermore, some are taken distant. Face score value can help us to understand the picture is clear or not. Also, age information is missing for some records. They all might confuse the model. We should ignore them. Finally, unnecessary columns should be dropped to occupy less memory.

```
#remove pictures does not include face
df = df[df[&amp;#039;face_score&amp;#039;] != -np.inf]

#some pictures include more than one face, remove them
df = df[df[&amp;#039;second_face_score&amp;#039;].isna()]

#check threshold
df = df[df[&amp;#039;face_score&amp;#039;] &amp;amp;amp;amp;amp

  ome records do not have a gender information
  = df[~df[&amp;#039;gender&amp;#039;].isna()]

df = df.drop(columns = [&amp;#039;name&amp;#039;,&amp;#039;face
```

Some pictures are taken for unborn people. Age value seems to be negative for some records. Dirty data might cause this. Moreover, some seems to be alive for more than 100. We should restrict the age prediction problem for 0 to 100 years.

```
#some guys seem to be greater than 100. some of these are painti
df = df[df[&amp;#039;age&amp;#039;] &amp;amp;amp;amp;amp;amp;amp;am

#some guys seem to be unborn in the data set
df = df[df[&amp;#039;age&amp;#039;] &amp;amp;amp;amp;amp;amp;amp;am
```

The raw data set will be look like the following data frame.

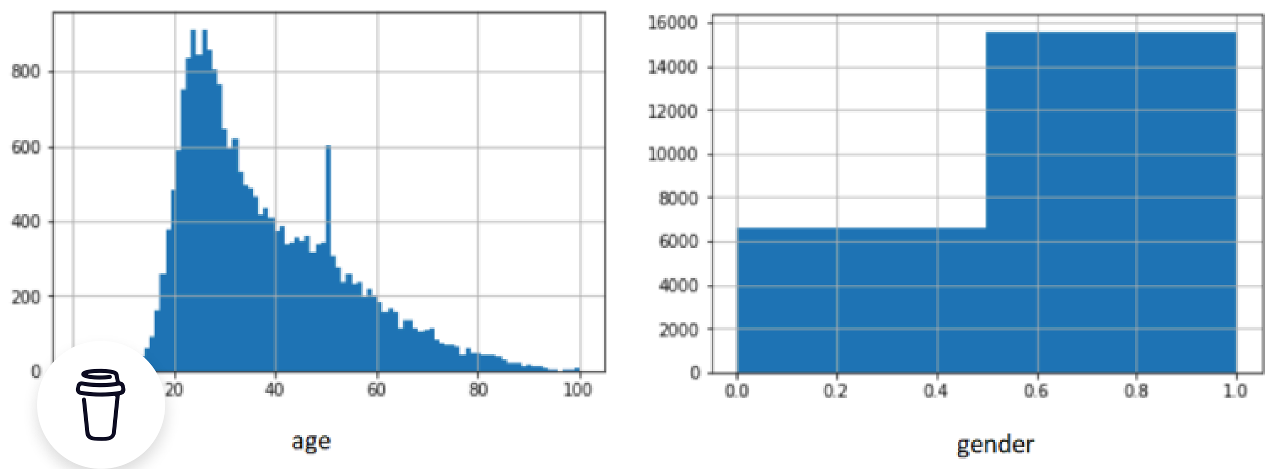


	full_path	gender	age
0	[17/10000217_1981-05-05_2009.jpg]	1.0	28
2	[12/100012_1948-07-03_2008.jpg]	1.0	60
4	[16/10002116_1971-05-31_2012.jpg]	0.0	41
5	[02/10002702_1960-11-09_2012.jpg]	0.0	52
6	[41/10003541_1937-09-27_1971.jpg]	1.0	34

Raw data set

We can visualize the target label distribution.

```
histogram_age = df[&amp;#039;age&amp;#039;].hist(bins=df[&amp;#0
histogram_gender = df[&amp;#039;gender&amp;#039;].hist(bins=df[&
```

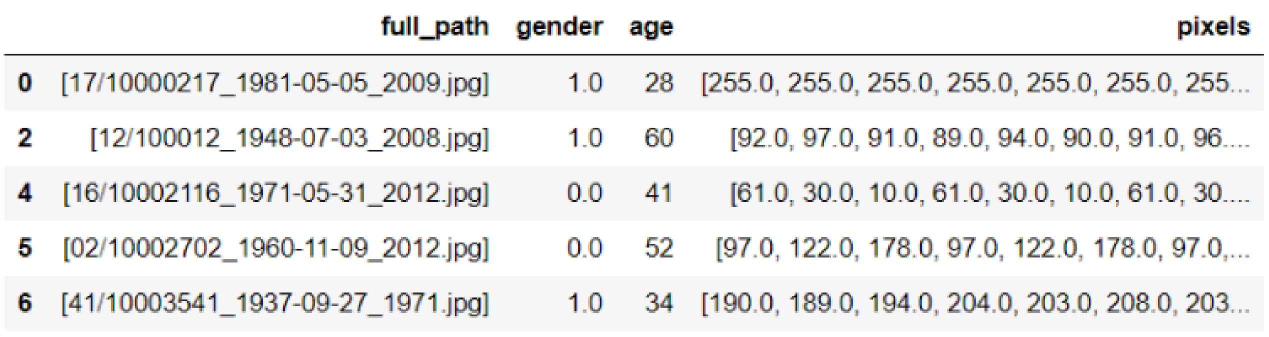


Age and gender distribution in the data set

Full path column states the exact location of the picture on the disk. We need its pixel values.

```
target_size = (224, 224)

def getImagePixels(image_path):
img = image.load_img(&amp;amp;quot;wiki_crop/%s&amp;amp;quot; %
x = image.img_to_array(img).reshape(1, -1)[0]
#x = preprocess_input(x)
return x

df[&amp;#039;pixels&amp;#039;] = df[&amp;#039;full_path&amp;#03
```

We can extract the real pixel values of pictures



	full_path	gender	age	pixels
0	[17/10000217_1981-05-05_2009.jpg]	1.0	28	[255.0, 255.0, 255.0, 255.0, 255.0, 255.0, 255...
2	[12/100012_1948-07-03_2008.jpg]	1.0	60	[92.0, 97.0, 91.0, 89.0, 94.0, 90.0, 91.0, 96....
4	[16/10002116_1971-05-31_2012.jpg]	0.0	41	[61.0, 30.0, 10.0, 61.0, 30.0, 10.0, 61.0, 30....
5	[02/10002702_1960-11-09_2012.jpg]	0.0	52	[97.0, 122.0, 178.0, 97.0, 122.0, 178.0, 97.0,...
6	[41/10003541_1937-09-27_1971.jpg]	1.0	34	[190.0, 189.0, 194.0, 204.0, 203.0, 208.0, 203...

Adding pixels

Apparent age prediction model

Age prediction is a regression problem. But researchers define it as a classification problem. There are 101 classes in the output layer for ages 0 to 100. they applied transfer learning for this duty. Their choice was VGG for imagenet.

Preparing input output

Pandas data frame includes both input and output information for age and gender prediction tasks. Wee should just focus on the age task.

```
classes = 101 #0 to 100
target = df[&amp;#039;age&amp;#039;].values
rget_classes = keras.utils.to_categorical(target, classes)

eatures = []

for i in range(0, df.shape[0]):
features.append(df[&amp;#039;pixels&amp;#039;].values[i])

features = np.array(features)
features = features.reshape(features.shape[0], 224, 224, 3)
```

Also, we need to split data set as training and testing set.

```
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(features, ta
```

The final data set consists of 22578 instances. It is splitted into 15905 train instances and 6673 test instances .

Transfer learning

As mentioned, researcher used VGG imagenet model. Still, they tuned weights for this data set. Herein, I prefer to use **VGG-Face** model. Because, this model is tuned for face recognition task. In this way, we might have outcomes for patterns in the human face.

```
#VGG-Face model
model = Sequential()
model.add(ZeroPadding2D((1,1),input_shape=(224,224, 3)))
model.add(Convolution2D(64, (3, 3), activation=&amp;#039;relu&a
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(64, (3, 3), activation=&amp;#039;relu&a
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(128, (3, 3), activation=&amp;#039;relu&
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(128, (3, 3), activation=&amp;#039;relu&
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, (3, 3), activation=&amp;#039;relu&
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, (3, 3), activation=&amp;#039;relu&
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, (3, 3), activation=&amp;#039;relu&
  del.add(MaxPooling2D((2,2), strides=(2,2)))

  del.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation=&amp;#039;relu&
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation=&amp;#039;relu&
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation=&amp;#039;relu&
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation=&amp;#039;relu&
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation=&amp;#039;relu&
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation=&amp;#039;relu&
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(Convolution2D(4096, (7, 7), activation=&amp;#039;relu
model.add(Dropout(0.5))
model.add(Convolution2D(4096, (1, 1), activation=&amp;#039;relu
model.add(Dropout(0.5))
model.add(Convolution2D(2622, (1, 1)))
model.add(Flatten())
model.add(Activation(&amp;#039;softmax&amp;#039;))
```

Load the pre-trained weights for VGG-Face model. You can find the related blog post here.

```
#pre-trained weights of vgg-face model.
#you can find it here: https://github.com/serengil/deepface_mode
#related blog post: https://sefiks.com/2018/08/06/deep-face-reco
model.load_weights(&amp;#039;vgg_face_weights.h5&amp;#039;)
```

We should lock the layer weights for early layers because they could already detect some patterns. Fitting the network from scratch might cause to lose this important information. I prefer to freeze all layers except last 3 convolution layers (make exception for last 7 model.add units). Also, I cut the last convolution layer because it has 2622 units. I need just 101 (ages from 0 to 100) units for age prediction task. Then, add a custom convolution layer consisting of 101 units.

```
for layer in model.layers[:-7]:
layer.trainable = False

base_model_output = Sequential()
base_model_output = Convolution2D(101, (1, 1), name=&amp;#039;pr
base_model_output = Flatten()(base_model_output)
base_model_output = Activation(&amp;#039;softmax&amp;#039;)(base

age_model = Model(inputs=model.input, outputs=base_model_output)
```

