Face **E**nabled **E**ntry

Goal

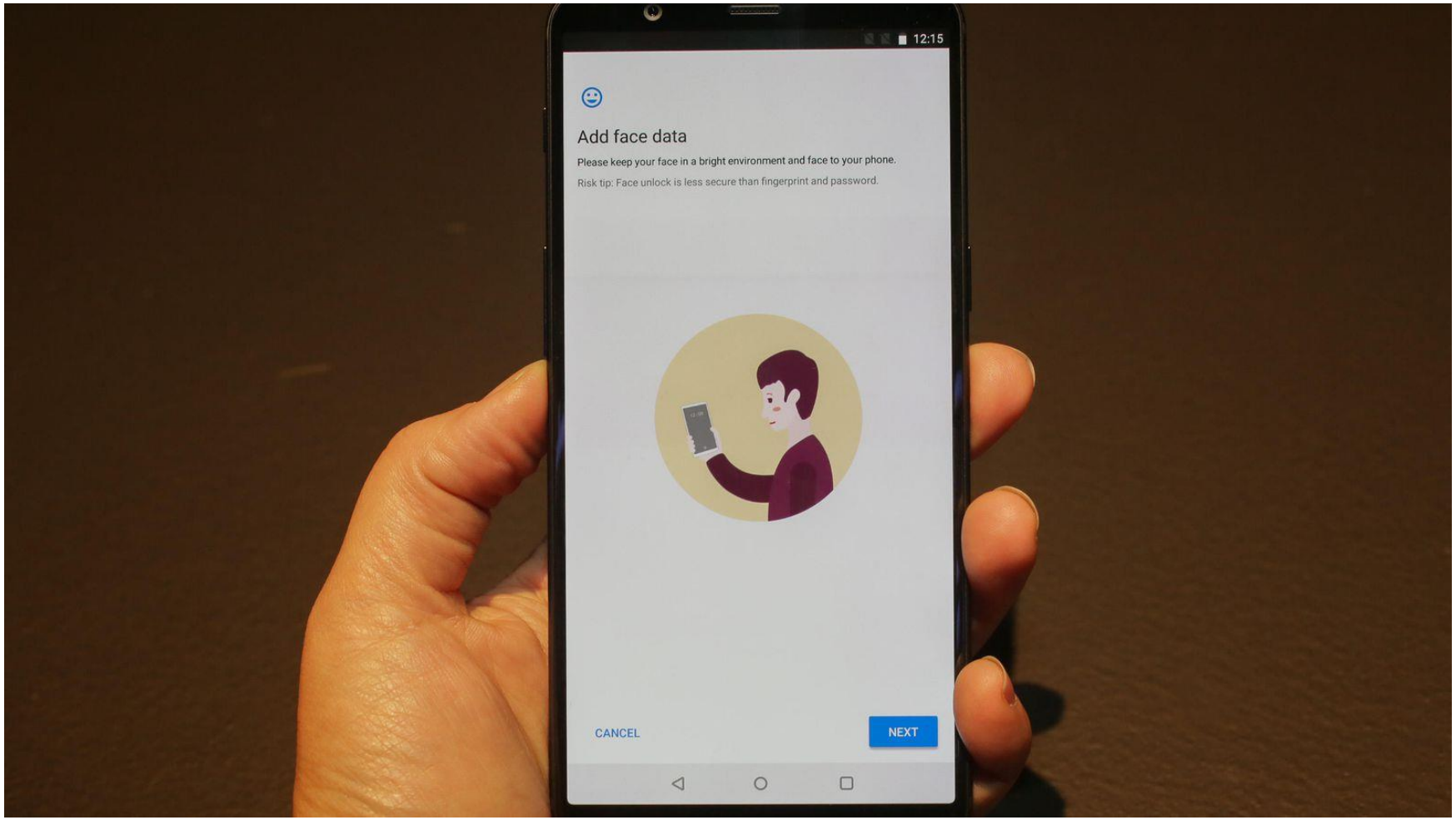
Deploy computer vision based face recognition for facility access

