Facial-Recognition-v0.1-MO-2020-3-6

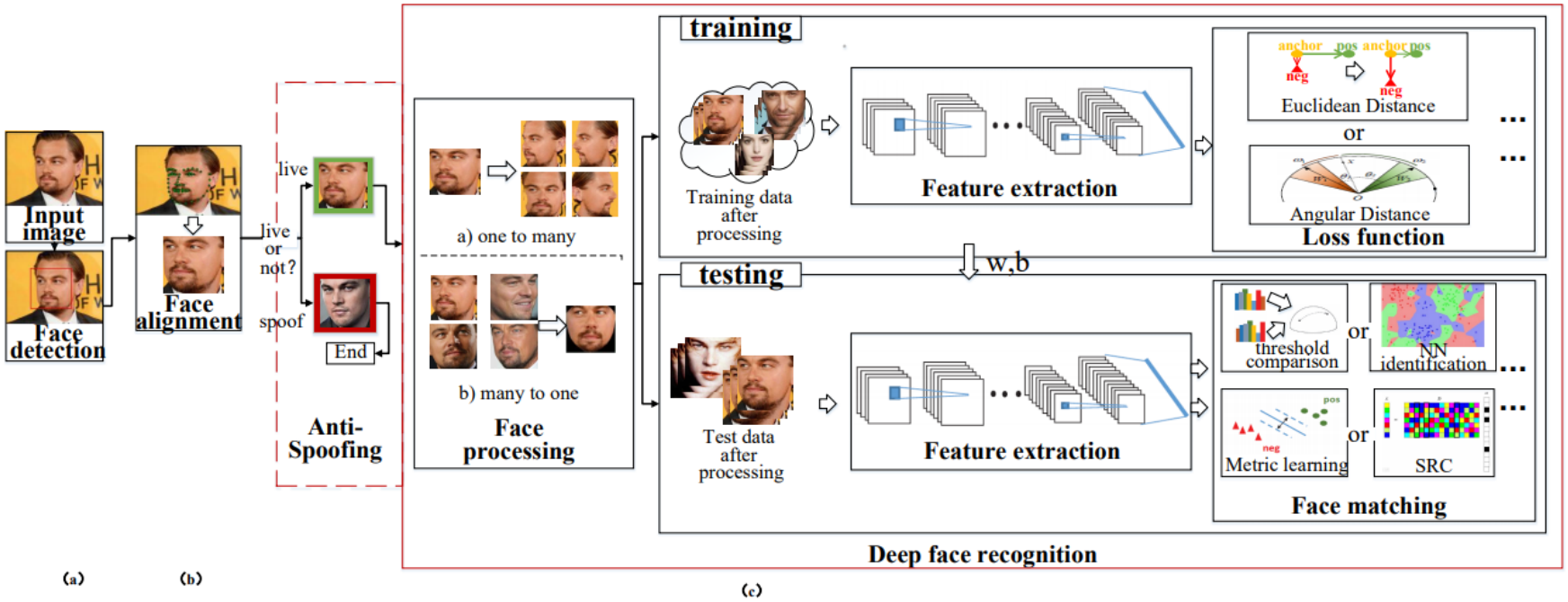
Version: x0.1 (Alpha)
Date: Mar 6, 2020
Project Lead: Harry Li, Ph.D.

Team members:
Minh Duc Ong

Deep Face

https://www.robots.ox.ac.uk/~vgg/publications/2015/Parkhi15/parkhi15.pdf
https://github.com/krasserm/face-recognition
https://en.wikipedia.org/wiki/DeepFace

"DeepFace is a deep learning facial recognition system created by a research group at Facebook. It identifies human faces in digital images. It employs a nine-layer neural network with over 120 million connection weights, organized as a siamese network, and was trained on four million images uploaded by Facebook users.[1][2] DeepFace shows human-level performance. The Facebook Research team has stated that the DeepFace method reaches an accuracy of 97.35% ± 0.25% where human beings have 97.53% [3]. This means that DeepFace is sometimes more successful than the human beings. It also leaves behind the FBI's Next Generation Identification system which have 85% performance [4] One of the creators of the software, Yaniv Taigman, came to Facebook via their 2007 acquisition of Face.com"



The process of DeepFace:

- Detect
- Align
- Represent
- Classify

Figure 1. Deep Face Process

VGG ResNet

The Core backbone network for our Deep Face is VGG-Resnet.