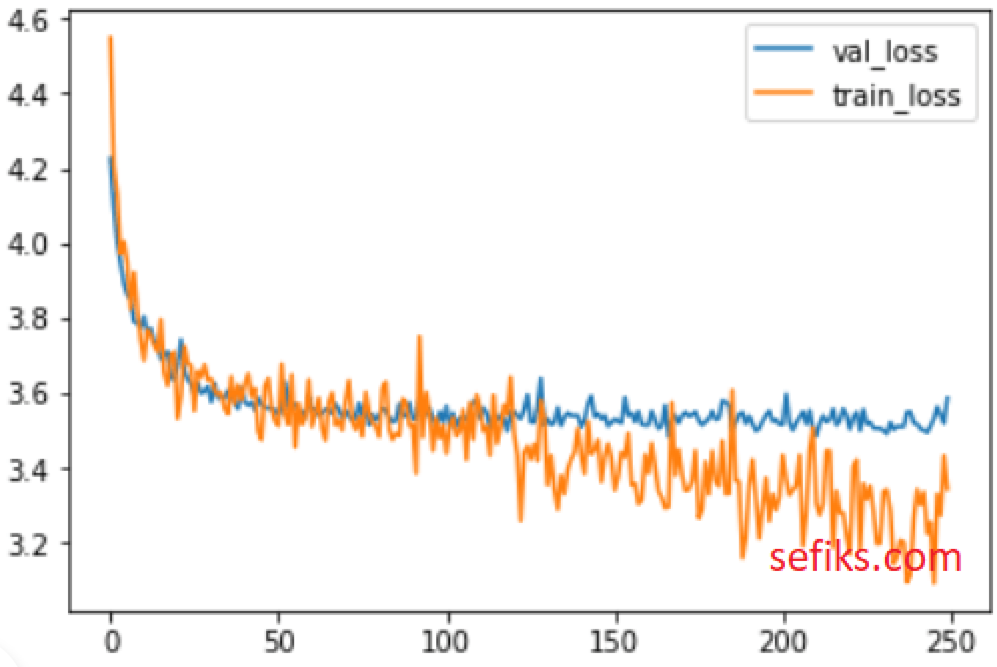
Training

This is a multi-class classification problem. Loss function must be categorical crossentropy. Optimization algorithm will be Adam to converge loss faster. I create a checkpoint to monitor model over iterations and avoid overfitting. The iteration which has the minimum validation loss value will include the optimum weights. That's why, I'll monitor validation loss and save the best one only.

To avoid overfitting, I feed random 256 instances for each epoch.

```
age_model.compile(loss=&amp;#039;categorical_crossentropy&amp;#

checkpointer = ModelCheckpoint(filepath=&amp;#039;age_model.hd
, monitor = &amp;amp;quot;val_loss&amp;amp;quot;, verbose=1, s

scores = []
epochs = 250; batch_size = 256

for i in range(epochs):
print(&amp;amp;quot;epoch &amp;amp;quot;,i)

ix_train = np.random.choice(train_x.shape[0], size=batch_size)

score = age_model.fit(train_x[ix_train], train_y[ix_train]
, epochs=1, validation_data=(test_x, test_y), callbacks=[check

scores.append(score)
```

It seems that validation loss reach the minimum. Increasing epochs will cause to overfitting.



Loss for age prediction task

Model evaluation on test set

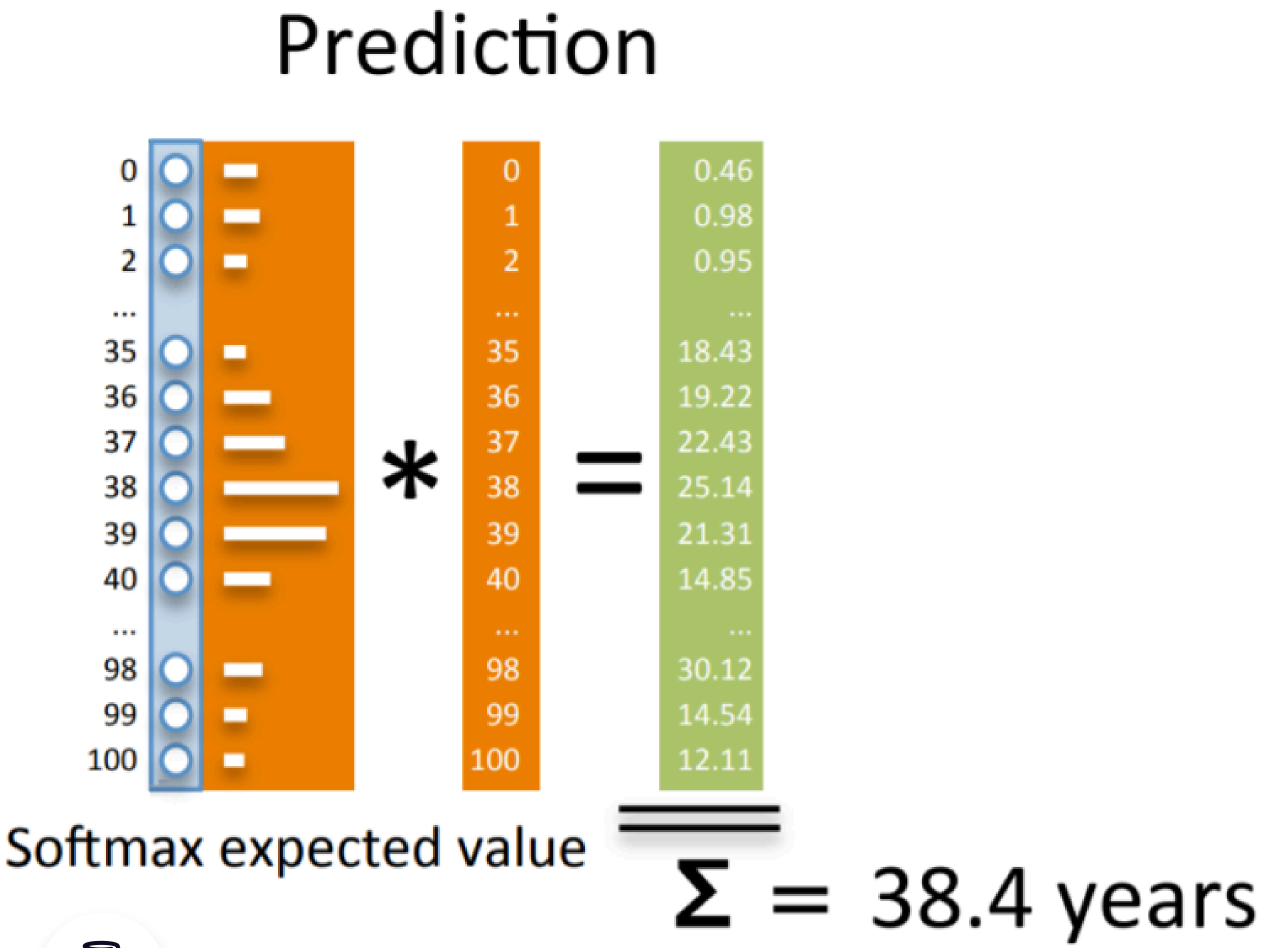
We can evaluate the final model on the test set.

```
age_model.evaluate(test_x, test_y, verbose=1)
```

This gives both validation loss and accuracy respectively for 6673 test instances. It seems that we have the following results.

[2.871919590848929, 0.24298789490543357]

24% accuracy seems very low, right? Actually, it is not. Herein, researchers develop an age prediction approach and convert classification task to regression. They propose that you should multiply each softmax out with its label. Summing this multiplications will be the apparent age prediction.



Age prediction approach

This is a very easy operation in Python numpy.

```
predictions = age_model.predict(test_x)
```

```

output_indexes = np.array([i for i in range(0, 101)])
apparent_predictions = np.sum(predictions * output_indexes, axi
```

Herein, mean absolute error metric might be more meaningful to evaluate the system.

```
mae = 0

for i in range(0 ,apparent_predictions.shape[0]):
prediction = int(apparent_predictions[i])
actual = np.argmax(test_y[i])

abs_error = abs(prediction - actual)
actual_mean = actual_mean + actual

mae = mae + abs_error

mae = mae / apparent_predictions.shape[0]

print(&amp;amp;quot;mae: &amp;amp;quot;,mae)
print(&amp;amp;quot;instances: &amp;amp;quot;,apparent_predict
```

Our apparent age prediction model averagely predict ages ± 4.65 error. This is acceptable.

Testing model on custom images

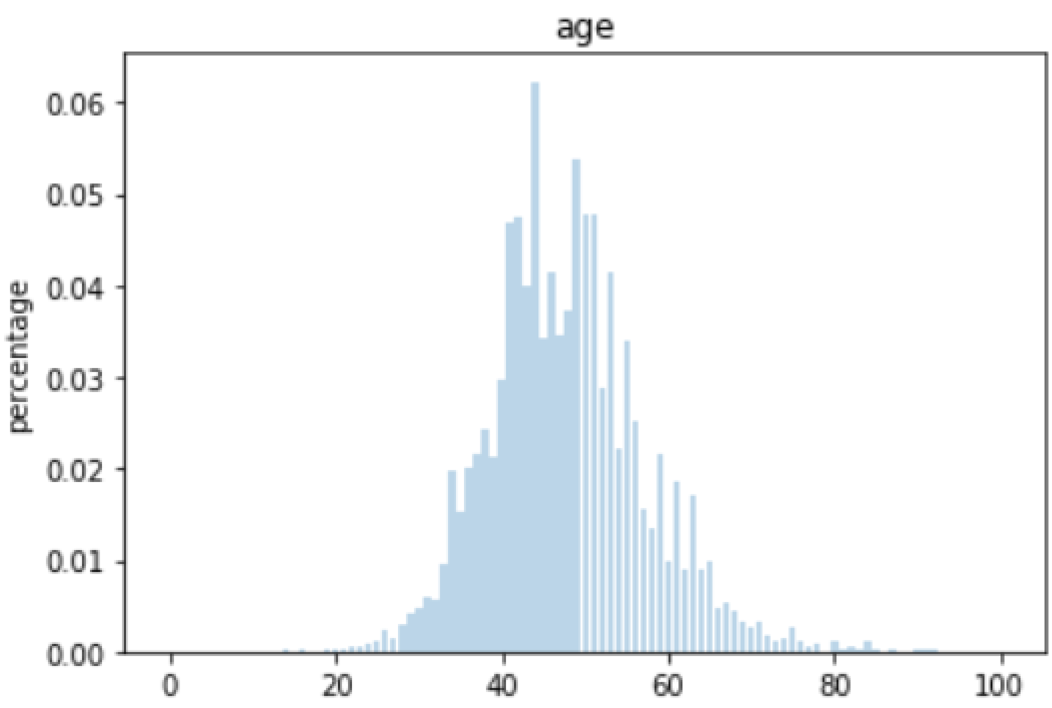
We can feel the power of the model when we feed custom images into it.

```
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator

def loadImage(filepath):
test_img = image.load_img(filepath, target_size=(224, 224))
test_img = image.img_to_array(test_img)
test_img = np.expand_dims(test_img, axis = 0)
test_img /= 255
return test_img

picture = &amp;amp;quot;marlon-brando.jpg&amp;amp;quot;
prediction = age_model.predict(loadImage(picture))
```