Something link this

https://youtu.be/wr4rx0Spihs

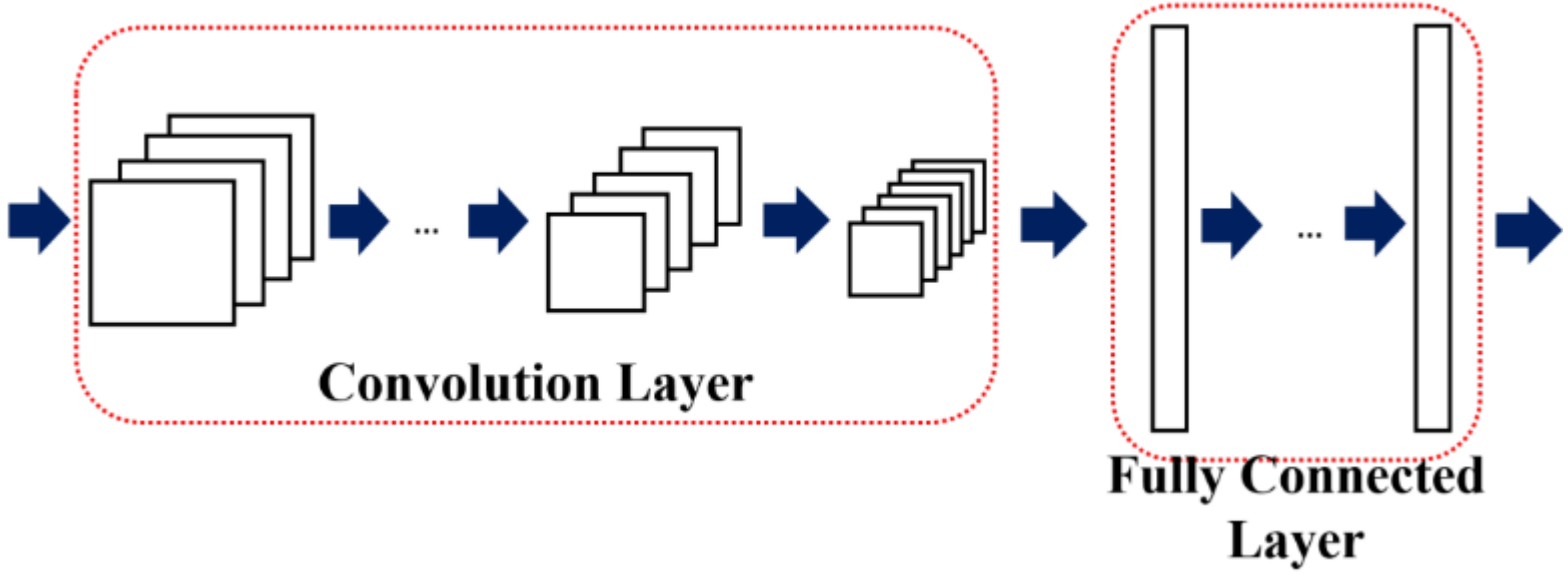
Face unlock

Does it train on your face?



New Face

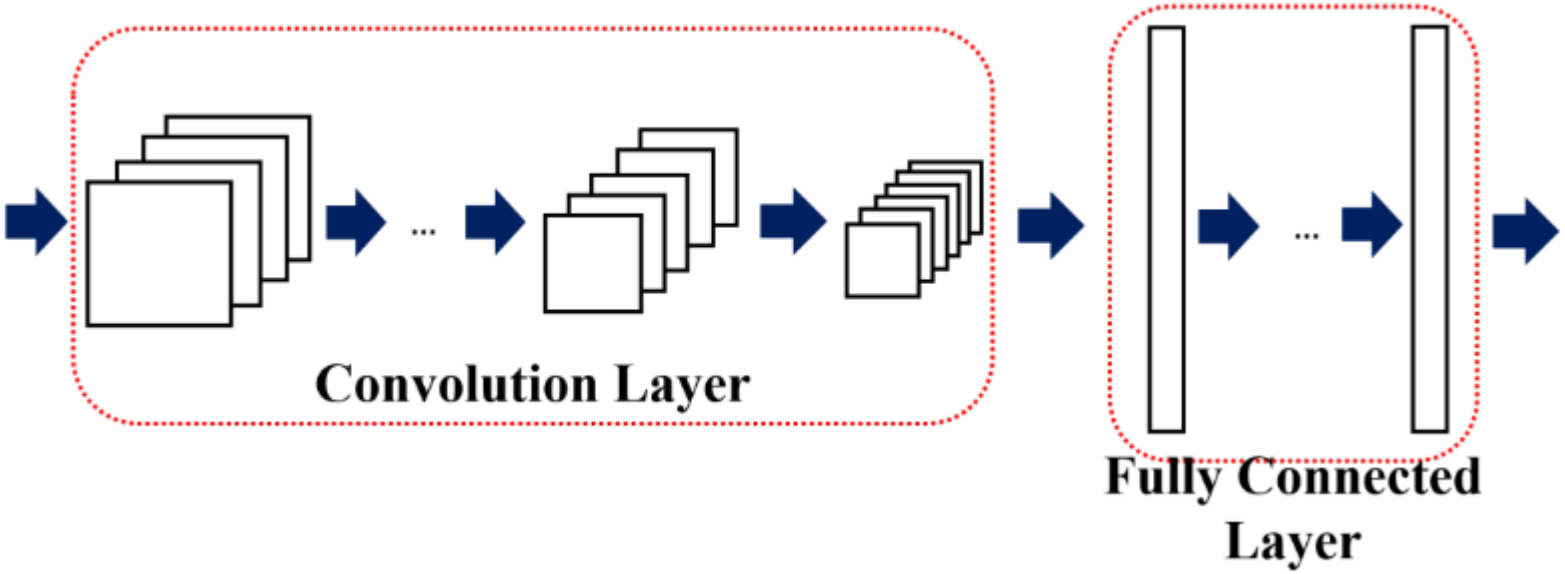
Train to Classify



Rajnikanth

Trisha

New Face



Rajnikanth

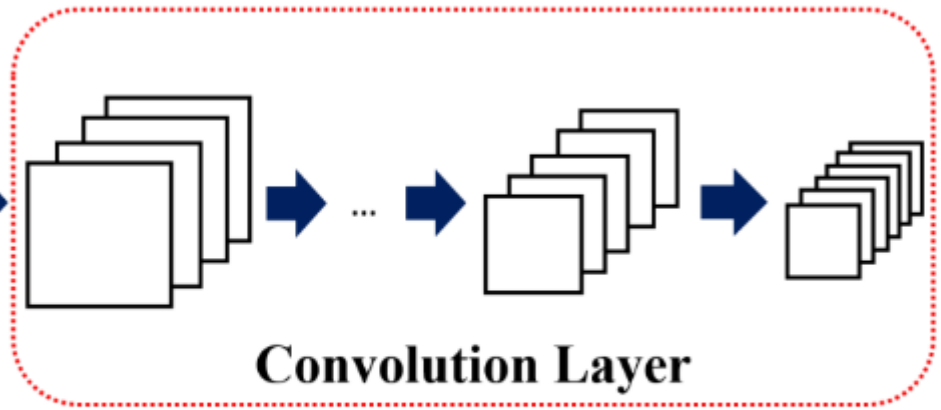
Trisha

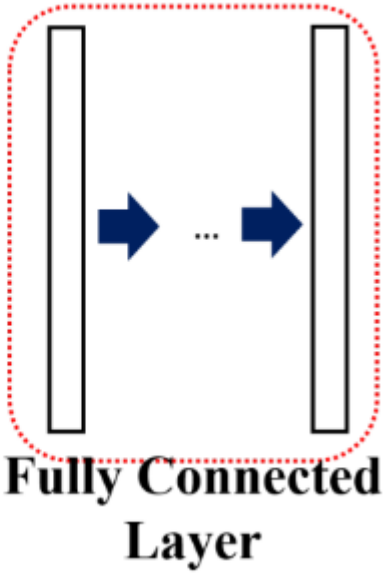
What happens when we bring in a new face??