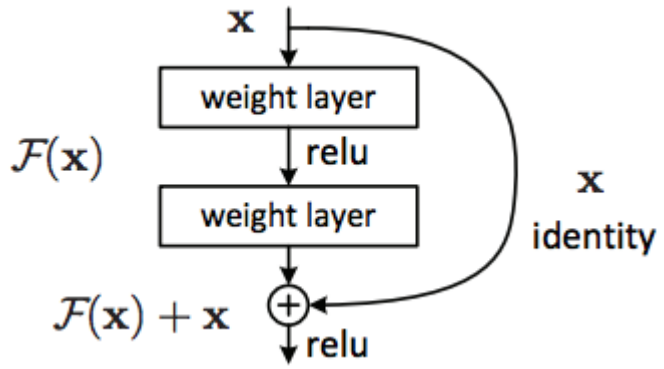


Figure 2. A building Block of Residual Learning

"The core idea of ResNet is introducing a so-called "identity shortcut connection" that skips one or more layers, as shown in the the figure 2 and 3"

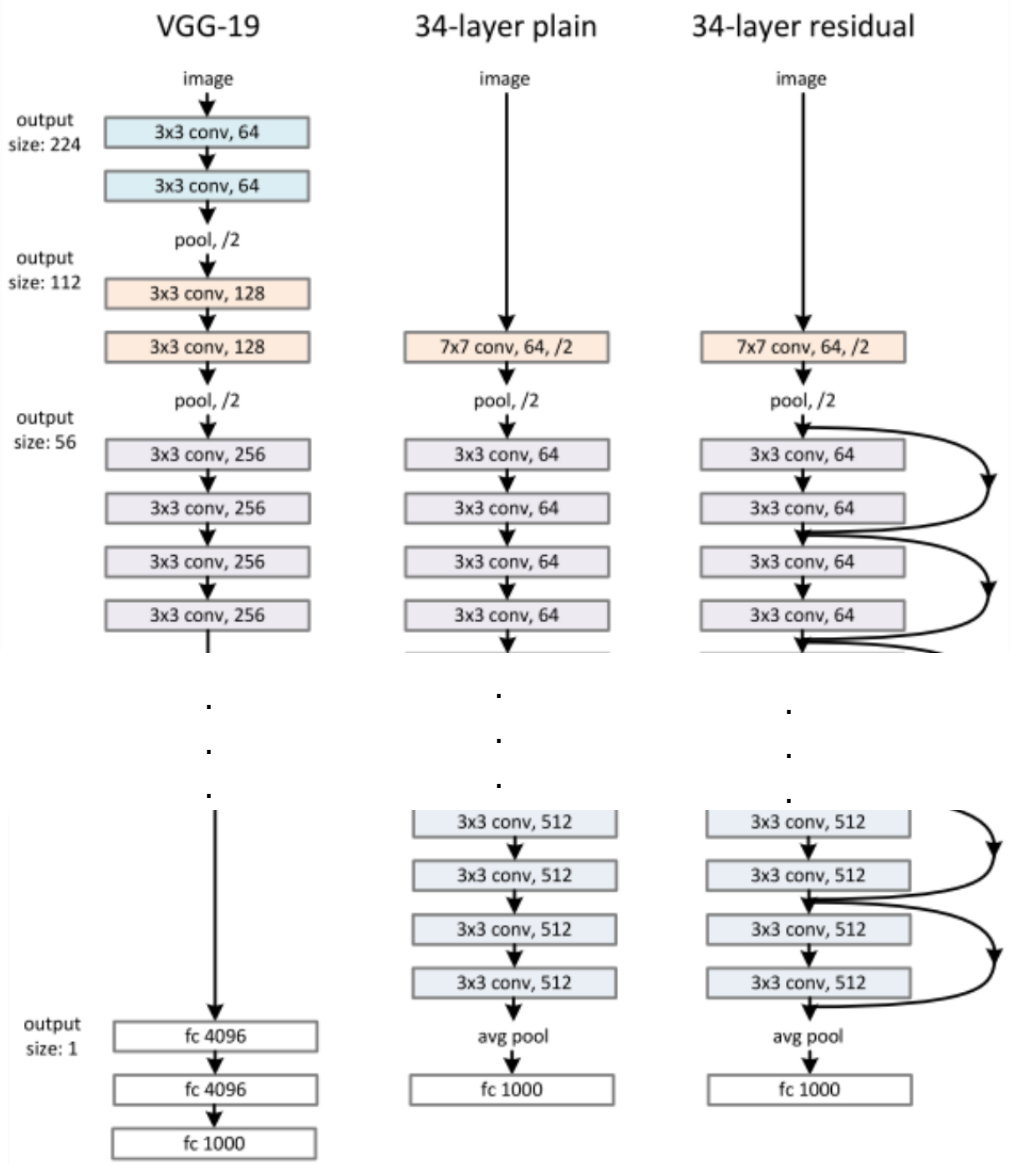


Figure 3. Resnet Architecture

RELU Activation Function

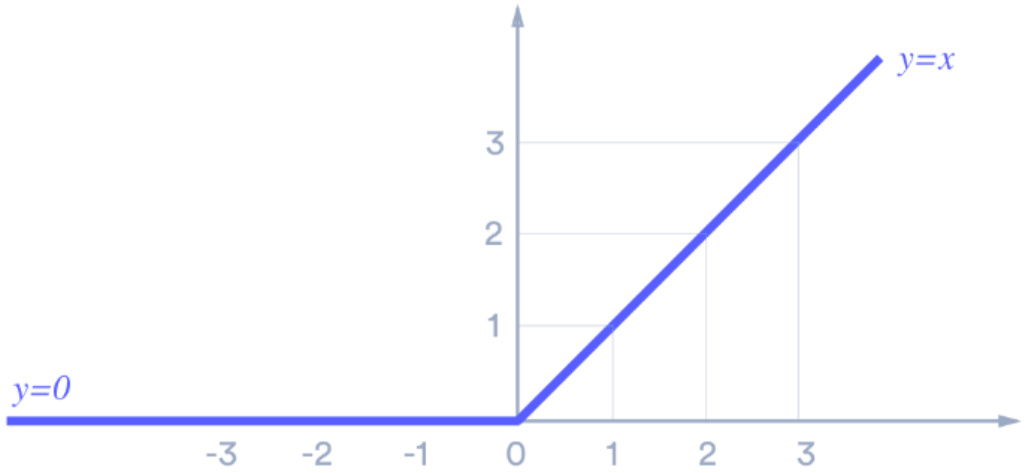
https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7

ReLU stands for rectified linear unit, and is a type of activation function.
Mathematically, it is defined as y = max(0, x).

1. It's cheap to compute as there is no complicated math, takes less time to train or run.

2. It converges faster. Linearity means that the slope doesn't plateau, or "saturate," when x gets large. It doesn't have the vanishing gradient problem suffered by other activation functions like sigmoid or tanh.

3. It's sparsely activated. Since ReLU is zero for all negative inputs.



Why is sparsity good?

It makes intuitive sense, a biological neural network, not all fires all the time.

Sparsity results in concise models that often have better predictive power and less overfitting/noise. In a sparse network, it's more likely that neurons are actually processing meaningful aspects of the problem. For example, in a model detecting cats in images, there may be a neuron that can identify ears, which obviously shouldn't be activated if the image is about a building.