Pre variable stores distribution for each age class. Monitoring it might be inte

```
y_pos = np.arange(101)
plt.bar(y_pos, prediction[0], align=&amp;#039;center&amp;#039;,
plt.ylabel(&amp;#039;percentage&amp;#039;)
plt.title(&amp;#039;age&amp;#039;)
plt.show()
```

This is the age prediction distribution of Marlon Brando in Godfather. The most dominant age class is 44 whereas weighted age is 48 which is the exact age of him in 1972

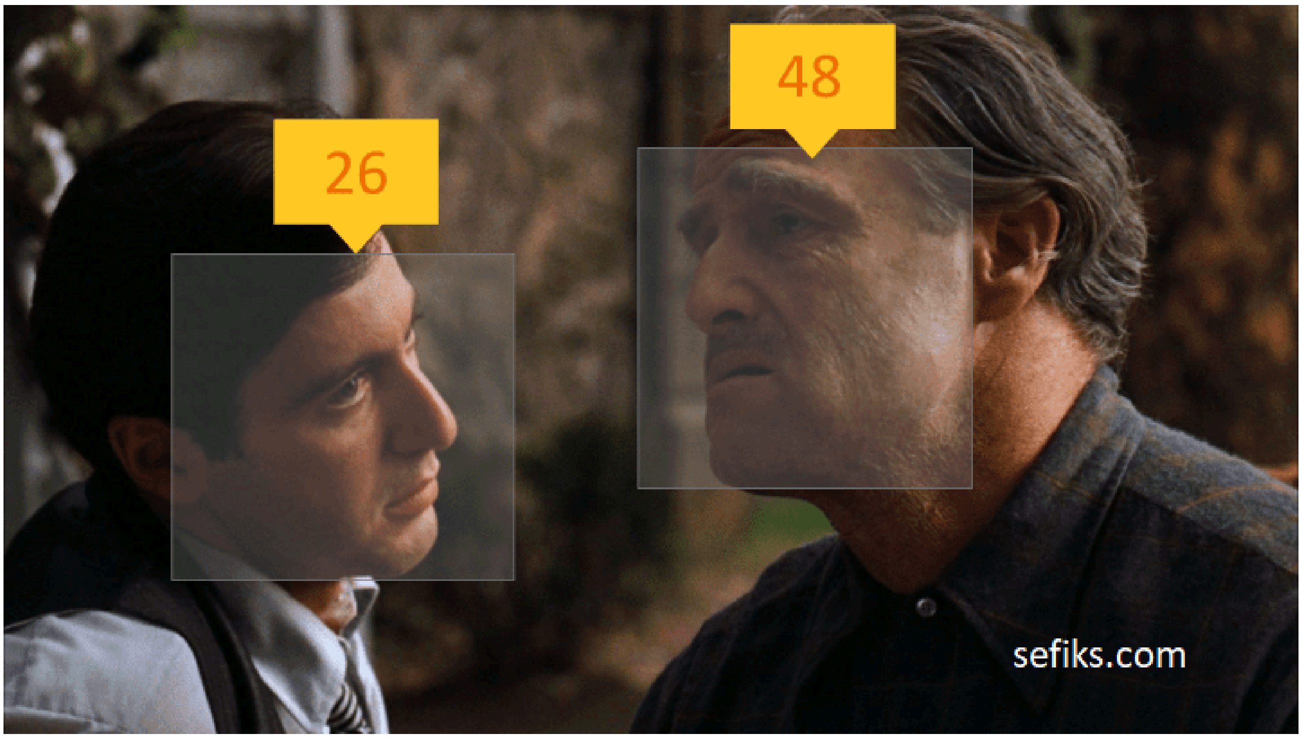


Age prediction distribution for Marlon Brando in Godfather

We'll calculate apparent age from these age distributions

```
img = image.load_img(picture)
plt.imshow(img)
plt.show()

print(&amp;amp;quot;most dominant age class (not apparent age):

apparent_age = np.round(np.sum(prediction * output_indexes, axi
print(&amp;amp;quot;apparent age: &amp;amp;quot;, int(apparent_a
```

Results are very satisfactory even though it does not have a good perspective. Marlon Brando was 48 and Al Pacino was 32 in Godfather Part I.



Apparent Age Prediction in Godfather

Compare to original study

As I mentioned before, we re-trained the base model because the original study is mainly based on Caffe and I need pre-trained weights for Keras. The original study was the winner of the ChaLearn Looking at People (LAP) challenge on Apparent age V1 (ICCV '15).

You are expected to predict the age of someone and there are several predictions of his/her age instead of actual age. So, your predictions will be evaluated by the mean and standard deviation the the jury predictions.

$$\mathcal{E} = 1 - e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

tion formula

If your prediction is equal to the mean of the predictions, then error becomes 0. Besides, if your prediction is not close to the mean of the predictions but the standard deviation of jury predictions are high, then the error closes to 0 as well.

On the other hand, you will be fined if your prediction is not close to the mean of predictions and the standard deviation of the jury predictions is low as well.

```
from math import e
df[&amp;#039;epsilon&amp;#039;] = e ** ( -1*( (df[&amp;#039;pre
df[&amp;#039;epsilon&amp;#039;].mean()
```

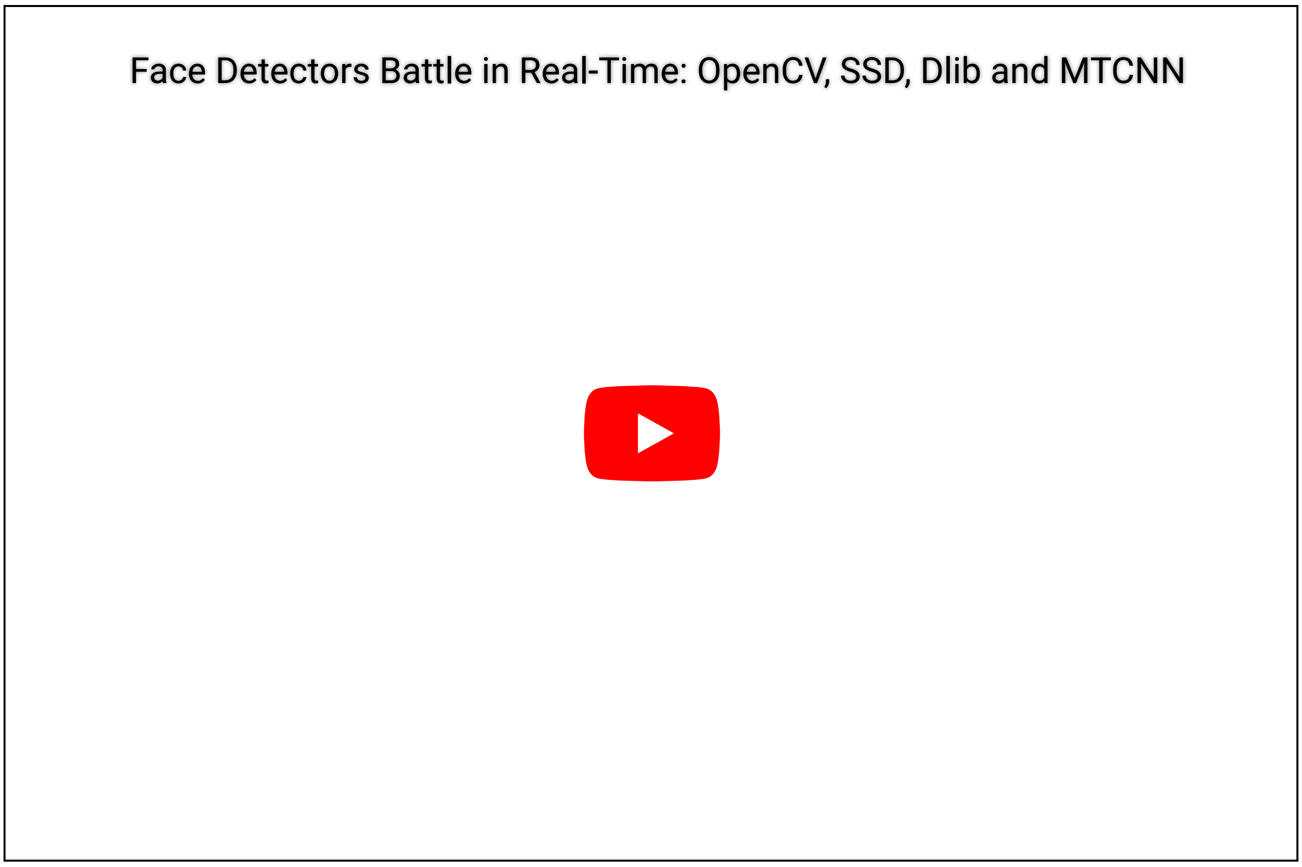
The ε value of this model is 0.387378, and MAE is 7.887859 for 1079 instances. On the other hand, the ε value of the original study was 0.264975. They declared that human reference of ε was 0.34. So, the original study is still little bit more accurate than the model I created in this post. Besides, my model is close to the human level for age prediction.

You can find the evaluation test data set and its labels here.