Solution

Train a network to **distinguish faces** rather than recognizing individual faces

Model

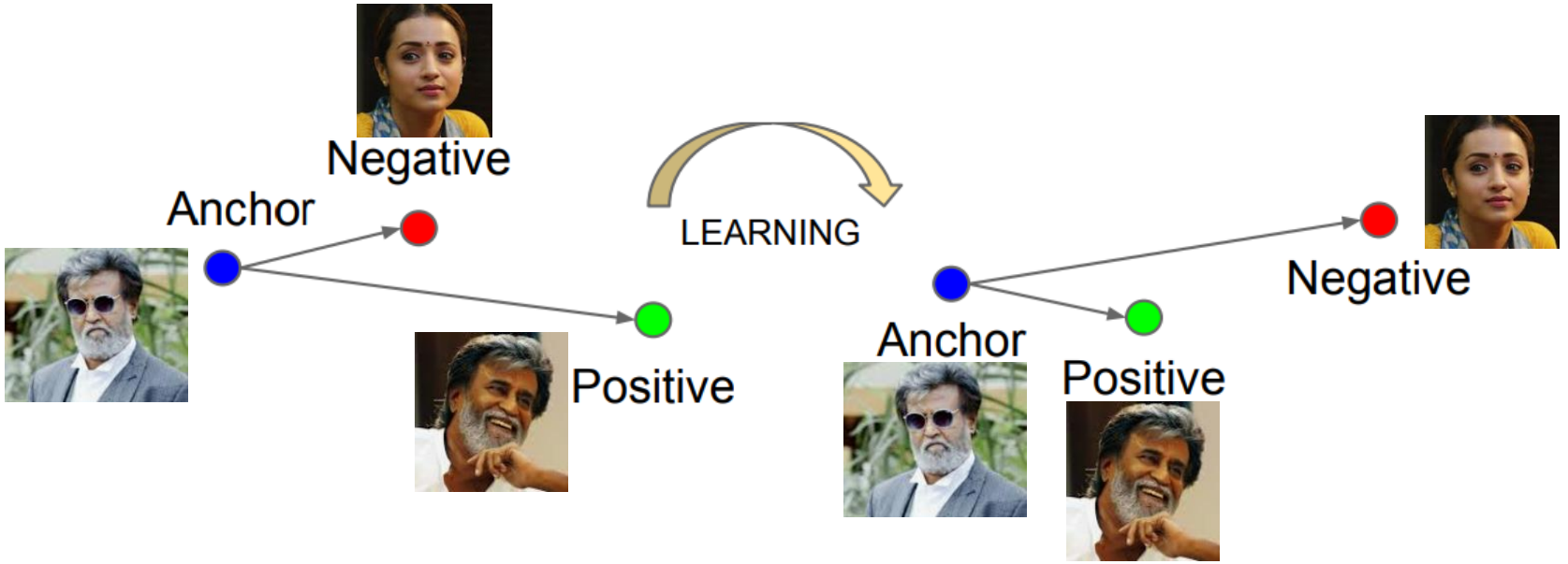




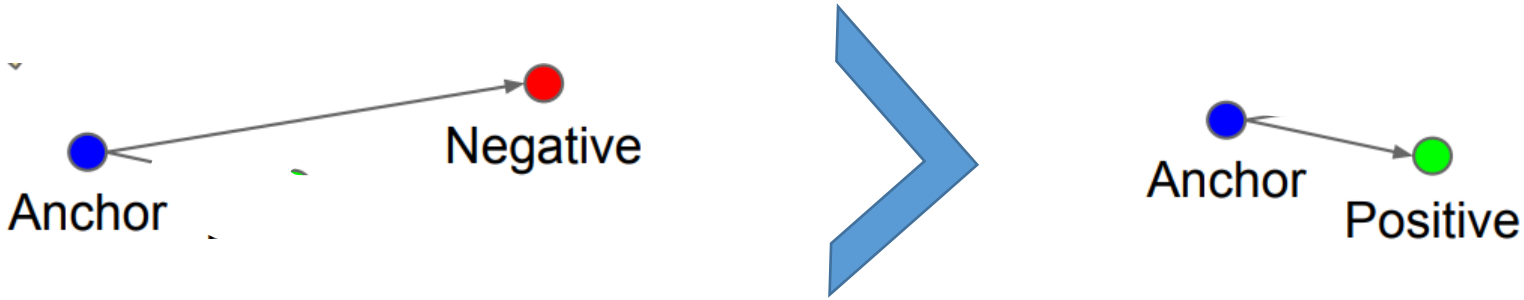
For each image

0.097496084868908	0.045223236083984	-0.1281466782093	0.032084941864014
0.12529824674129	0.060309179127216	0.17521631717682	0.020976085215807
0.030809439718723	-0.01981477253139	0.10801389068365	-0.00052163278451189
0.036050599068403	0.065554238855839	0.0731306001544	-0.1318951100111
-0.097486883401871	0.1226262897253	-0.029626874253154	-0.0059557510539889
-0.0066401711665094	0.036750309169292	-0.15958009660244	0.043374512344599
-0.141315251588882	0.14114324748516	-0.031351584941149	-0.053343612700701
-0.048540540039539	-0.061901587992907	-0.15042643249035	0.078198105096817
-0.12567175924778	-0.105685450136666	-0.12728653848171	-0.076289616525173
-0.061418771743774	-0.074287034571171	-0.065365232527256	0.12369467318058
0.046741496771574	0.006176188122481 1	0.14746543765068	0.056418422609568
-0.12113650143147	-0.21055991947651	0.0041091227903962	0.089727647602558
0.061606746166945	0.11345765739679	0.021352224051952	-0.0085843298584223
0.061989940702915	0.19372203946114	-0.086726233363152	-0.022388197481632
0.10904195904732	0.084853030741215	0.094635948538 78	0.020696049556136
-0.019414527341723	0.006481129676 1036	0.21803123354 91	-0.050584398210049
0.15245945751667	-0.16582328081131	-0.035577941685915	-0.072376452386379
