Face Recognition Using Vgg16

Imagenet

High level category	# synset (subcategories)	Avg # images per synset	Total # images
amphibian	94	591	56K
animal	3822	732	2799K
appliance	51	1164	59K
bird	856	949	812K
covering	946	819	774K
device	2385	675	1610K
fabric	262	690	181K
fish	566	494	280K
flower	462	735	339K
food	1495	670	1001K
fruit	309	607	188K
fungus	303	453	137K
furniture	187	1043	195K
geological formation	151	838	127K
invertebrate	728	573	417K
mammal	1138	821	934K
musical instrument	157	891	140K
plant	1666	600	999K
reptile	268	707	190K
sport	166	1207	200K
structure	1239	763	946K
tool	316	551	174K
tree	993	568	564K
utensil	86	912	78K
vegetable	176	764	135K
vehicle	481	778	374K
person	2035	468	952K

VGGFace

- Number Of Images= 2.6 million
- Number Of subjects=2,622

ImageNet Pre-trained models

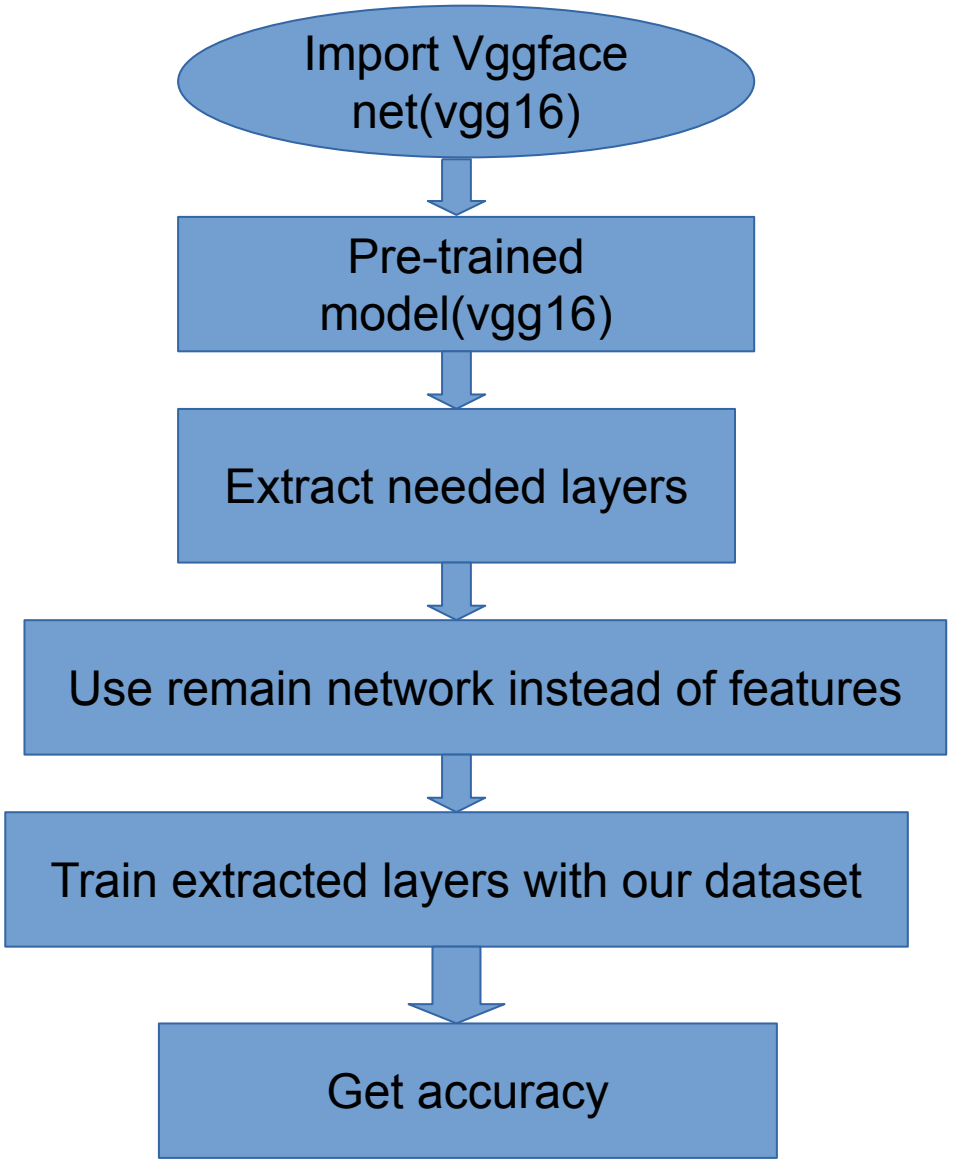
Available models

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-
EfficientNetB0	29 MB	-	-	5,330,571	-
EfficientNetB1	31 MB	-	-	7,856,239	-
EfficientNetB2	36 MB	-	-	9,177,569	-
EfficientNetB3	48 MB	-	-	12,320,535	-
EfficientNetB4	75 MB	-	-	19,466,823	-
EfficientNetB5	118 MB	-	-	30,562,527	-
EfficientNetB6	166 MB	-	-	43,265,143	-
EfficientNetB7	256 MB	-	-	66,658,687	-