Face detection

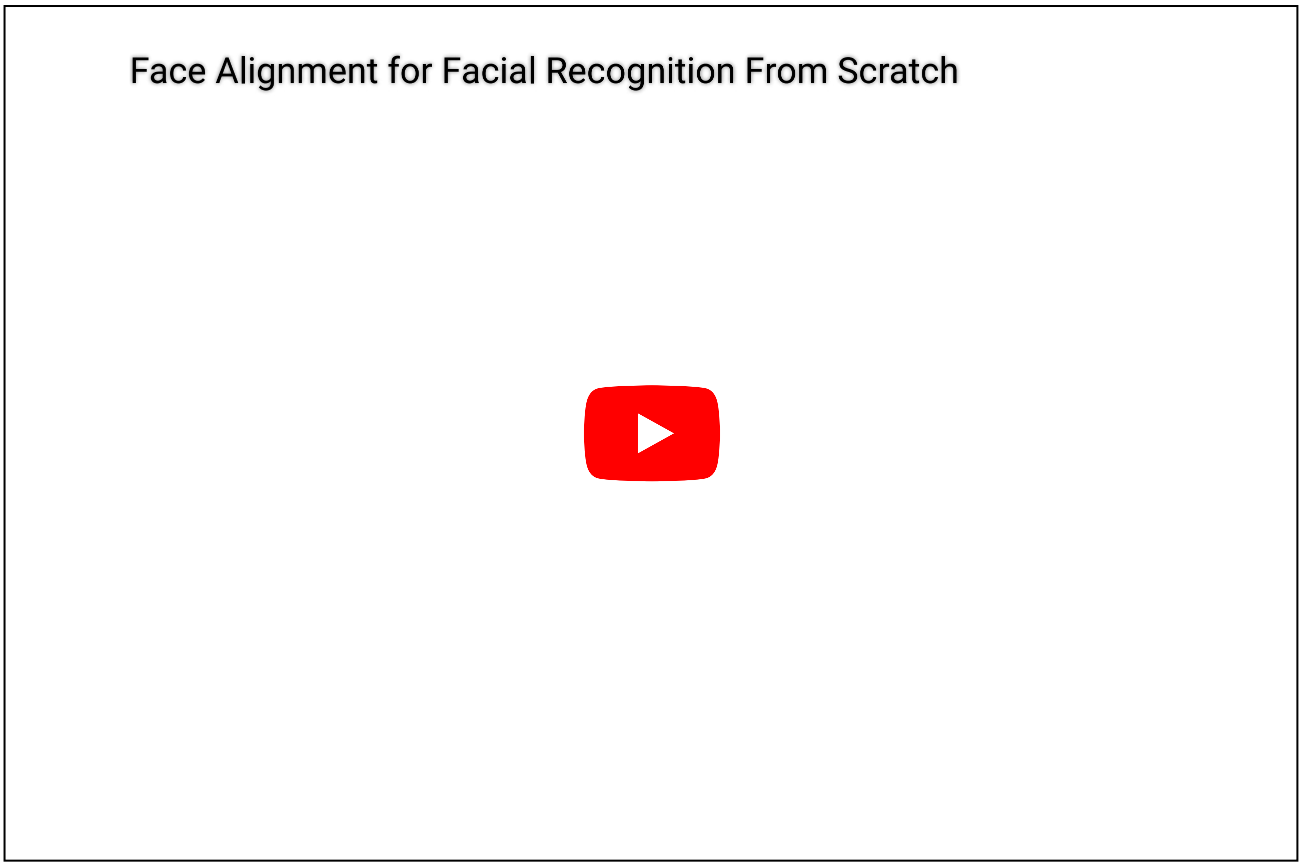
Train set images are already cropped and just facial areas are mentioned. Testing a custom image requires to detect faces. This will increase the accuracy dramatically. Besides, face alignment is not a must but it is a plus for this study.

There are several face detection solutions. OpenCV offers haar cascade and single shot multibox detector (SSD). Dlib offers Histogram of Oriented Gradients (HOG) and Max-Margin Object Detection (MMOD). Finally Multi-task Cascaded Convolutional Networks (MTCNN) is a common solution for face detection. Herein, haar cascade and HoG are legacy methods whereas SSD, MMOD and MTCNN are deep learning based modern solutions. You can see the face detection performance of those model in the following video.

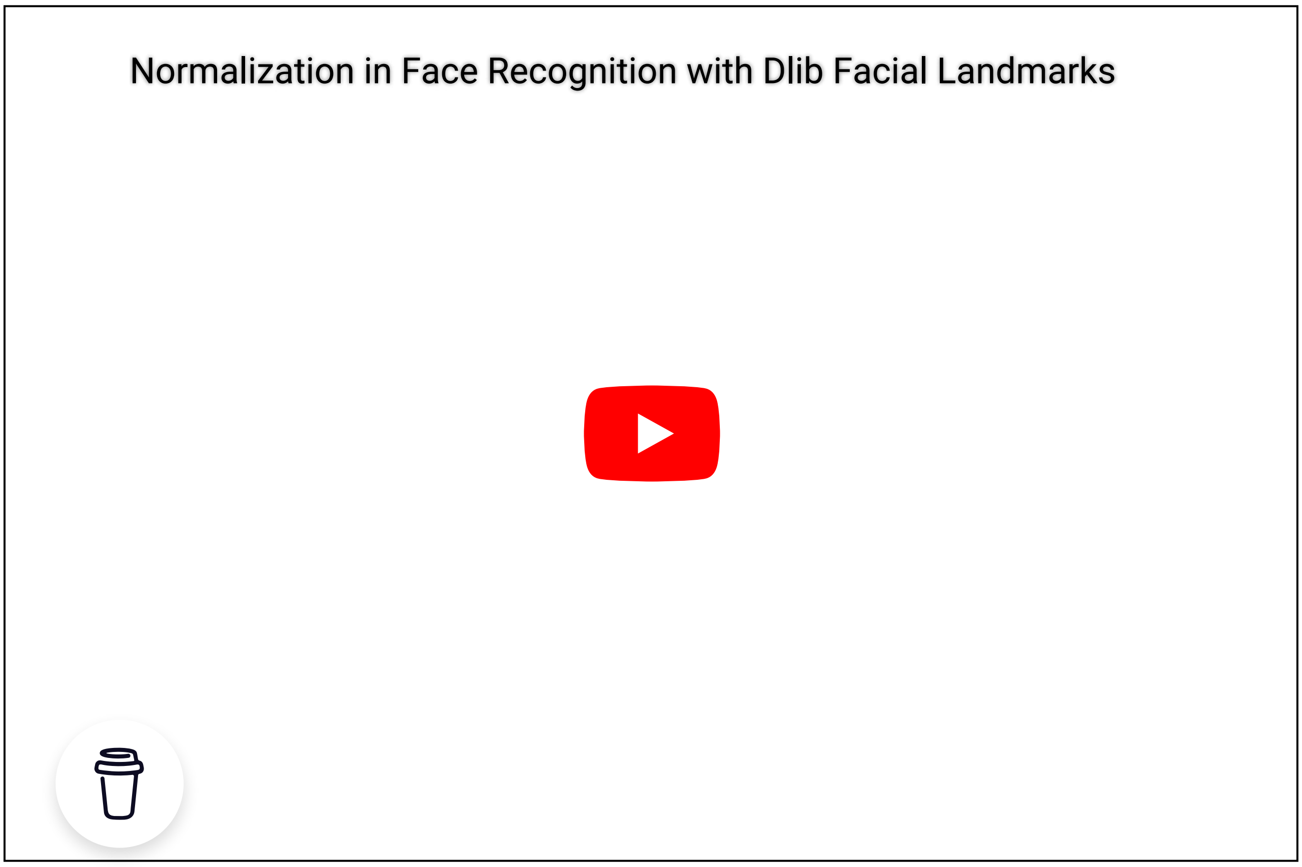

Face Detectors Battle in Real-Time: OpenCV, SSD, Dlib and MTCNN

Here, you can also see how to run those different face detectors in a single line of code with deepface framework for python.

You can find out the math behind face alignment more on the following video:



Face detectors extract faces in a rectangle area. So, it comes with a noise such as background color. Here, we can find 68 landmarks of a facial image with dlib.


Normalization in Face Recognition with Dlib Facial Landmarks

Here, retinaface is the cutting-edge face detection technology. It can even detect faces in the crowd and it finds facial landmarks including eye coordinates. That's why, its alignment score is very high.

InsightFace: RetinaFace and ArcFace for Facial Recognition in Python

Gender prediction model

Apparent age prediction was a challenging problem. However, gender prediction is much more predictable.

We’ll apply binary encoding to target gender class.

```
target = df[&amp;#039;gender&amp;#039;].values
target_classes = keras.utils.to_categorical(target, 2)
```

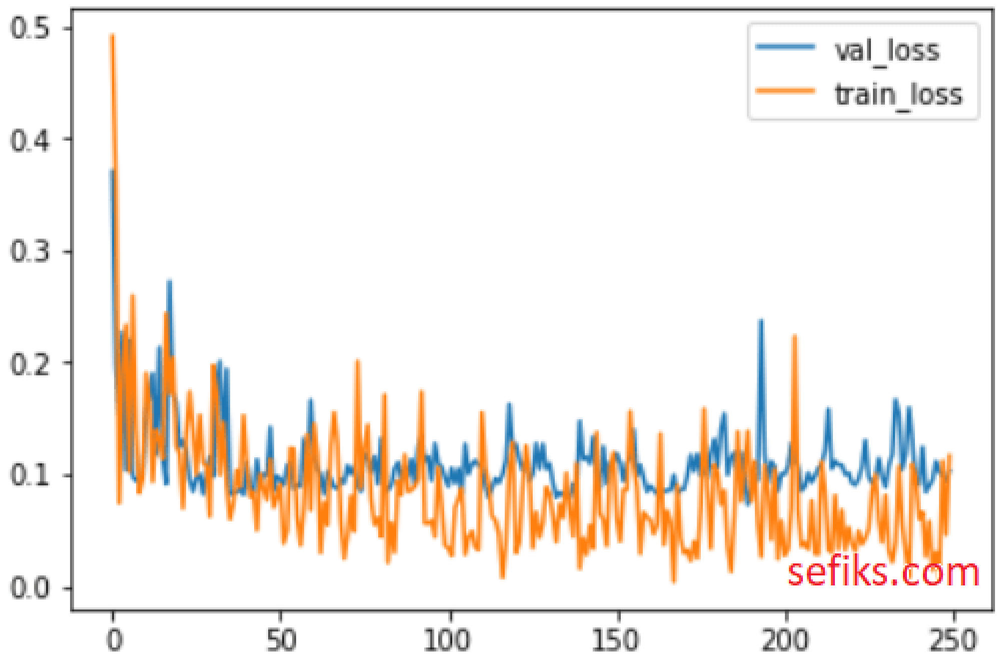
We then just need to put 2 classes in the output layer for man and woman.

```
    layer in model.layers[:-7]:
    r.trainable = False

base_model_output = Sequential()
base_model_output = Convolution2D(2, (1, 1), name=&amp;#039;prec
base_model_output = Flatten()(base_model_output)
base_model_output = Activation(&amp;#039;softmax&amp;#039;)(base

gender_model = Model(inputs=model.input, outputs=base_model_out
```

Now, the model is ready to fit.

```
scores = []
epochs = 250; batch_size = 256

for i in range(epochs):
print(&amp;amp;quot;epoch &amp;amp;quot;,i)

ix_train = np.random.choice(train_x.shape[0], size=batch_size)

score = gender_model.fit(train_x[ix_train], train_y[ix_train]
, epochs=1, validation_data=(test_x, test_y), callbacks=[check

scores.append(score)
```

It seems that the model is saturated. Terminating training will be clever.