-0.12216668576002	-0.007277755558491	-0.036901291459799	-0.034365277737379
0.083934605121613	-0.059730969369411	-0.070026844739914	-0.045013956725597
0.087945111095905	0.11478432267904	-0.089621491730213	-0.013955107890069
-0.021407851949334	0.14841195940971	0.078333757817745	-0.17898085713387
-0.018298890441656	0.049525424838066	0.13227833807468	-0.072600327432156
-0.011014151386917	-0.051016297191381	-0.14132921397686	0.005051192827522 8
0.009367933496832 8	-0.062812767922878	-0.134074985980 99	-0.014820395338803
0.058139257133007	0.004863874055445 2	-0.039491076022387	-0.043765489012003
-0.024210374802351	-0.11443792283535	0.071997955441475	-0.012062266469002
-0.057223934680223	0.014683869667351	0.052281547337 77	0.012774495407939
0.023535015061498	-0.081752359867096	-0.031709920614958	0.069833360612392
-0.009803973138332 4	0.037022035568953	0.11009479314089	0.116387888778 18
0.020220354199409	0.12788131833076	0.18632389605045	-0.015336792916059
0.004033768083900 2	-0.094398014247417	-0.11768248677254	0.10281457751989
0.051597066223621	-0.10034311562777	-0.040977258235216	-0.082041138086128

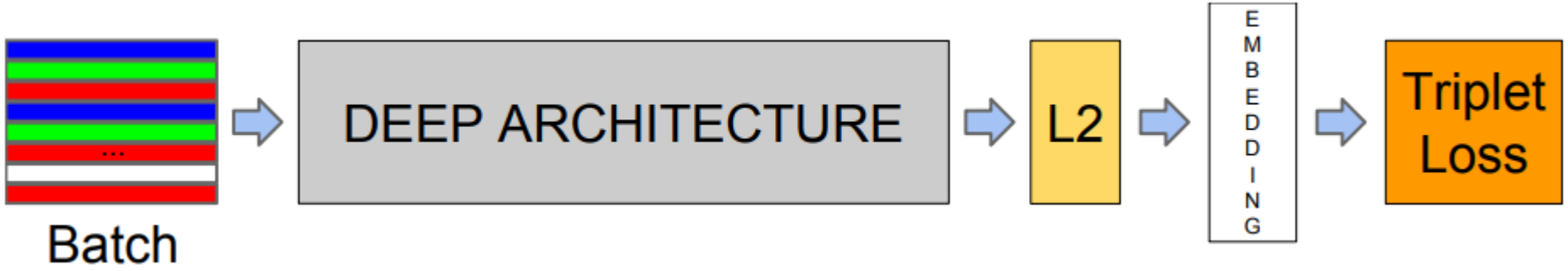
Triplets Network



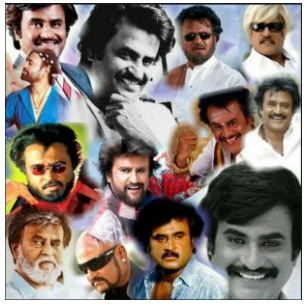
- The Triplet Loss minimizes the distance between an anchor and a positive, both of which have the same identity, and maximizes the distance between the anchor and a negative of a different identity.