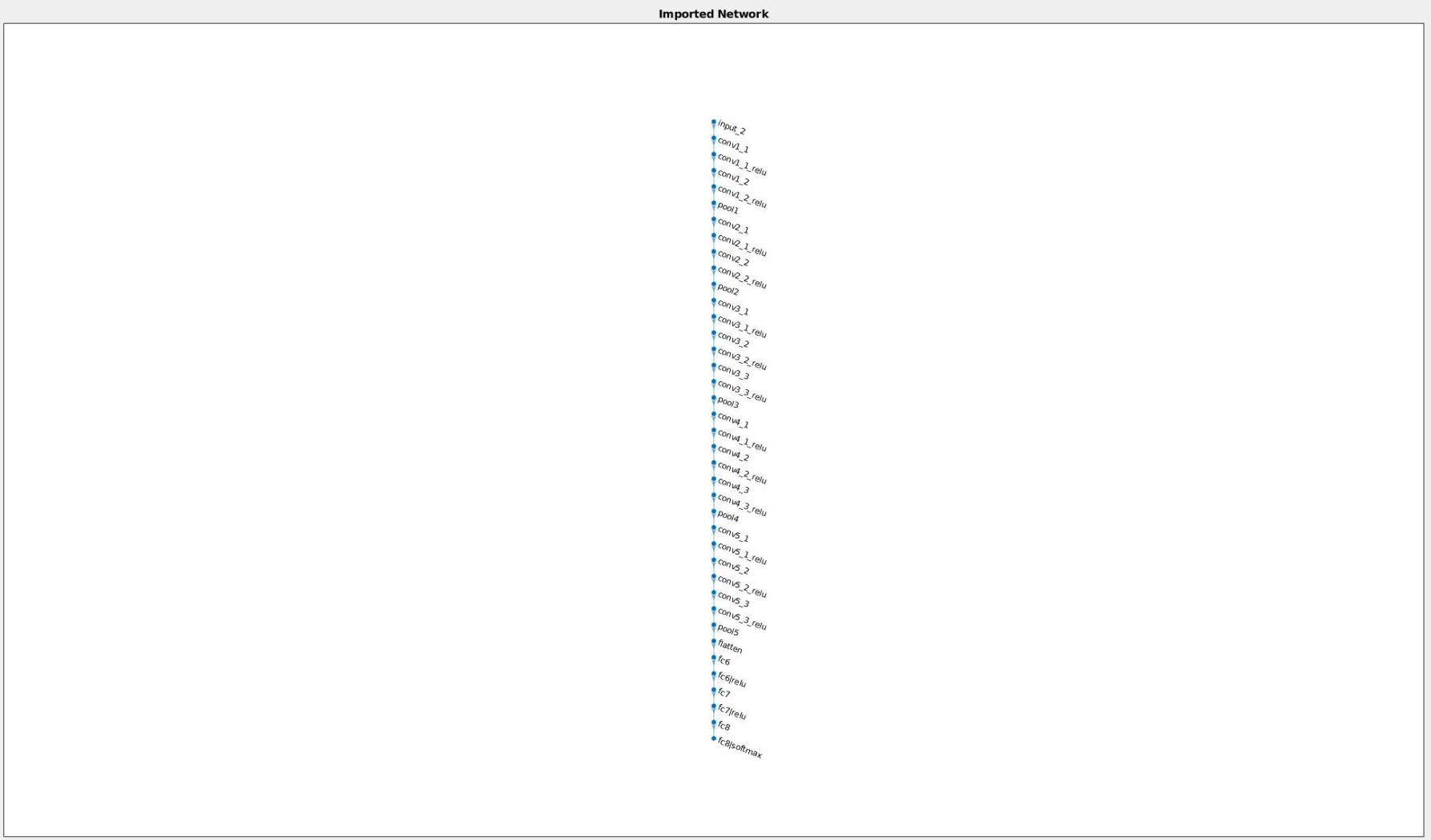
Oxford VGGFace Implementation

- from keras_vggface.vggface import VGGFace
- vggface = VGGFace(model='vgg16')
- vggface = VGGFace(model='resnet50')
- vggface = VGGFace(model='senet50')

Flowchart


Import Vggface net(vgg16)
Pre-trained model(vgg16)
Extract needed layers
Use remain network instead of features
Train extracted layers with our dataset
Get accuracy

Imported Network



Program results

```
cLayer =

  SoftmaxLayer with properties:

    Name: 'fc8|softmax'

Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 4).

inputSize =

   224   224     3


numClasses =

     5
```