Finally, a sparse network is faster than a dense network, fewer things to compute.

Coding RELU Activation Function

https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7

TF

```
import tensorflow as tfconv_layer =
tf.layers.conv2d(
    inputs=input_layer,
    filters=32,
    kernel_size=[5, 5],
    padding='same',
    activation=tf.nn.relu,
)
```

TensorFlow provides ReLU and its variants through the tf.nn module. For example, the following creates a convolution layer (for CNN) with tf.nn.relu:

Keras

```
from keras.layers import Activation,Dense
model.add(Dense(64, activation='relu'))
```

Keras provides ReLU and its variants through the keras.layers.Activation module. The following adds a ReLU layer to the model:

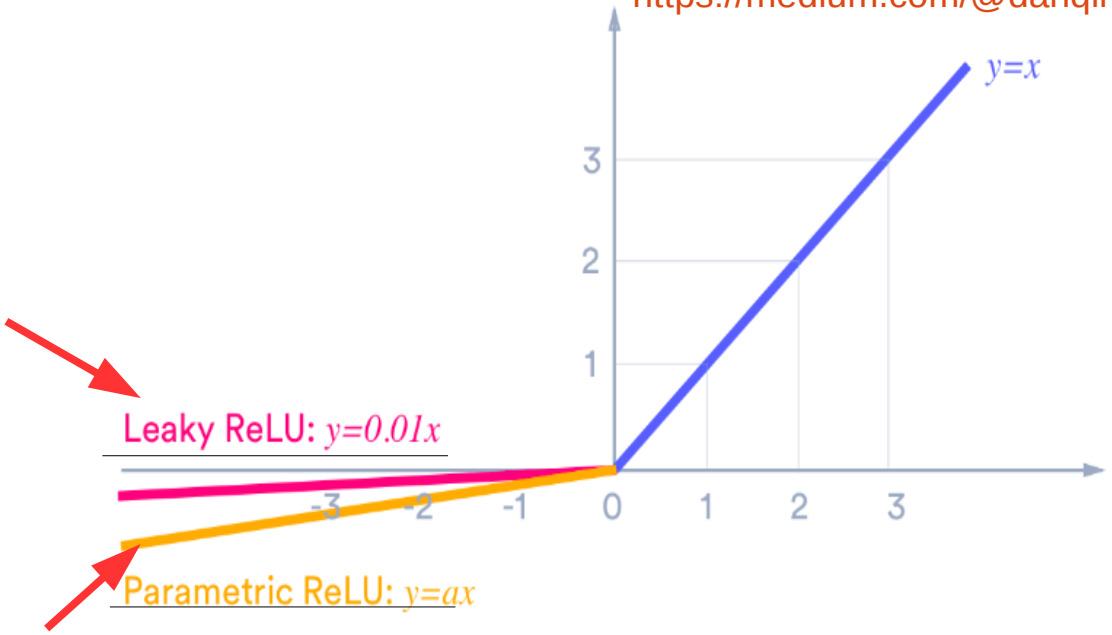
Pytorch

PyTorch provides ReLU and its variants through the torch.nn module. The following adds 2 CNN layers with ReLU:

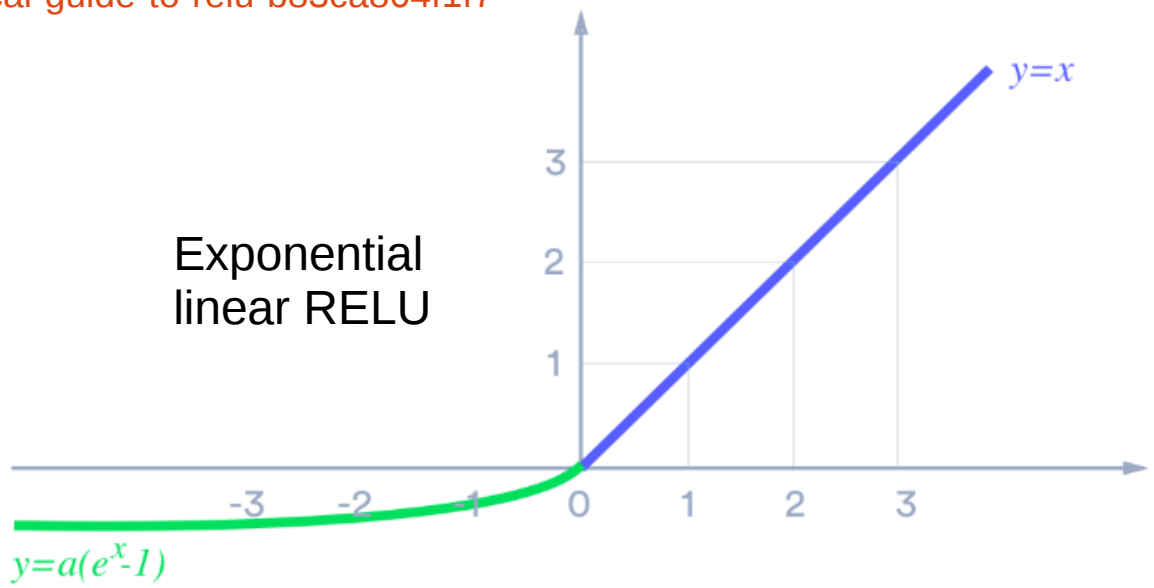
```
from torch.nn import RNNmodel = nn.Sequential(
    nn.Conv2d(1, 20, 5),
    nn.ReLU(),
    nn.Conv2d(20, 64, 5),
    nn.ReLU()
)
```

Dying RELU Improvement

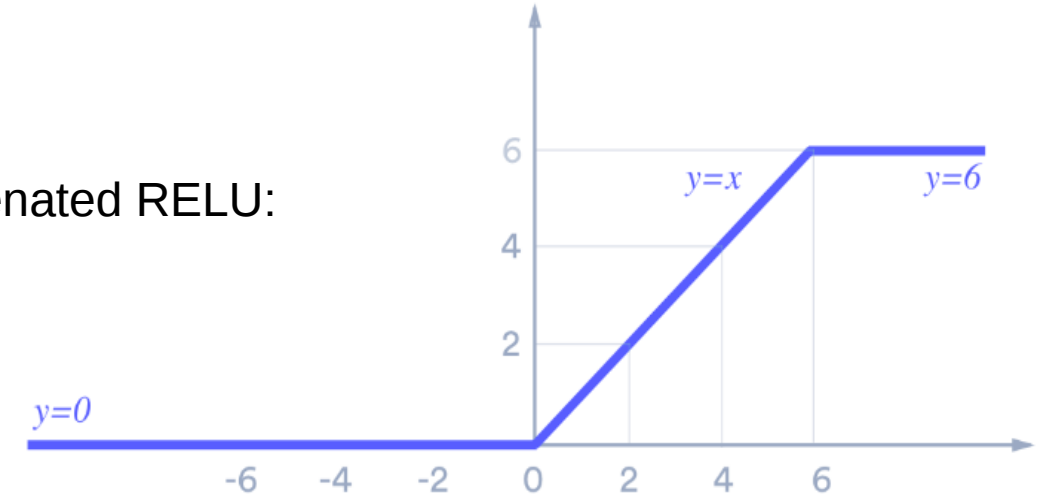
https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7