Loss for gender prediction

Evaluation

```
ender_model.evaluate(test_x, test_y, verbose=1)
```

The has the following validation loss and accuracy. It is really satisfactory.

[0.07324957040103375, 0.9744245524655362]

Confusion matrix

This is a real classification problem instead of age prediction. The accuracy should not be the only metric we need to monitor. Precision and recall should also be checked.

```
from sklearn.metrics import classification_report, confusion_ma

predictions = gender_model.predict(test_x)

pred_list = []; actual_list = []

for i in predictions:
pred_list.append(np.argmax(i))

for i in test_y:
actual_list.append(np.argmax(i))

confusion_matrix(actual_list, pred_list)
```

The model generates the following confusion matrix. Columns are prediction whereas rows are actual value labels.

	Female	Male
Female	1873	98
Male	72	4604

This means that we have 96.29% precision, 95.05% recall. These metrics are as satisfactory as the accuracy.

Testing gender for custom images

We just need to feed images to the model.

```
picture = &amp;amp;quot;katy-perry.jpg&amp;amp;quot;
prediction = gender_model.predict(loadImage(picture))

[illegible] = image.load_img(picture)#, target_size=(224, 224))
[illegible]imshow(img)
[illegible].show()
gender = &amp;amp;quot;Male&amp;amp;quot; if np.argmax(predicti
print(&amp;amp;quot;gender: &amp;amp;quot;, gender)
```

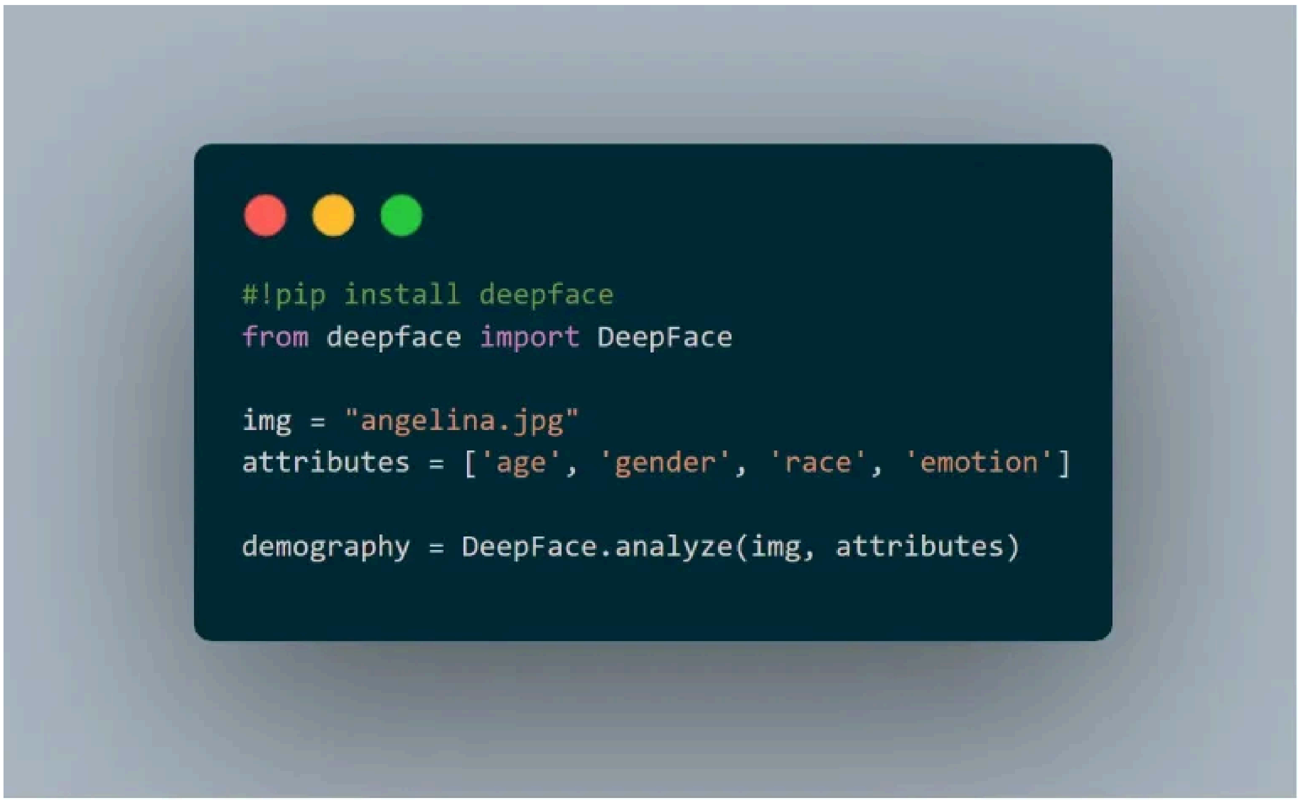
Conclusion

So, we’ve built an apparent age and gender predictors from scratch based on the research article of computer vision group of ETH Zurich. In particular, the way they proposed to calculate apparent age is an over-performing novel method. Deep learning really has a limitless power for learning.

I pushed the source code for both apparent age prediction and gender prediction to GitHub. Similarly, real time age and gender prediction implementation is pushed here. You might want to just use pre-trained weights. I put pre-trained weights for age and gender tasks to Google Drive.

Python library

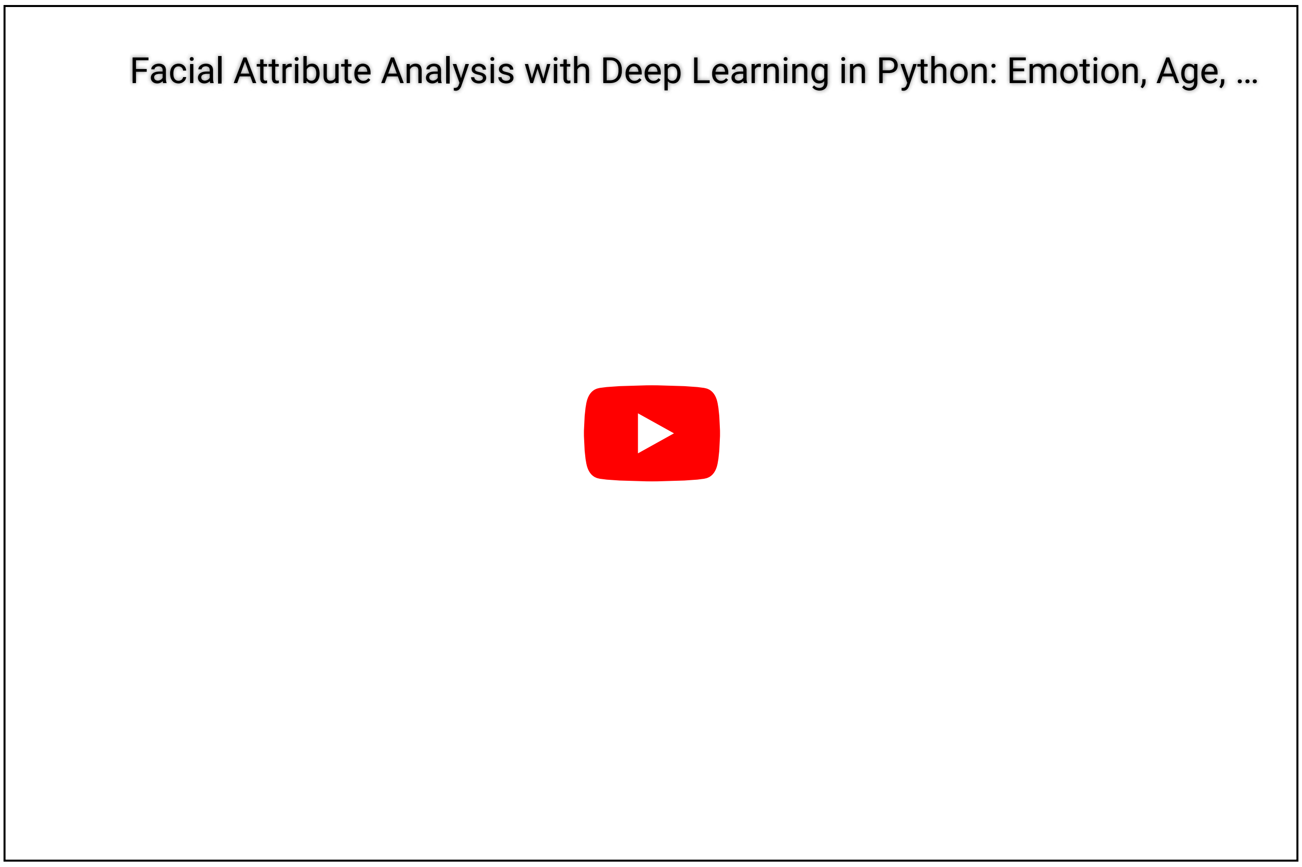
Herein, deepface is a lightweight facial analysis framework covering both face recognition and demography such as age, gender, race and emotion. If you are not interested in building neural networks models from scratch, then you might adopt deepface. It is fully open-source and available on PyPI. You can make predictions with a few lines of code.