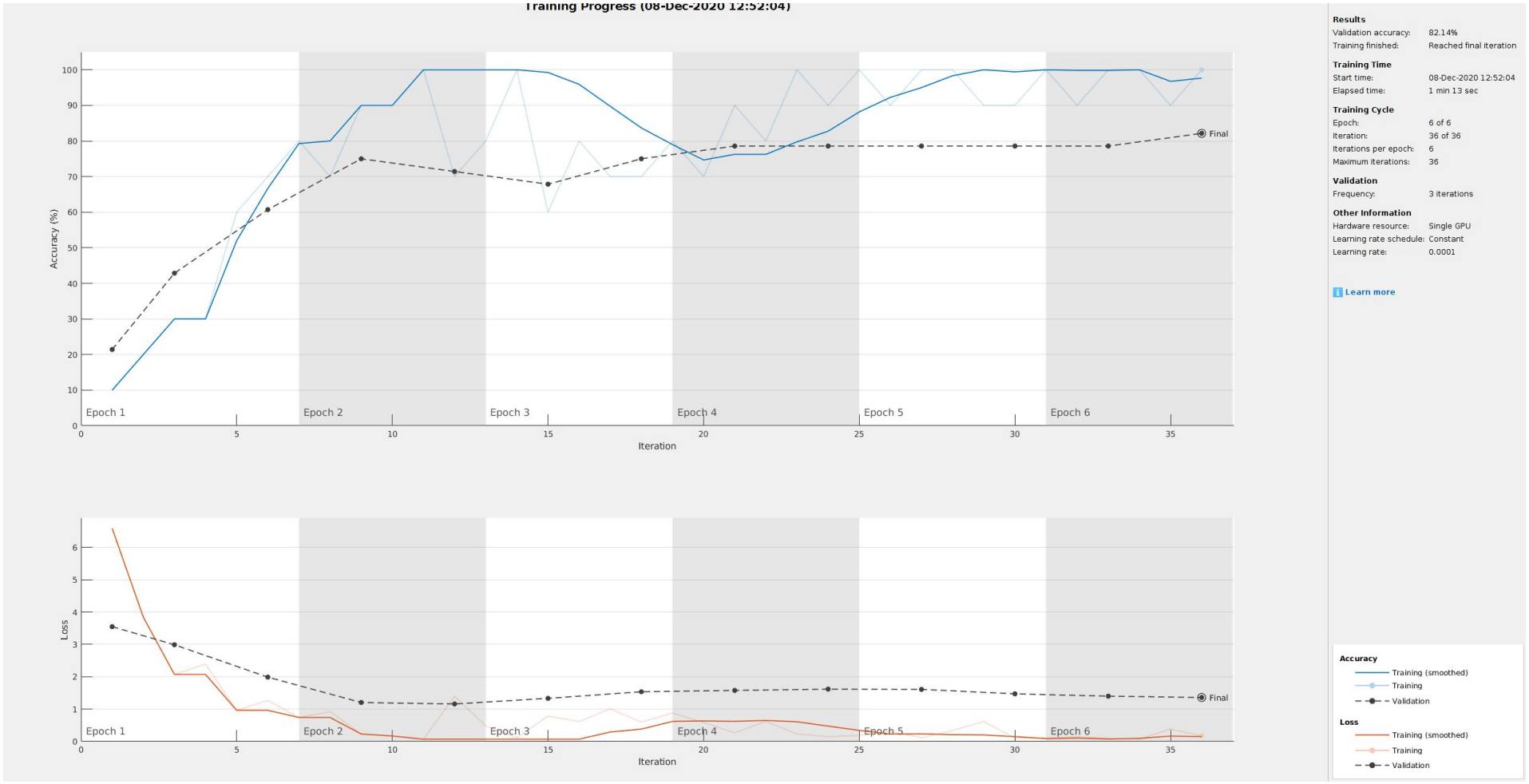
Transfer all layers except 2 last ones

- Extract 2 last layers (Fc8 & Soft max)
- layersTransfer = net.Layers(1:end-2);
- Train the extracted layers with our images
- 5 celebrates dataset used to train the last layers.
- About 100 images