Exponential
linear RELU



Concatenated RELU:
RELU-6



Result and Performance Problem

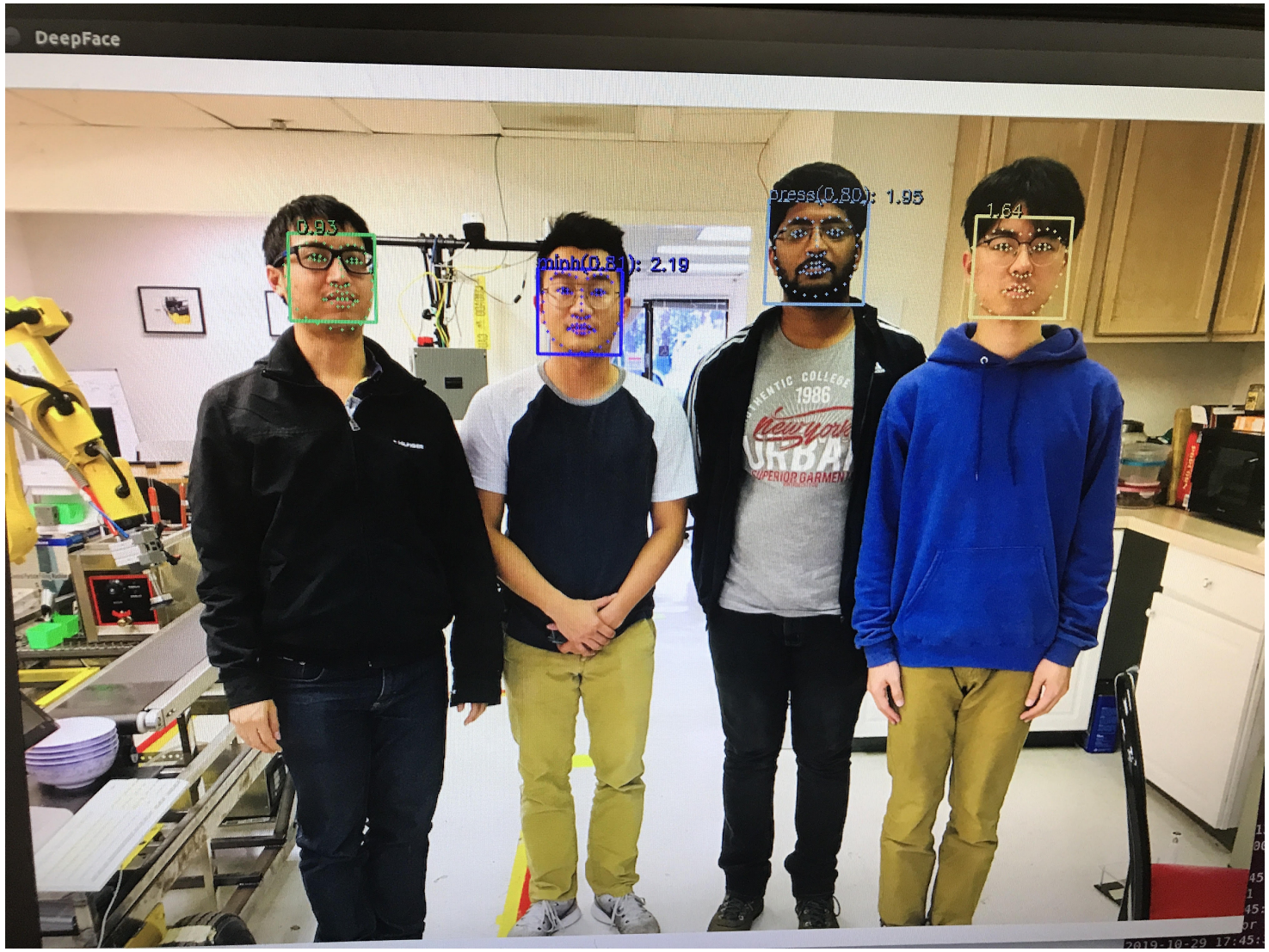


Figure 3. The Result from DeepFace with VGG Resnet. Threshold is 0.6

1. As our real time testing, the accuracy of this model is 94% with Asian Indentities.

2. There is one problem is the performance: Processing time is 1.4s/frame

   Solution: Do not need to Detect every frame. Integrate a “Shallow” method, not “Deep” method: Object ID tracking so, The face indentities can be link to ID of each face object and be remembered when the people are moving inside the frame.