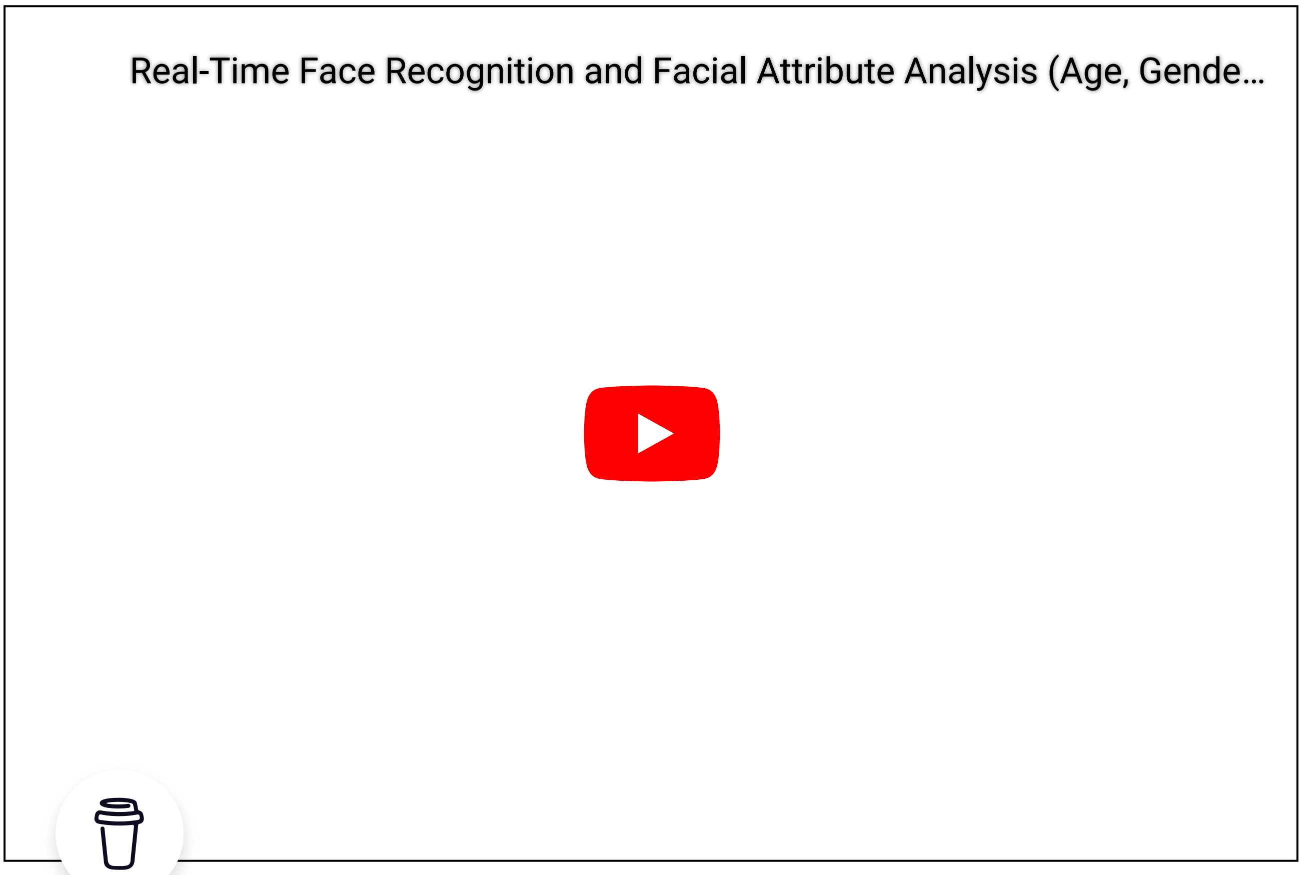
```python
#!pip install deepface
from deepface import DeepFace

img = "angelina.jpg"
attributes = ['age', 'gender', 'race', 'emotion']

demography = DeepFace.analyze(img, attributes)
```

Deep [illegible] Analysis

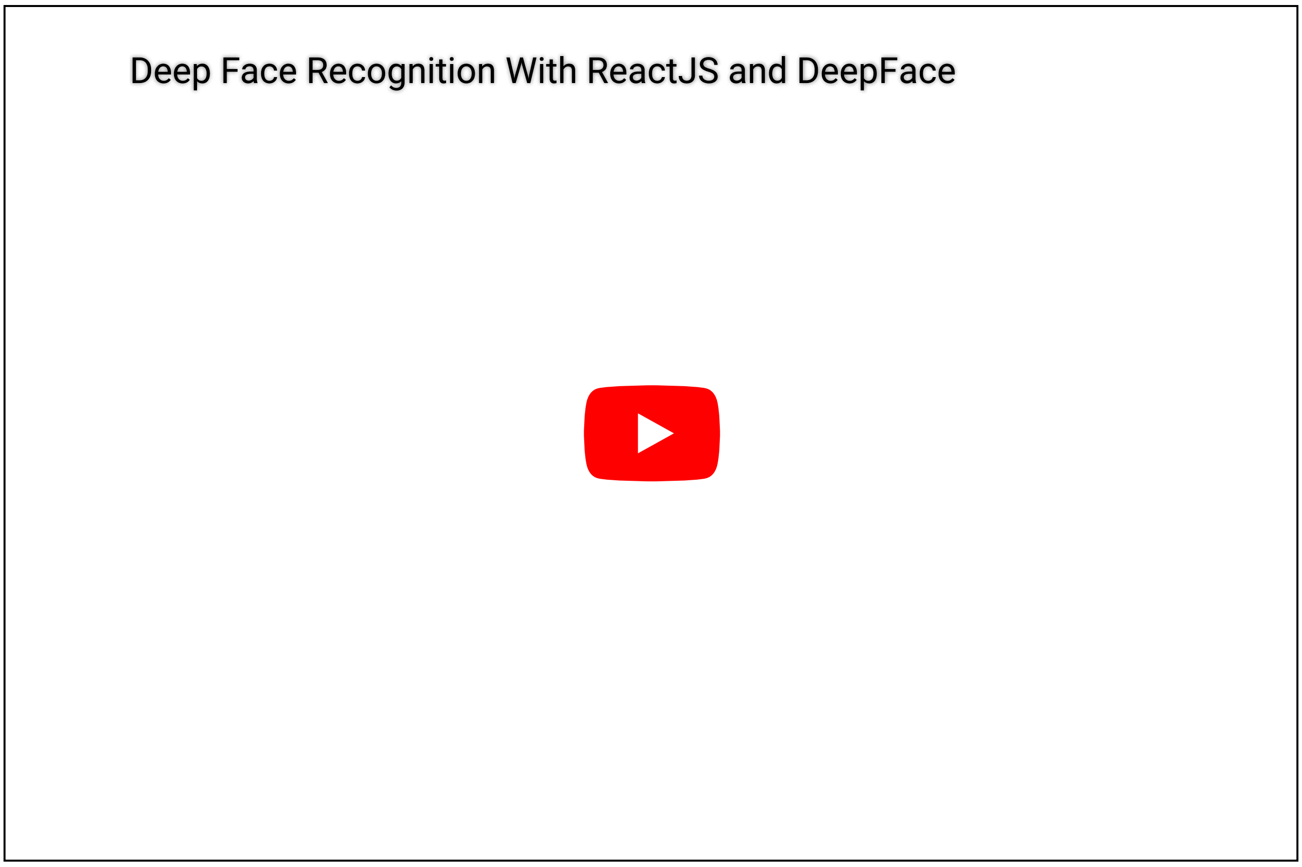
Here [illegible] can watch how to apply facial attribute analysis in python with a just few lines of code.


Facial Attribute Analysis with Deep Learning in Python: Emotion, Age, ...

You can run deepface in real time with your web cam as well.


Real-Time Face Recognition and Facial Attribute Analysis (Age, Gende...

Meanwhile, you can run face verification tasks directly in your browser with its custom ui built with ReactJS.

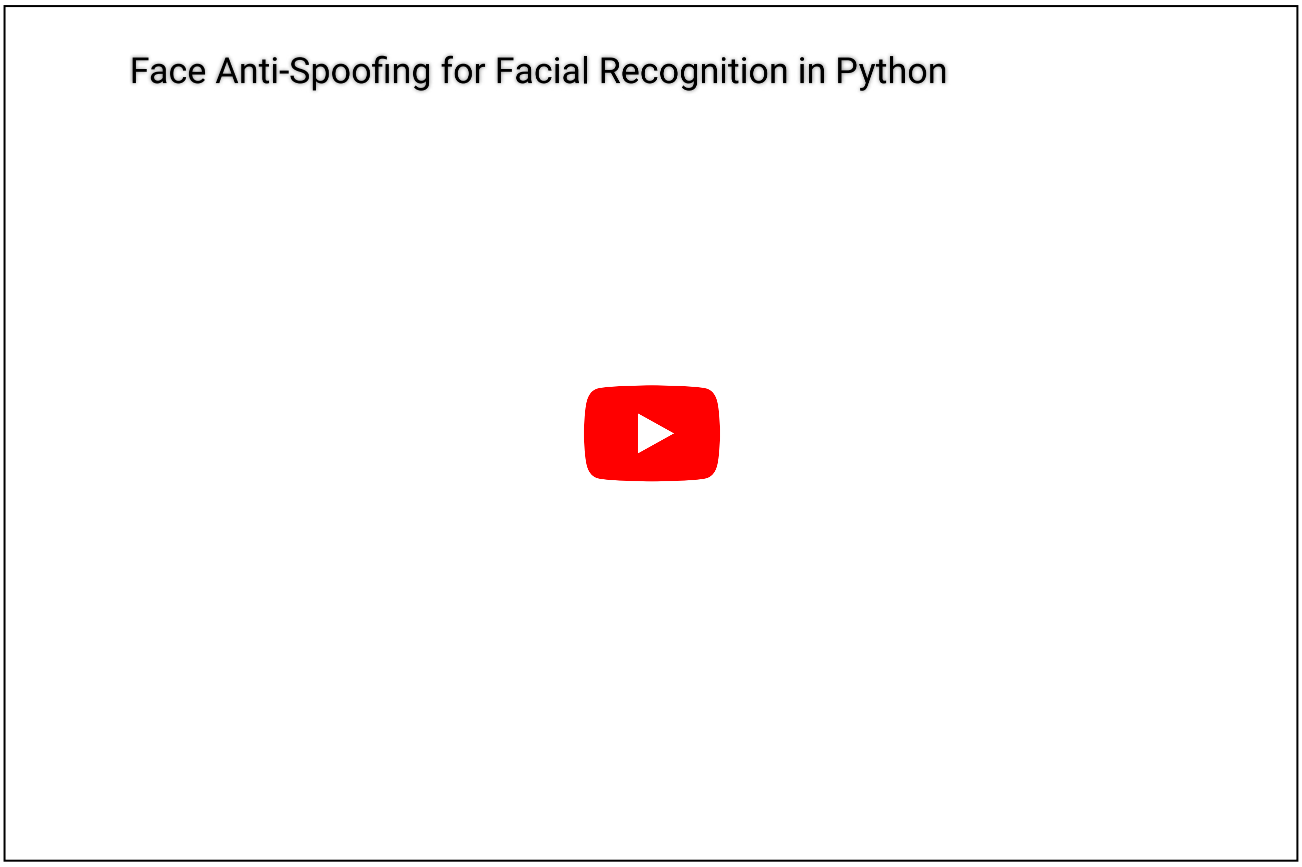

Deep Face Recognition With ReactJS and DeepFace

Also, deepface has its React JS ui for facial attribute analysis purposes.