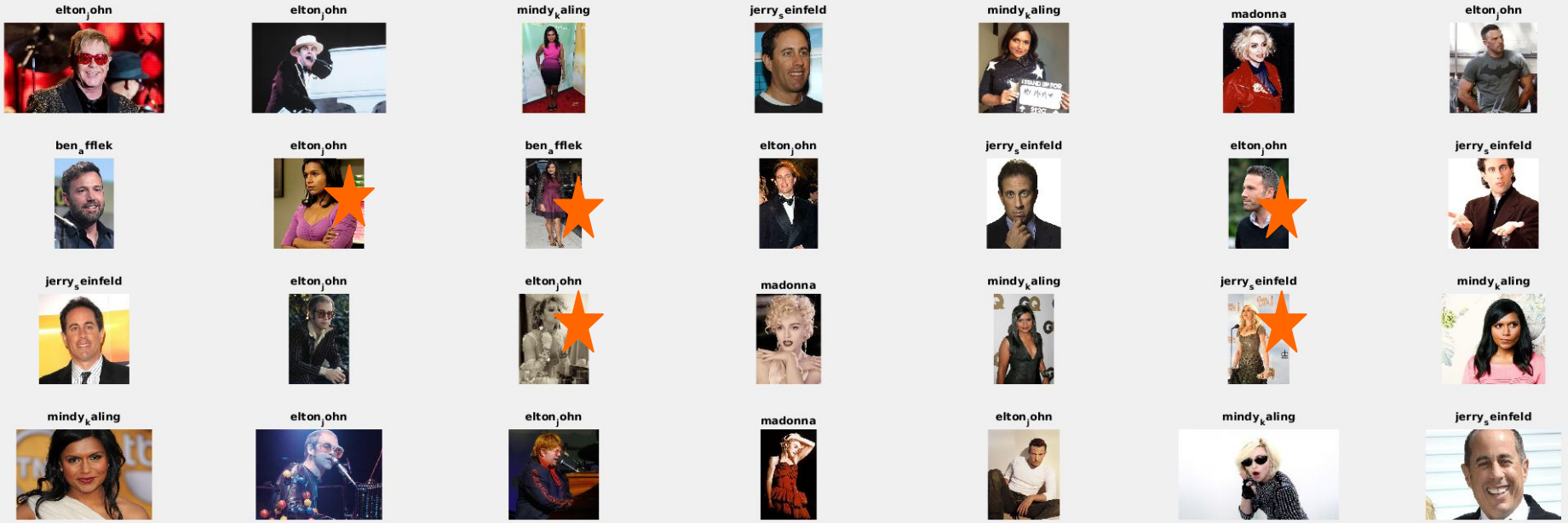
Training Progress

```
Training on single GPU.
|=========================================================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |      Rate       |
|=========================================================================================================================|
|       1 |           1 |       00:00:02 |       10.00% |       21.43% |       6.5863 |       3.5478 |      1.0000e-04 |
Warning: GPU is low on memory, which can slow performance due to additional data transfers with main memory. Try reducing the 'MiniBatchSize'
training option. This warning will not appear again unless you run the command: warning('on','nnet_cnn:warning:GPULowOnMemory').
|       1 |           3 |       00:00:07 |       30.00% |       42.86% |       2.0705 |       2.9877 |      1.0000e-04 |
|       1 |           6 |       00:00:13 |       70.00% |       60.71% |       1.2677 |       1.9878 |      1.0000e-04 |
|       2 |           9 |       00:00:18 |       90.00% |       75.00% |       0.2288 |       1.2047 |      1.0000e-04 |
|       2 |          12 |       00:00:24 |       70.00% |       71.43% |       1.3975 |       1.1586 |      1.0000e-04 |
|       3 |          15 |       00:00:30 |       60.00% |       67.86% |       0.7832 |       1.3325 |      1.0000e-04 |
|       3 |          18 |       00:00:36 |       70.00% |       75.00% |       0.5995 |       1.5357 |      1.0000e-04 |
|       4 |          21 |       00:00:42 |       90.00% |       78.57% |       0.2674 |       1.5767 |      1.0000e-04 |
|       4 |          24 |       00:00:48 |       90.00% |       78.57% |       0.1500 |       1.6170 |      1.0000e-04 |
|       5 |          27 |       00:00:54 |      100.00% |       78.57% |       0.1116 |       1.6098 |      1.0000e-04 |
|       5 |          30 |       00:01:00 |       90.00% |       78.57% |       0.1049 |       1.4717 |      1.0000e-04 |
|       6 |          33 |       00:01:06 |      100.00% |       78.57% |       0.0938 |       1.3989 |      1.0000e-04 |
|       6 |          36 |       00:01:12 |      100.00% |       82.14% |       0.1731 |       1.3590 |      1.0000e-04 |
|=========================================================================================================================|
Elapsed time is 77.728312 seconds.

accuracy =

    0.8214
```