Model



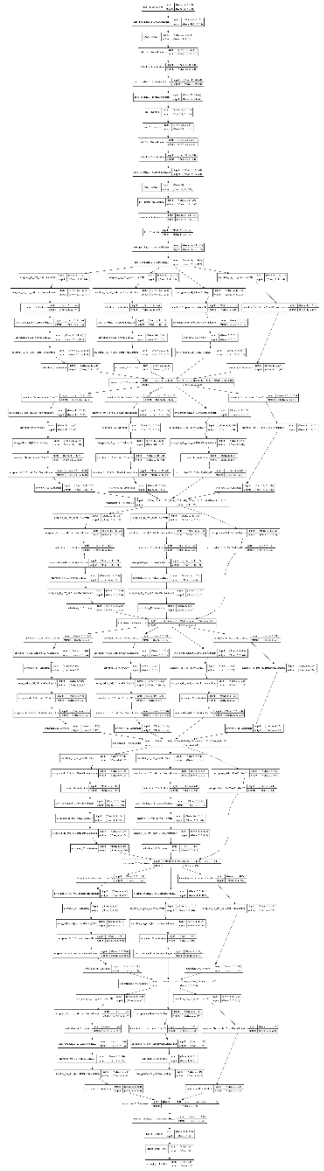
Model Learning



New Face

Model V1

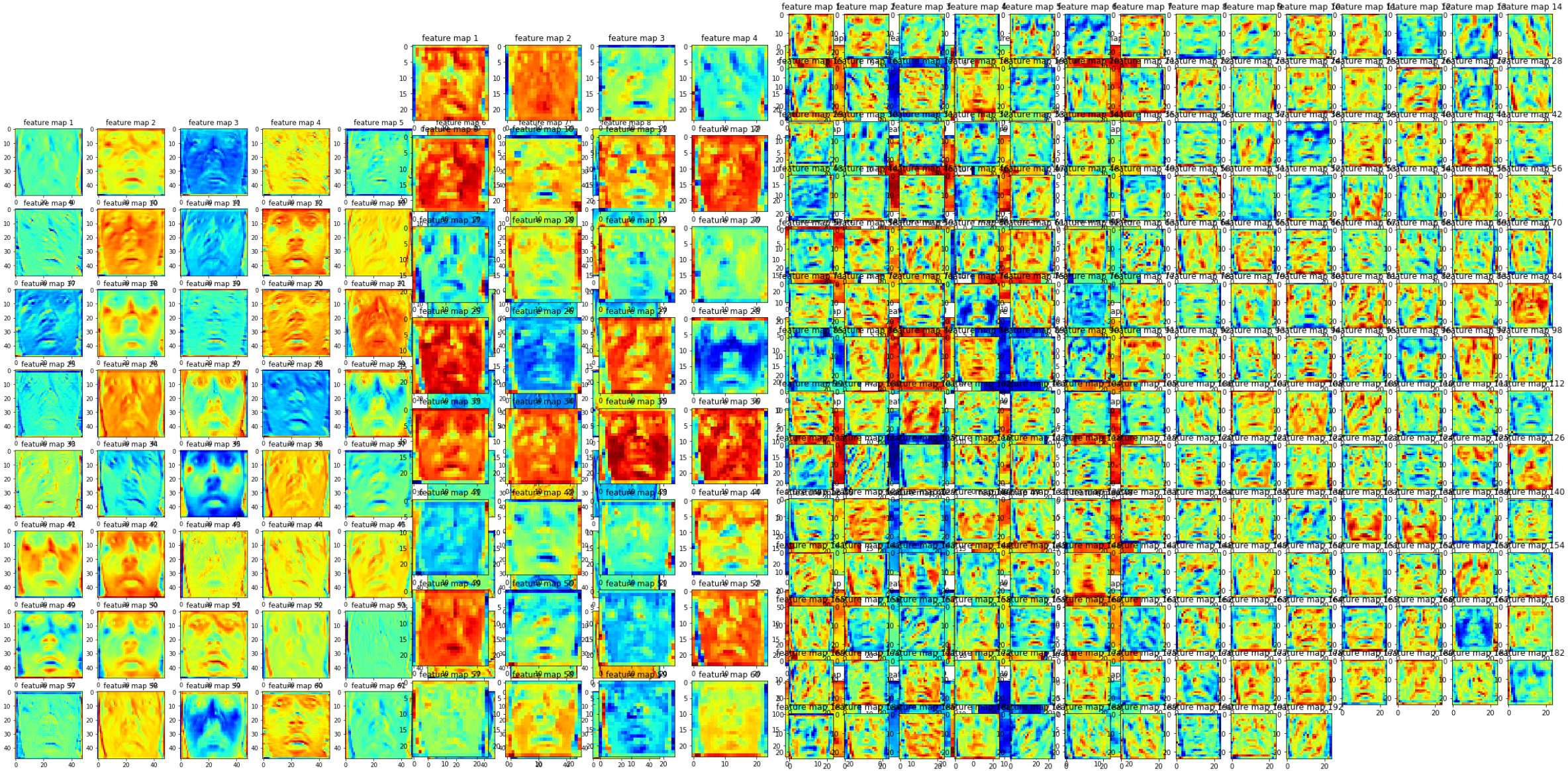
- Trained on 80 Lakh identities
- Total of 10 – 20 Crore images
- Trained on a CPU cluster for 3 months



layer	size-in	size-out	kernel	param	FLPS
conv1	220×220×3	110×110×64	7×7×3, 2	9K	115M
pool1	110×110×64	55×55×64	3×3×64, 2	0	
rnorm1	55×55×64	55×55×64		0	
conv2a	55×55×64	55×55×64	1×1×64, 1	4K	13M
conv2	55×55×64	55×55×192	3×3×64, 1	111K	335M
rnorm2	55×55×192	55×55×192		0	
pool2	55×55×192	28×28×192	3×3×192, 2	0	
conv3a	28×28×192	28×28×192	1×1×192, 1	37K	29M
conv3	28×28×192	28×28×384	3×3×192, 1	664K	521M
pool3	28×28×384	14×14×384	3×3×384, 2	0	
conv4a	14×14×384	14×14×384	1×1×384, 1	148K	29M
conv4	14×14×384	14×14×256	3×3×384, 1	885K	173M
conv5a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv5	14×14×256	14×14×256	3×3×256, 1	590K	116M
conv6a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv6	14×14×256	14×14×256	3×3×256, 1	590K	116M
pool4	14×14×256	7×7×256	3×3×256, 2	0	
concat	7×7×256	7×7×256		0	
fc1	7×7×256	1×32×128	maxout p=2	103M	103M
fc2	1×32×128	1×32×128	maxout p=2	34M	34M
fc7128	1×32×128	1×1×128		524K	0.5M
L2	1×1×128	1×1×128		0	
total				140M	1.6B

Table 1. **NN1.** This table show the structure of our Zeiler&Fergus [22] based model with 1×1 convolutions inspired by [9]. The input and output sizes are described in $rows \times cols \times \#filters$. The kernel is specified as $rows \times cols, stride$ and the maxout [6] pooling size as $p = 2$.