Face Analysis with ReactJS & DeepFace: Age, Gender, Emotion, Race

Anti-Spoofing and Liveness Detection

What if DeepFace is given fake or spoofed images? This becomes a serious issue if it is used in a security system. To address this, DeepFace includes an anti-spoofing feature for face verification or liveness detection.



#age prediction, #deep learning, #gender prediction, #keras, #python, #vgg

« Previous / Next »

83 Comments

riccardo di marcantonio

April 6, 2019 at 7:30 am

Great article!
I tried the code and the results are really good.
I have verified, however, that even by submitting an image that does not contain a face, the model returns a prediction of age and gender. Is there a way to detect images where there is no face?

Sefik Serengil
April 6, 2019 at 7:51 am

First of all, thank you for your feedback.

Yes, you are right. Current implementation always returns a prediction. You can detect face first and apply age/gender prediction to detected faces. I have used OpenCV's haarcascade module to detect faces. You can find a similar implementation here: https://sefiks.com/2018/01/10/real-time-facial-expression-recognition-on-streaming-data/ . In this case, I detect faces and apply emotion prediction instead of age/gender. I think you can easily adapt to this problem.

riccardo di marcantonio
April 6, 2019 at 10:06 am

Dear Sefik,
thanks for your kind reply.
I will try your example immediately.

Sefik Serengil
April 10, 2019 at 2:00 pm

BTW, I forget that I already implement this. Please look this repo
ps://github.com/serengil/tensorflow-101/blob/master/python/age-
er-prediction-real-time.py

riccardo di marcantonio
April 10, 2019 at 8:14 pm

thanks Sefik!
I'm doing tests and everything seems to work very well!
I wanted to ask you if you have any suggestions to increase the recall in the face recognition process and in the age detection process.
I am currently using the pre-trained Wikipedia-based model. Do you think that if I used IMDB (7GB) the recall of the age detection process would improve?

Sefik Serengil
April 10, 2019 at 8:16 pm

I actually trained the model with both imdb and wiki data set. Currently, the model can predict with error plus minus 4 ages.

You might re-train with an alternative model such as inception v3 or regular vgg to increase the accuracy.

kalz
May 31, 2020 at 7:32 am

Hi Sefik, when i used the imdb dataset, the jupyter notebook and colab(with GPU) crashes whenever i loop through the observations to get the pixels. My guess is that its a memory issue. do you know any way i can get around it?

Sefik Serengil
May 31, 2020 at 12:14 pm

Even though colab offers you GPU, it has a limited memory. I've run this study on my local environment and I do not have a memory problem. I recommend you to decrease the size of the data set.

Poone
July 15, 2019 at 2:44 pm

HI sefik
im poone and computer student
i search for AGE AND GENDER ESTIMATION and find your project
I copy and paste and run your project and it doesent work for me

https://sefiks.com/2019/02/13/apparent-age-and-gender-prediction-in-keras/
Could you send your python script this project for me pllllssssss
Poone.qbni@gmail.com

Sefik Serengil
July 15, 2019 at 2:45 pm

That post directs you a github repo. Could you try the iPython notebook in there?

Arun
April 15, 2019 at 11:12 am

I followed the same steps, but got an accuracy of 0.5% [3.497109861968012, 0.057720696795011364], kindly help me

Sefik Serengil
April 15, 2019 at 11:15 am

This is for gender prediction? If so, 1st index value is greater than 2nd index. This means gender would be woman.

Arun
April 16, 2019 at 6:46 am

This for Age prediction i followed the same github https://github.com/serengil/tensorflow-101/blob/master/python/apparent_age_prediction.ipynb.
After the model is trained when model is evaluated on test set the github page says accuracy is
6774/6774 [==============================] – 17s 2ms/step
Out [illegible]
[2.8[illegible]90848929, 0.24298789490543357].

For me it is 6774/6774 [==============================] – 28s 4ms/step
[3.493283847809401, 0.056834957193652544].

I really love your tutorial, please help me to achieve the same accuracy.

Sefik Serengil
April 16, 2019 at 7:12 am

Oke I understand what you mean right now. First of all, I trained the model with both wiki and imdb data set. The more data brings the more accurcy. Secondly, you cannot get same accuracy because of random initialization. Why you need to get same accuracy? You might use same pre-trained weight and by-pass traininig.

Arun
April 16, 2019 at 8:43 am

Sorry i mean you got 24%, whereas i got only 0.5%. So you mean i should also train IMDB dataset along with Wiki dataset? for better accuracy?

Sefik Serengil
April 16, 2019 at 8:45 am

You should focus on loss value because we will not approach this problem as classification problem. The following steps will calculate weighted ages by multiplying age label and its probability.

Jake
May 21, 2019 at 10:39 pm

Firstly, thanks for this guide. I have a question, how could we predict first some age groups and then the age for improving the model?

Sefik Serengil
May 22, 2019 at 5:47 am

Suppose that you want to classify ages in 3 classes: young, middle age and old. You can add the following code block after the 14th block in https://github.com/serengil/tensorflow-101/blob/master/python/apparent_age_prediction.ipynb

#add after 14th block
df[df[df['age'] >= 50].index, 'age_class'] = 'old'
df[df[(df['age'] < 50) & (df['age'] >= 30)].index, 'age_class'] = 'middle_age'
df[df[df['age'] < 30].index, 'age_class'] = 'young' df['age'] = df['age_class'] df = df.drop(columns=['age_class']) But I do not know this approach can reach the accuracy level of the already implemented one.

James
May 25, 2019 at 5:40 pm

I am trying to implement this in resnet50. However, the model does not predict well on older people. How could I increase this?

I tried to retrain multiple times with different datasets, But when I do this, the MAE and loss are not going to be improved it starts from different higher values then the first network was trained on. It looks that the learned weights are forgotten.

Sefik Serengil
May 26, 2019 at 6:47 am

1- Do you freeze the early layers with trainable = False command?
2- Ignore mae and loss because we finally calculate the weighted ages. I mean that we would not get the highest score age, each age score will be multiplied with its label.
3- Resnet50 is designed for object recognition. But Face version of VGG is designed for face recognition. This means that model detects face oriented features in early layer. That's why, VGG would overperform most probably.

James
May 27, 2019 at 7:35 am

1. I freeze all layers
2. For evaluating, I checked on MAE on different datasets. For some, I achieve between 4,80 – 7.50 MAE. However older People around 70 old are years are

predicted mostly around 50-60.
3. Good to know, didnt knew it.

Is it normal that age estimation works better at younger people than 60+ ages?

Sefik Serengil
May 27, 2019 at 7:37 am

Please freeze just early layers. I mostly freeze all layers except the last 4 convolution layer. In this way, pre-trained model detect some patterns in early layers and my fully connected layers can find relation between detected patterns and my custom problem (in this case age and gender)

Veeru
November 6, 2019 at 1:50 pm

for layer in model.layers[:-7]:
layer.trainable = False
The above is some thing like freezing from 7 to prediction layers.

Wondering is this right to freeze the layers. I think it should be as below to freeze the first few layers (from input to 7 layers)
for layer in model.layers[:7]:
layer.trainable = False

is my understanding wrong?

Sefik Serengil
November 6, 2019 at 1:54 pm

model.layers[:-7]: means freeze all layers except last 7. I mean 7 on the right. Early layers freezed.

Harsh Tamakuwala
June 25, 2019 at 4:34 pm

Thanks, the tutorial is pretty nice but I am having memory issues. I tried to train the model on Google colab too without using the weights but just after 5 iterations of batch size 128 and that too from first 1000 images the GPU memory runs out. I wanted to ask if you implemented some other code for memory runout???

Sefik Serengil
June 25, 2019 at 4:40 pm

In the In [112] (enableFit) it feeds random 256 image in every iteration. This should solve the memory issue. Larger epoch, smaller batch size you need.
Repo: https://github.com/serengil/tensorflow-101/blob/master/python/apparent_age_prediction.ipynb

---

Harsh Tamakuwala
June 25, 2019 at 4:42 pm

Yes, I have kept it to 5 epochs and batch_size to 128 but the memory seems to overflow

---

Sefik Serengil
June 25, 2019 at 4:43 pm

What about 64 or 32 batch size?

Harsh Tamakuwala
June 25, 2019 at 4:49 pm

I have a doubt the load_model and save_weights line in the github repo for age prediction is outside the loop while for gender prediction it is inside the loop.
eping the lines outside the loop will too restore best weights or it ould be inside only?? Because, I think it is creating the memory problem…

Sefik Serengil
June 25, 2019 at 4:53 pm

They should both be outside of for loop

Harsh Tamakuwala
June 25, 2019 at 4:56 pm

Thanks, this seems to solve the memory issue but did you train for all wiki images at once?

Sefik Serengil
June 25, 2019 at 5:10 pm

Nope, I fed as batches. ix_train stores imdb images length of batch size. Every iteration it is stored randomly in all imdb data set.

Harsh Tamakuwala
June 25, 2019 at 5:13 pm

ok while storing in np array above the training the memory runs out…
During this line:
df['pixels'] = df['full_path'].apply(getImagePixels)
Here, I am feeding only 2000 images of the dataset which seems to be hindering the accuracy and the loss because of insufficient amount of images.
Did you load the complete dataset?

Sefik Serengil
June 25, 2019 at 5:15 pm

las data frame stores all imdb data set

Harsh Tamakuwala
June 25, 2019 at 5:18 pm

Dataframe stores the path right? And then we load it using np amd store pixel values but these values and seems to cause memory error if I use all of the images… Hence, I am using only the first 2000 images and am getting 5% accuracy…

Sefik Serengil
June 25, 2019 at 5:20 pm

Raw data frame stores path but then we load it as pixels. You might reload pixels in the for loop to reduce memory allocation. Because data frame size is low without pixels.

Harsh Tamakuwala
June 28, 2019 at 12:58 pm

Thanks I was able to increase the iterations…

useless
April 6, 2022 at 8:05 am

Can you let me know how you did that?