Training Progress



Face recognition of 28 validation images

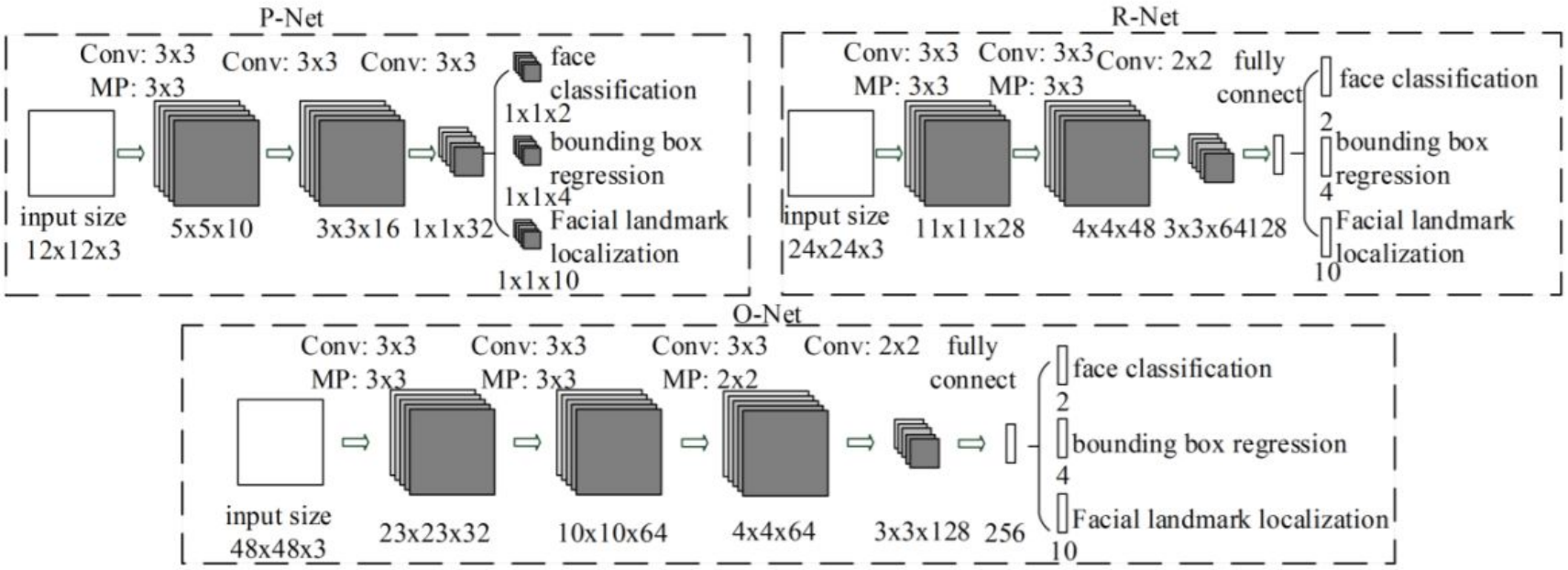


Added preprocessing method

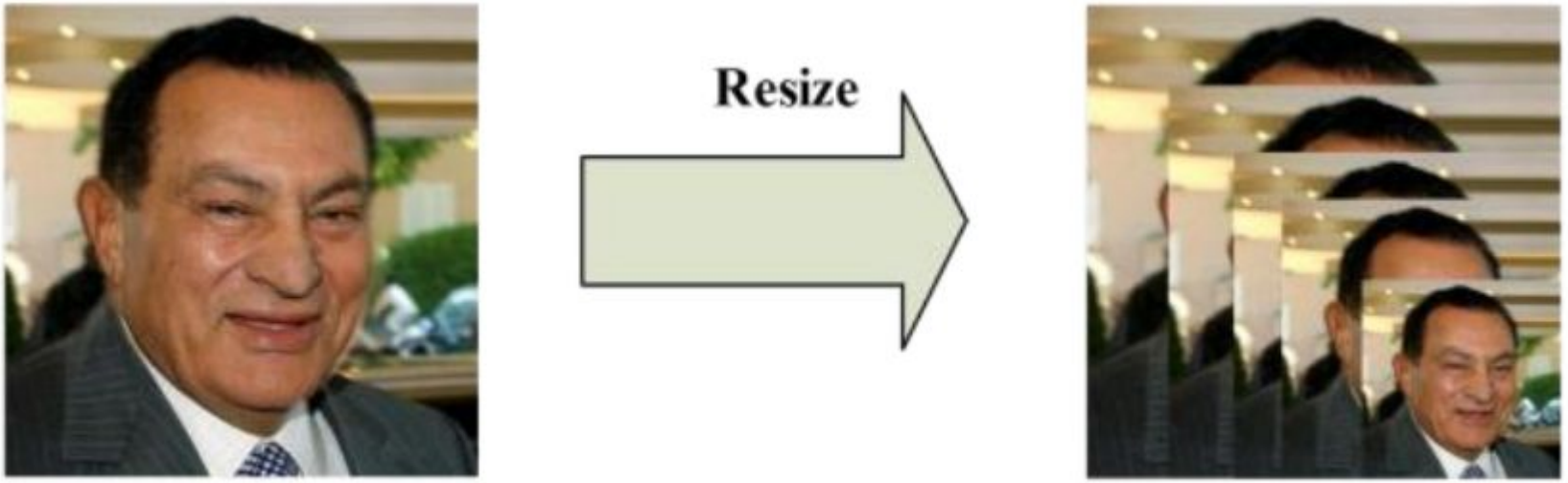
- 1- MTCNN
- 2-vision.CascadeObjectDetector

Multi-task Cascaded Convolutional Network (MTCNN)

- Has three convolutional networks (P-Net, R-Net, and O-Net)



How does MTCNN work?



Train a Cascade Object Detector

- The cascade object detector uses the Viola-Jones algorithm to detect people's faces, noses, eyes, mouth, or upper body.