ObjectID Tracker

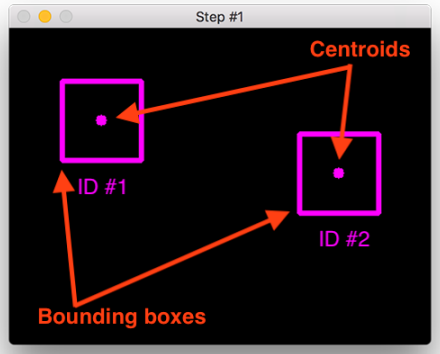
https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/

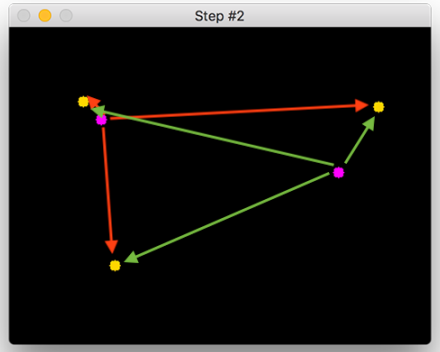
The Process of Object ID tracking is:

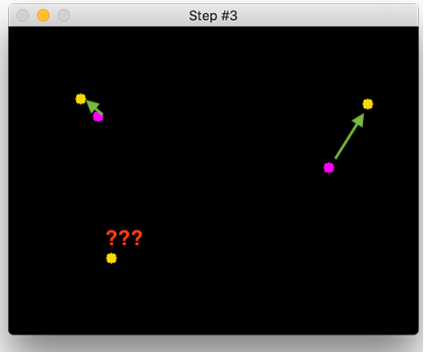
1. Taking an initial set of object detections (such as an input set of bounding box coordinates).
2. Creating a unique ID for each of the initial detections.
3. And then tracking each of the objects as they move around frames in a video, maintaining the assignment of unique IDs.

Algorithm:

Accept bounding box coordinates and compute centroids of each face object → Compute Euclidean distance between new bounding boxes and existing objects → Update (x, y)-coordinates of existing object → Register new objects → Deregister old objects







ObjectID Tracker Algorithm

https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/

An ideal object tracking algorithm will:

1. Only require the object detection phase once (i.e., when the object is initially detected)

2. Will be extremely fast — much faster than running the actual object detector itself

3. Be able to handle when the tracked object “disappears” or moves outside the boundaries of the video frame

4. Be robust to occlusion

5. Be able to pick up objects it has “lost” in between frames

DeepFace+ObjectID Tracker

Algorithm:

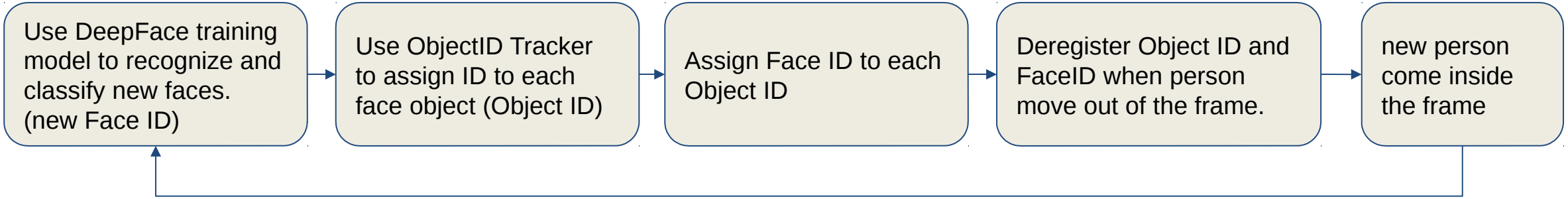


Fig4. Result of FaceID + ObjectID with frame rate is 20 frames/s