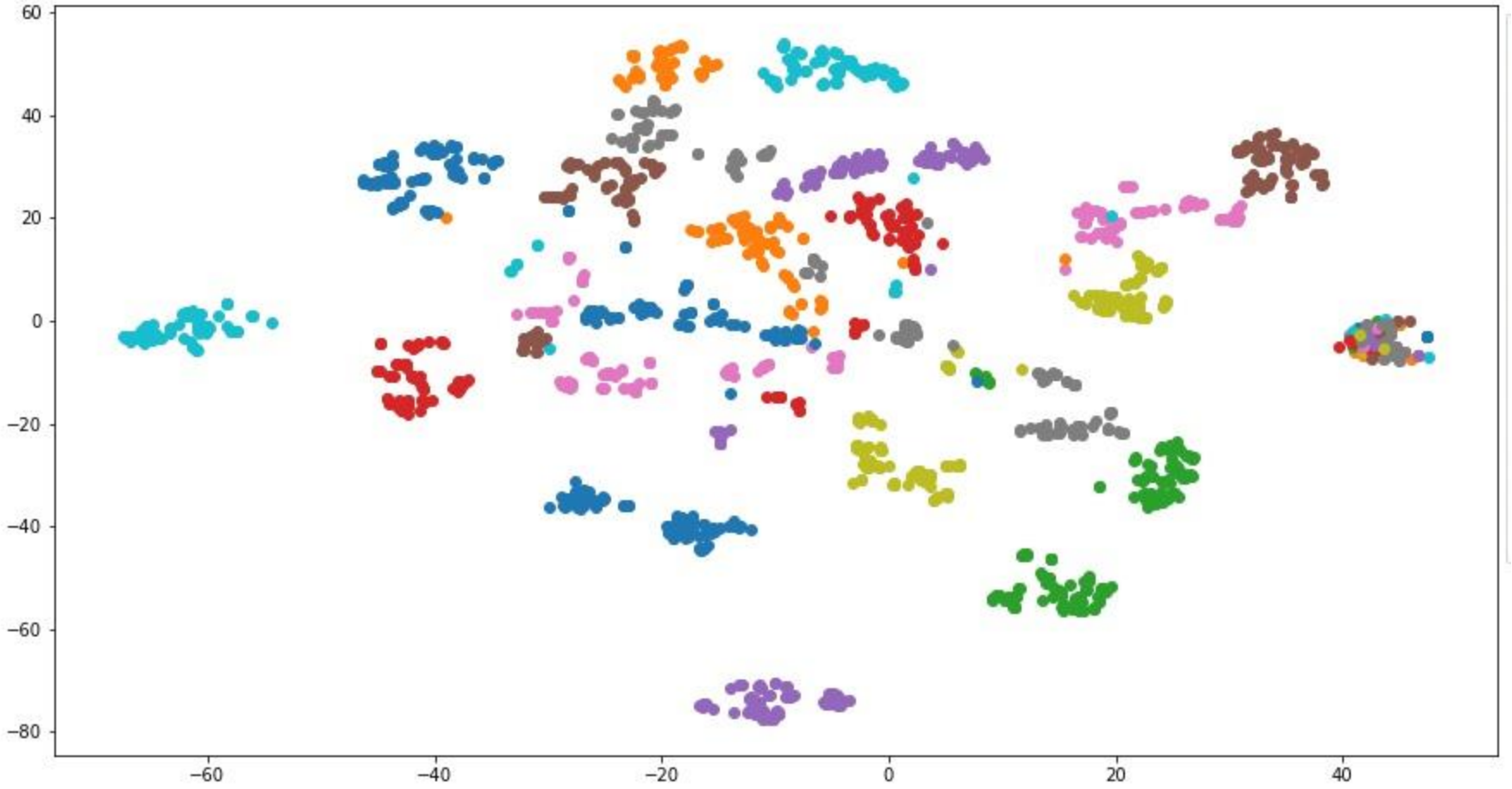
Layer Visualization



Embedding

```
[[ -1.88833997e-02   9.83617157e-02   2.07915023e-01   1.44615799e-01
    5.80447316e-02  -5.01837255e-03   5.85894957e-02  -1.61365584e-01
    1.08106136e-01  -9.40666199e-02  -3.61320451e-02   3.85740548e-02
   -8.14742073e-02   1.75796300e-02   5.56245185e-02  -1.77704059e-02
   -6.06725598e-03  -1.26523552e-02  -1.35551542e-01   2.15995181e-02
   -3.33796218e-02  -3.24995108e-02  -6.60371706e-02   1.04821518e-01
    1.33706376e-01  -1.17994018e-01  -1.66413590e-01   1.35322377e-01
    7.56528648e-03   1.77322906e-02   1.10007100e-01   1.40904322e-01
   -3.51901874e-02   8.07275325e-02   1.09597817e-02  -2.76857102e-03
    3.21411937e-02   2.11234558e-02   2.21219230e-02  -9.64475647e-02
    1.28499866e-01  -2.32732054e-02   1.22369967e-01  -2.37366855e-01
    7.43315667e-02  -5.40643372e-02  -3.93253677e-02   2.84477277e-03
    9.68252867e-03  -5.14560561e-05  -4.07368317e-02  -2.68205744e-03
    1.23449631e-01  -3.70667987e-02   8.02015290e-02   7.38517987e-03
   -1.55605435e-01   6.90959394e-02  -9.79714617e-02  -1.63167082e-02
   -9.28533301e-02   9.92244408e-02  -5.91525398e-02  -1.15624070e-01
    3.33569851e-03   3.21331292e-01   2.83122901e-03  -8.59396830e-02
   -2.78891325e-01  -1.97359012e-03   8.56173132e-03   4.03305404e-02
    1.56095612e-03   1.38616443e-01   1.03200153e-01   2.66493652e-02
    1.19524017e-01  -1.65473357e-01   3.80755551e-02  -7.34731602e-03
    2.65281014e-02   4.98776734e-02   2.79466677e-02  -3.88022028e-02
    7.26109296e-02  -6.04184382e-02  -7.25998580e-02   5.06432615e-02
   -4.96999137e-02   1.18302144e-01   8.90203118e-02  -5.81578165e-02
    1.02084234e-01   7.57777393e-02  -9.05318465e-03   4.72530760e-02
   -4.98152263e-02  -6.24167509e-02   5.94433844e-02   3.41376266e-03
    6.60767639e-03  -9.23319682e-02  -6.70323223e-02   4.40597795e-02
   -2.18763519e-02   1.32832915e-01   6.76512942e-02  -3.69001441e-02
    5.74899130e-02  -1.86466798e-02   4.03219368e-03   7.26963058e-02
   -1.67973861e-02   7.07234293e-02   4.92062047e-02  -8.89770091e-02
   -6.33207336e-02   1.81835130e-01   3.00272349e-02   7.91064501e-02
    4.25065830e-02   7.04212636e-02  -3.45968120e-02  -1.61775518e-02
    2.40377150e-02   1.24002919e-01   8.01019445e-02  -1.23444684e-01]]
```