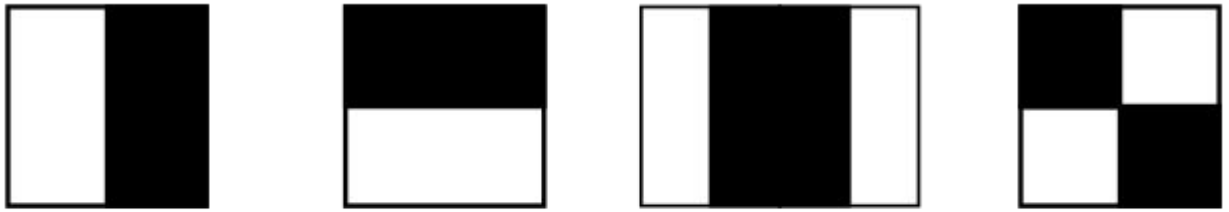
- Robust
- Real time
- Face detection only (not recognition)

Viola-Jones algorithm

- The algorithm has four stages:
- 1-Haar Feature Selection
- 2-Creating an Integral Image
- 3-Adaboost Training
- 4-Cascading Classifiers

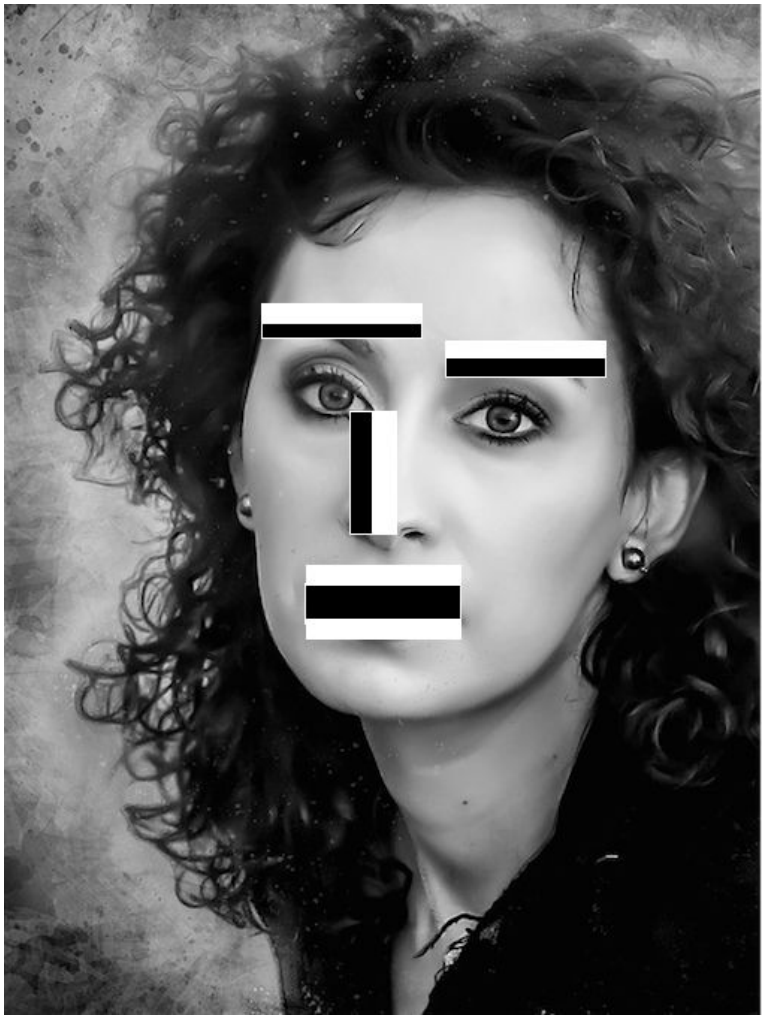
Haar Features

- All human faces share some similar properties. These regularities may be matched using Haar Features.



(1) (2) (3) (4)

- Haar Features



Integral Image Representation

The Value of any point in an Integral Image,
is the sum of all the pixels above and left of that point.

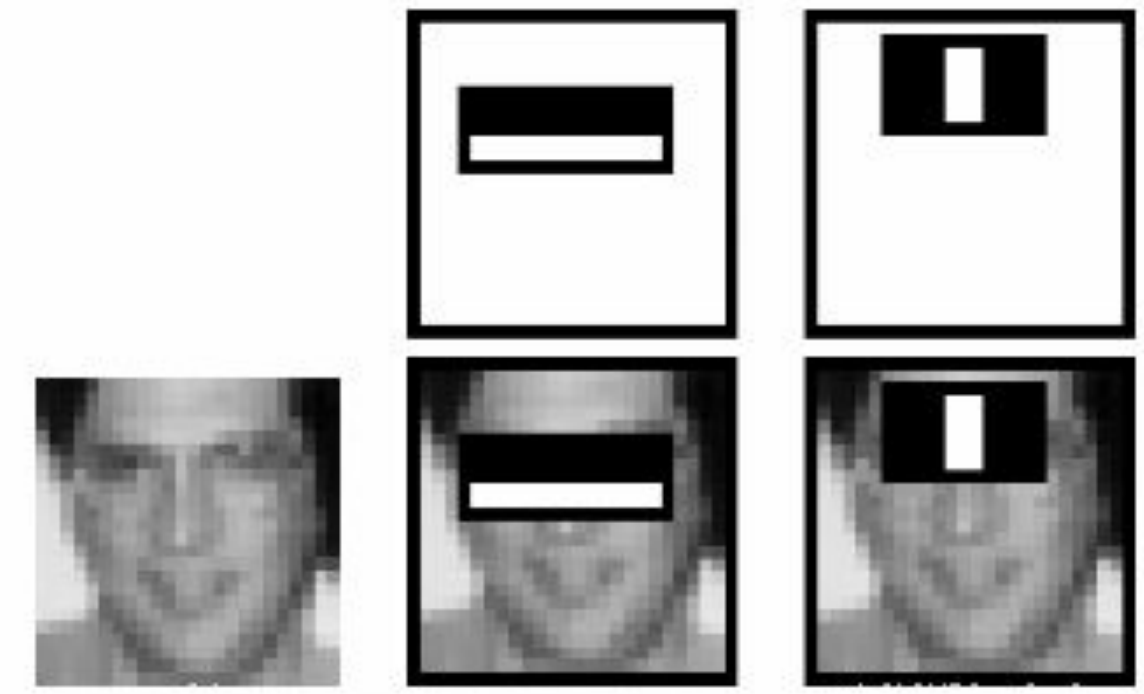
5	2	3	4	1
1	5	4	2	3
2	2	1	3	4
3	5	6	4	5
4	1	3	2	6