---

krishna vamshi
June 28, 2019 at 11:19 am

how to train the model?
is the gpu necessary?

Sefik Serengil
June 28, 2019 at 12:05 pm

N[illegible]s not a must. But it speeds you up radically.

Krishna Vamshi

July 1, 2019 at 9:38 am

just changing enablefit to true is enough?
what about agemodel (hdf5) ? should i download it or it will be created on process(if it doesn't exist already)?

Sefik Serengil
July 1, 2019 at 10:10 am

You must download and put hdf5 file in the same directory, then set enableFit param to true. Reference link of weight file exists in the post.

Krishna Vamshi
July 2, 2019 at 8:55 am

hdf5 is a checkpoint model file. right?
I have already downloaded weights (h5 files)
I am hoping it will create a checkpoint model on the process, am I wrong?

Sefik Serengil
July 2, 2019 at 9:25 am

Correction: it should be h5, not hdf5. If you installed weights file, that's enough.

Harsh Tamakuwala
June 28, 2019 at 12:59 pm

If you don't have a system with good GPU, then try to implement it on google colab which has pretty good processing power on GPU

---

aurav Goyal
June 29, 2019 at 8:35 am

Great article. I have one question though. On every epoch, after calculating the weights for current epoch, model validates loss on all validation_data(xtest, ytest)

?
If yes, Is there any way, to perform same in batches like we pass the xtrain in batches?

I seem to have issues of memory overflow for all xtest (50000+) images loaded in numpy array.

Sefik Serengil
June 29, 2019 at 8:42 am

Validation performs on all test data. Instead of applying batches, you should select a smaller size sub set of test data and validate it. In this way, you can compare your loss with same conditions.

Gaurav Goyal
June 29, 2019 at 8:45 am

Okay. But still, is there any way to perform validation step separately in a loop?

Sefik Serengil
June 29, 2019 at 8:46 am

To avoid overfitting, it should be in for loop.

---

feras
November 17, 2019 at 2:46 pm

Hi Sefik,
thank you for your great article!
I have a problem while getting the pixels values from the full_path.
It se at PIL can't find the path of each image inside the full_path column.

df['pixels'] = df['full_path'].apply(getImagePixels)

FileNotFoundError: [Errno 2] No such file or directory: 'wiki_crop/17/10000217_1981-05-05_2009.jpg'

Best Regards

Sefik Serengil
November 17, 2019 at 2:54 pm

wiki_crop folder is in the same directory of your notebook?

Kan John
March 25, 2021 at 4:09 pm

Thank you for your amazing article
I have exactly the same problem, I am sure that wiki_crop folder in the same directory of my notebook? Could you please help me to solve the problem?
Thanks.

Bram
December 9, 2019 at 11:18 am

Hi Sefik,

First of all, thanks for publishing this tutorial and the pre-trained weights.
I have a problem when trying to implement the face model. It gives an error for the conv2d layer (7,7), it says 'negative dimension caused by subtracting 7 from 3…'.
Did you occur a similar problem, or have a suggestion how to avoid this error?

Sefik Serengil
December 9, 2019 at 11:30 am

What is your tensorflow and keras version? I tested it on TensorFlow 1.9.0 and Keras 2.2.0.

Bram
December 9, 2019 at 1:29 pm

I'm using Tensoflow version 1.15 and Keras version 2.2.5 (Google Colab)

Sefik Serengil
December 9, 2019 at 1:31 pm

Could you try it same version of my environment? I could not solve this issue for that environment soon.

Farah
December 14, 2019 at 2:10 pm

Hi, thank you for being in your block and a dimension of the information you have provided to us. I'm constantly taking advantage of your blocks in my classes. I would really appreciate it if you help me I'm constantly getting error in your printer's real-time prediction, and the other code may be the librarian who can ever walk away but what I'm doing is missing something.

Sefik Serengil
December 15, 2019 at 8:38 am

What exactly the trouble you have?

Clement
March 30, 2020 at 5:46 pm

Hello Sefik,
First of all thank you for this great presentation and word, it has truly helped me working on a similar project!
I used imdb + wiki, cropped faces using Haarcascade (resulting in a 200k faces approximately), trained my model using transfer learning as well, my score is pretty similar, but i can't explain the weird difference:
[2.86, 24%], but an MAE of 6.8, using your multiplication technique.
How [illegible]'d you explain this?

Sefik Serengil
March 31, 2020 at 5:45 am

The original idea is not mine. I applied some approach in the original paper as I mentioned. Herein, age prediction is not a classification problem. If you evaluate your predictor as classifier, then you count inaccurate 1 or 2 year deviations. On the other hand, this is very satisfactory prediction even for people. Researchers find a robust way to transform a classification problem to a regression problem.

They might build a neural networks with a single output node. In this way, they can build a regressor but I think this approach shows more accurate results.

Clement
March 31, 2020 at 12:36 pm

Ok yes apparently it does.
And what do you think about the fact that the data is highly unbalanced? I feel like the model tends to predict a lot between 30 and 50, and has trouble predicting young people age.
What would you suggest to fight that?

Sefik Serengil
March 31, 2020 at 1:26 pm

The only solution is to increase the sample data. You are absolutely right. Because I tested the model on kids and kids less than 7 years old seems 20 years old based on the model prediction 🙄

Ruihan
April 3, 2020 at 6:37 pm

Thank you and it is a great article to follow.

I am working on the age estimation. One thing I noticed is that the data in your training loss vs validation loss plot is different from the final result you showed. You ved around 3.2-3.6 validation error but in your result you are showing the on error is around 2.8 and achieved 24% accuracy for age estimation.

You mentioned that you actually used both wiki and imdb dataset to achieve this. But I used both datasets as well and final achieved around 3.2 for validation error and 12% accuracy for age estimation and MAE around 6.

I was wondering could you explain exactly what you changed here for the code to achieve higher accuracy and lower MAE. Do not need to be the same but at least I would like to be closer to your results. Thank you!

Sefik Serengil
April 3, 2020 at 7:52 pm

I just combine wiki and imdb data sets and applied batch learning. Even I cannot get same score if I re-train the model from scratch. Luckily, I stored the pre-trained weights to re-use.

Ruihan
April 3, 2020 at 9:02 pm

Thanks for your reply Sefik. It's normal that there will be a bit variation since each epoch is different and assigned images randomly. But there must be a range of variation, if anything between 20%-28% for accuracy, I guess it could be caused by variation. But I can only achieve 12% for accuracy, what would be your 'normal' accuracy rate and any suggestion how to improve? Would the change of optimizer to sgd instead of adam work or maybe another pre-train model such as inception V3 instead of vgg?

Sefik Serengil
April 4, 2020 at 12:13 pm

Nope. The model was VGG-Face and optimizer was Adam but you might try to use SGD. Of course you should spend more time in this case.

Ruhan
May 4, 2020 at 7:14 am

wh[illegible] assification _age _model.hdf5 and classification _age _model.hdf5 plea[illegible] me because without it this code is not run

Sefik Serengil

May 4, 2020 at 7:17 am

You can find links in the conclusion section of the post.

Kalz
May 25, 2020 at 7:37 am

Thanks for this great work, i highly appreciate. When i use the imdb dataset, the full path is in this format [01/nm0000001_rm124825600_1899-5-10_1968.jpg], im unable to get image pixels from this file format. thanks in advance

Sefik Serengil
May 25, 2020 at 7:46 am

The program you are running in imdb_crop folder? I mean that the both 01 folder and your program have to be in same folder.

Valery
October 9, 2020 at 8:45 am

Hi, Sefik!

Thanks a lot for your post. He very clearly and consistently explains the solution to the problem of determining age from a photo. Perhaps this is the best thing that can be found on the net on the topic of age determination.

I would like to expand the range of age definitions towards 0-15 years. I have a dataset with pictures with these ages. How can I train your model on this dataset?

Sefik Serengil
October 9, 2020 at 12:53 pm

T u firstly. You should apply the procedures mentioned in this post. In ot ords, you need to retrain it from scratch.

Valery

October 9, 2020 at 1:55 pm

Thanks. How long did it take you to train the neural network? And what equipment was used, if not a secret?

Sefik Serengil
October 9, 2020 at 2:27 pm

1-2 hours with a basic gpu

Valery
October 9, 2020 at 2:43 pm

Very good! I will look forward to new posts)

Valery
November 6, 2020 at 11:39 am

Hi, Sefik!
Another question arose. In the section of the script with the preparation of input data for training the network, there is a commented out line
#img = preprocess_input (img).
Shouldn't you uncomment this line? You use the VGG-Face model as a basis. She trained images in which pixels are normalized in the range [-1, +1] (https://sefiks.com/2018/08/06/deep-face-recognition-with-keras/). Why are pixels in this article normalized from 0 to 1?

Sefik Serengil
November 9, 2020 at 7:51 am

The both are true. In my experiments, normalizing in the range [0, 1] returns m[illegible]ust results but if you normalize it in the range [-1, +1] then it will still w[illegible]

Pasindu
December 11, 2020 at 2:04 am

I went through all the videos ,tutorials related to your Deepface project .You have explained all the things step by step very clearly and this is by far the best and comprehensive guide in the web for face delection/recognition i could find You have done an amazing work for all the students and to the open source community friend, . Really amazing .Keep up the good work

---

Pingback: DeepFace - Most Popular Deep Face Recognition in 2021 (Guide) | viso.ai

---

Ahmed Abouelnas
November 26, 2021 at 10:09 pm

I get this error when I'm trying to run…

AttributeError: module 'tensorflow.compat.v2.__internal__' has no attribute 'register_clear_session_function'

Sefik Serengil
November 27, 2021 at 8:55 am

Seems that your tf installation not okay

Zaki
February 5, 2023 at 9:44 pm

Hi, Thanks for the amazing tutorial..

The link to the pre-trained weight is broken, could you pin point where can I download it?

Best

Sefik Serengil
February 7, 2023 at 10:33 pm

Fixing it. Thank you.

Comments are closed.

Please cite this post if it helps your research. Here is an example of BibTex entry:

```
@misc{sefiks13053,
  author ={Serengil, Sefik Ilkin},
  title = { Apparent Age and Gender Prediction in Keras },
  howpublished = { https://sefiks.com/2019/02/13/apparent-age-and-gender-prediction-in-keras/ },
  year = { 2019 },
  note = "[Online; accessed 2025-01-01]"
}
```