Clustering



Recognition

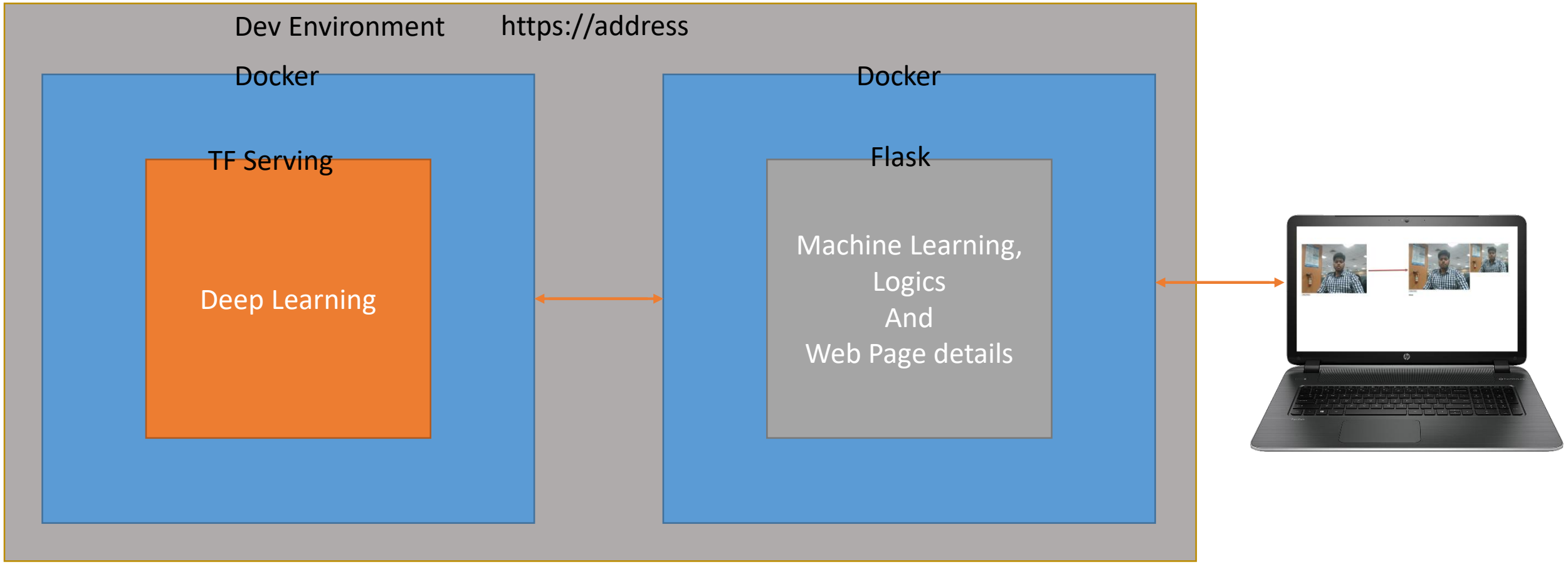
KNN and/or SVM to get predictions

```
[[ -1.88833997e-02   9.83617157e-02   2.07915023e-01   1.44615799e-01
    5.80447316e-02  -5.01837255e-03   5.85894957e-02  -1.61365584e-01
    1.08106136e-01  -9.40666199e-02  -3.61320451e-02   3.85740548e-02
   -8.14742073e-02   1.75796300e-02   5.56245185e-02  -1.77704059e-02
   -6.06725598e-03  -1.26523552e-02  -1.35551542e-01   2.15995181e-02
   -3.33796218e-02  -3.24995108e-02  -6.60371706e-02   1.04821518e-01
    1.33706376e-01  -1.17994018e-01  -1.66413590e-01   1.35322377e-01
    7.56528648e-03   1.77322906e-02   1.10007100e-01   1.40904322e-01
   -3.51901874e-02   8.07275325e-02   1.09597817e-02  -2.76857102e-03
    3.21411937e-02   2.11234558e-02   2.21219230e-02  -9.64475647e-02
    1.28499866e-01  -2.32732054e-02   1.22369967e-01  -2.37366855e-01
    7.43315667e-02  -5.40643372e-02  -3.93253677e-02   2.84477277e-03
    9.68252867e-03  -5.14560561e-05  -4.07368317e-02  -2.68205744e-03
    1.23449631e-01  -3.70667987e-02   8.02015290e-02   7.38517987e-03
   -1.55605435e-01   6.90959394e-02  -9.79714617e-02  -1.63167082e-02
   -9.28533301e-02   9.92244408e-02  -5.91525398e-02  -1.15624070e-01
    3.33569851e-03   3.21331292e-01   2.83122901e-03  -8.59396830e-02
   -2.78891325e-01  -1.97359012e-03   8.56173132e-03   4.03305404e-02
    1.56095612e-03   1.38616443e-01   1.03200153e-01   2.66493652e-02
    1.19524017e-01  -1.65473357e-01   3.80755551e-02  -7.34731602e-03
    2.65281014e-02   4.98776734e-02   2.79466677e-02  -3.88022028e-02
    7.26109296e-02  -6.04184382e-02  -7.25998580e-02   5.06432615e-02
   -4.96999137e-02   1.18302144e-01   8.90203118e-02  -5.81578165e-02
    1.02084234e-01   7.57777393e-02  -9.05318465e-03   4.72530760e-02
   -4.98152263e-02  -6.24167509e-02   5.94433844e-02   3.41376266e-03
    6.60767639e-03  -9.23319682e-02  -6.70323223e-02   4.40597795e-02
   -2.18763519e-02   1.32832915e-01   6.76512942e-02  -3.69001441e-02
    5.74899130e-02  -1.86466798e-02   4.03219368e-03   7.26963058e-02
   -1.67973861e-02   7.07234293e-02   4.92062047e-02  -8.89770091e-02
   -6.33207336e-02   1.81835130e-01   3.00272349e-02   7.91064501e-02
    4.25065830e-02   7.04212636e-02  -3.45968120e-02  -1.61775518e-02
    2.40377150e-02   1.24002919e-01   8.01019445e-02  -1.23444684e-01]]
Recognized as dtilak
```