Original image

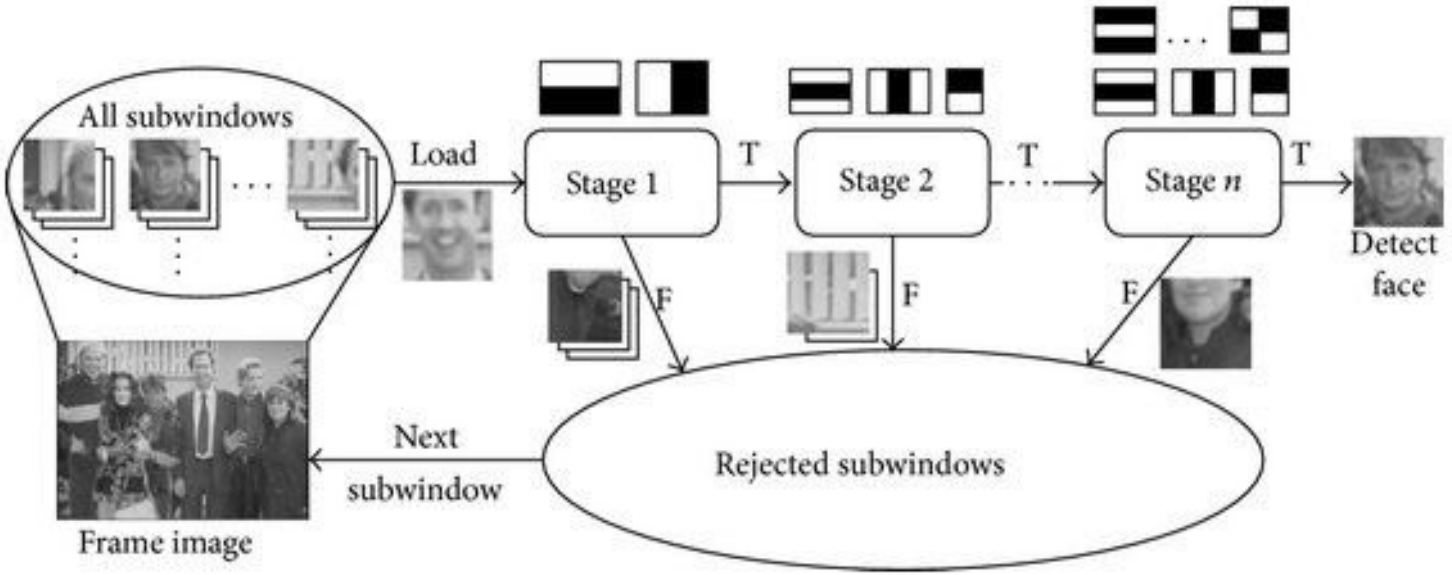
0	0	0	0	0	0
0	5	7	10	14	15
0	6	13	20	26	30
0	8	17	25	34	42
0	11	25	39	52	65
0	15	30	47	62	81

Integral image

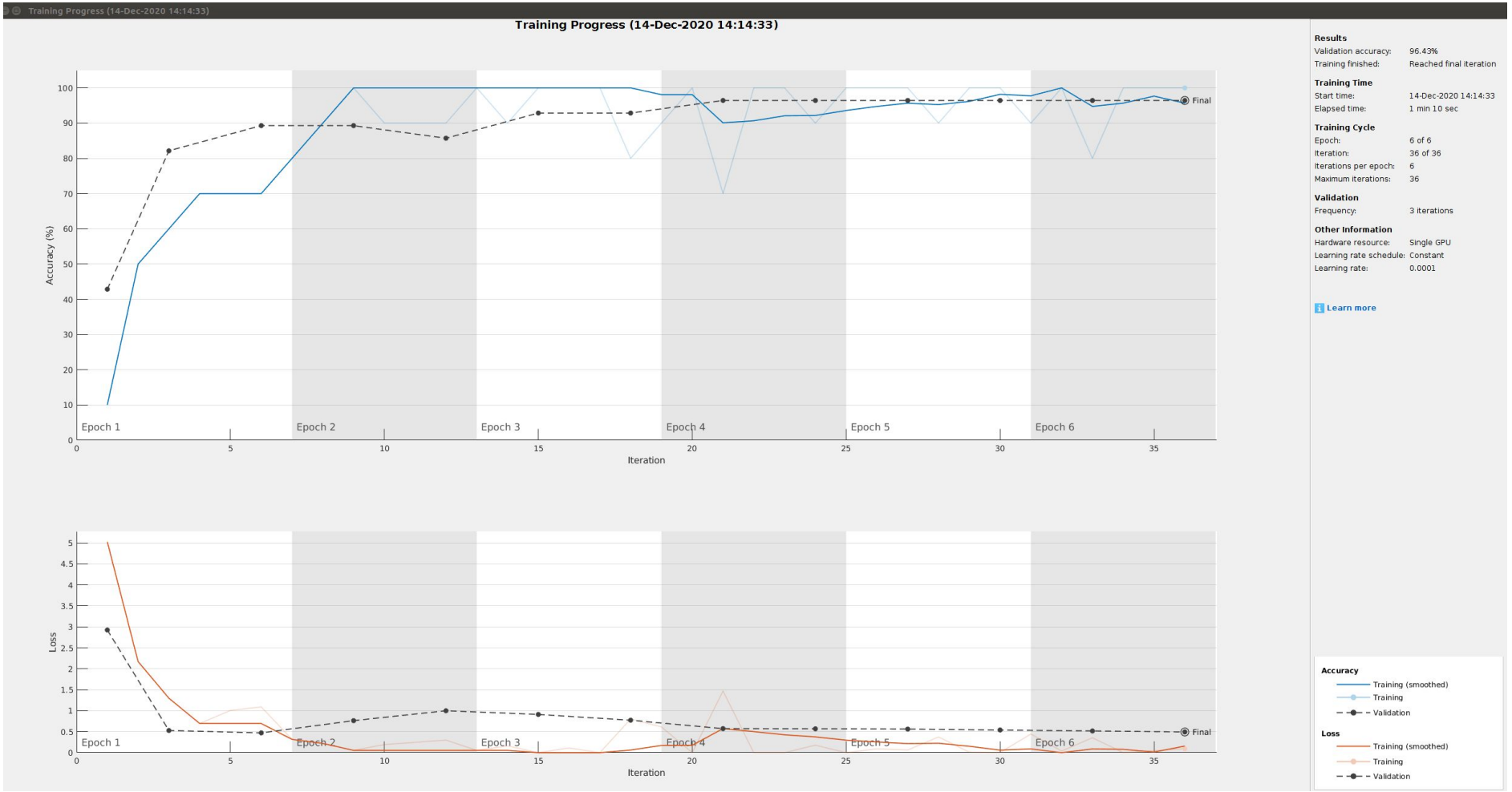
Adaboost Training



Haar Cascade Classifiers



Accuracy after preprocessing methods



Train and validation process

```
numClasses =

     5

Elapsed time is 19.224704 seconds.
Training on single GPU.
|=========================================================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |      Rate       |
|=========================================================================================================================|
|       1 |           1 |       00:00:03 |       10.00% |       42.86% |       5.0264 |       2.9255 |      1.0000e-04 |
Warning: GPU is low on memory, which can slow performance due to additional data transfers with main memory. Try
reducing the 'MiniBatchSize' training option. This warning will not appear again unless you run the command:
warning('on','nnet_cnn:warning:GPULowOnMemory').
|       1 |           3 |       00:00:07 |       60.00% |       82.14% |       1.2987 |       0.5291 |      1.0000e-04 |
|       1 |           6 |       00:00:13 |       70.00% |       89.29% |       1.0913 |       0.4721 |      1.0000e-04 |
|       2 |           9 |       00:00:18 |      100.00% |       89.29% |       0.0546 |       0.7639 |      1.0000e-04 |
|       2 |          12 |       00:00:24 |       90.00% |       85.71% |       0.3008 |       0.9998 |      1.0000e-04 |
|       3 |          15 |       00:00:30 |      100.00% |       92.86% |       0.0001 |       0.9117 |      1.0000e-04 |
|       3 |          18 |       00:00:35 |       80.00% |       92.86% |       0.8042 |       0.7740 |      1.0000e-04 |
|       4 |          21 |       00:00:41 |       70.00% |       96.43% |       1.4708 |       0.5731 |      1.0000e-04 |
|       4 |          24 |       00:00:47 |       90.00% |       96.43% |       0.1795 |       0.5694 |      1.0000e-04 |
|       5 |          27 |       00:00:52 |      100.00% |       96.43% |       0.0633 |       0.5626 |      1.0000e-04 |
|       5 |          30 |       00:00:58 |      100.00% |       96.43% |       0.0046 |       0.5406 |      1.0000e-04 |
|       6 |          33 |       00:01:04 |       80.00% |       96.43% |       0.3559 |       0.5179 |      1.0000e-04 |
|       6 |          36 |       00:01:09 |      100.00% |       96.43% |       0.0953 |       0.4940 |      1.0000e-04 |
|=========================================================================================================================|
Elapsed time is 78.803296 seconds.

accuracy =

    0.9643
```

Face recognition of 28 validation images