Deployment

Deployment

Deployment


Dev Environment
https://address
Docker
TF Serving
Deep Learning
Docker
Flask
Machine Learning,
Logics
And
Web Page details

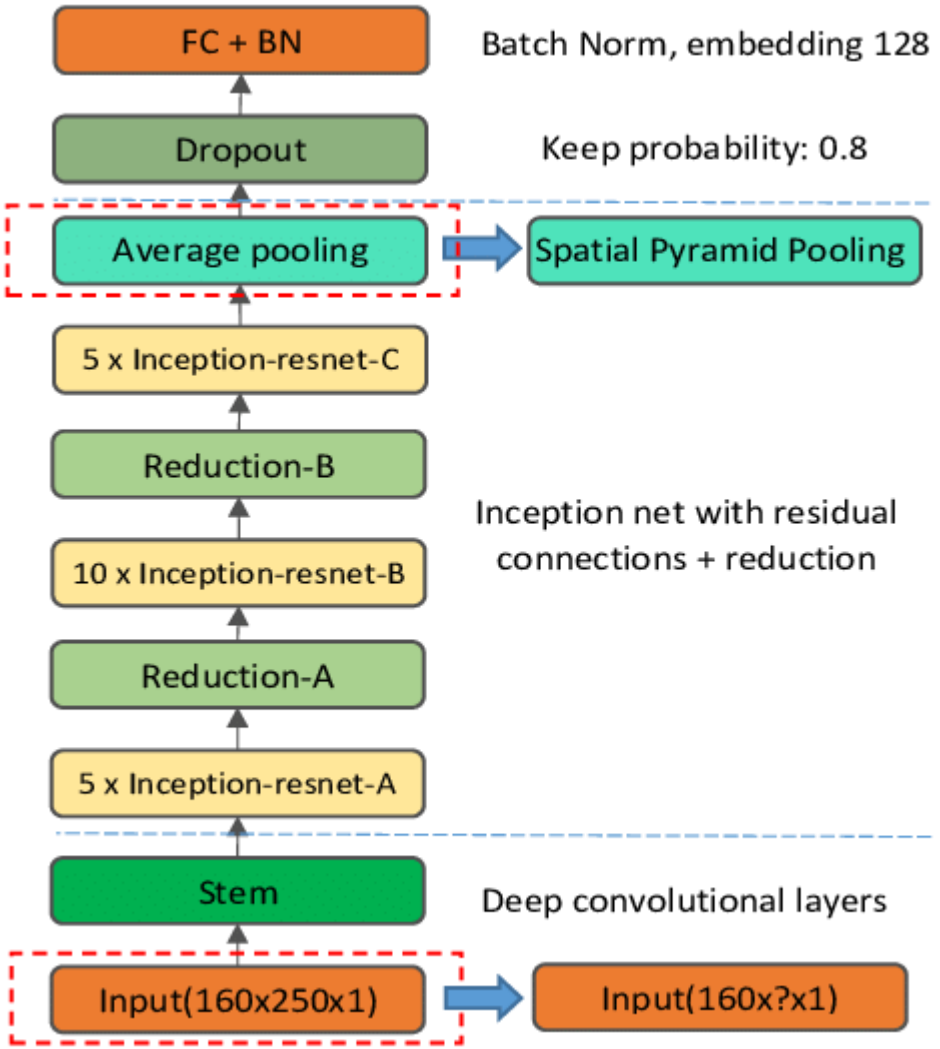
Next Steps

Data Collection

- Webpage for Data Collection
- More Face Images
- Varied angles and lighting conditions
- Data collected over time

Better Model

Inception
to
Inception ResNet



That’s it!

Appendix

Triplet Loss Extra

TRIPLET LOSS

- $\left\|f(x_i^{anchor}) - f(x_i^{positive})\right\|_2^2 + \alpha < \left\|f(x_i^{anchor}) - f(x_i^{negative})\right\|_2^2$

For all possible triplets in training set.

Assuming that we have N triplets sets. Than, the loss function to minimize become:

- $\sum_i^N \left[\left\|f(x_i^{anchor}) - f(x_i^{positive})\right\|_2^2 - \left\|f(x_i^{anchor}) - f(x_i^{negative})\right\|_2^2 + \alpha\right]$

Triplet Loss Extra

TRIPLET SET

- Generating all possible triplets would result in many triplets that are easily satisfied. These triplets would not contribute to the training and result in slower convergence .
- In order to ensure fast convergence it is crucial to select triplets that violate the triplet constraint .
- This means that – given x_i^{anchor} the optimal selection is :
  - An 'hard positive' x_i^p such that $\left\|f(x_i^{anchor}) - f(x_i^p)\right\|_2^2$ is maximal
  - An 'hard negative' x_i^n such that $\left\|f(x_i^{anchor}) - f(x_i^n)\right\|_2^2$ is minimal