FaceNet

A Unified Embedding for Face Recognition and Clustering

Florian Schroff*, Dmitry Kalenichenko* & James Philbin* [*Google Inc.]

BITS F312 Neural Networks & Fuzzy Logic
Paper ID 51

Nikhil Kumar 2017B5A70658P
Pratyush Goel 2017B5A70899P
Aashya Kumar 2017B5A30981P

Agenda

- Paper Description
- Looking at the Data
- Network Architectures
  - Zeiler & Fergus (Z&F) Architecture
  - Inception Architecture
- Our Results
  - Z&F
  - Inception
  - SVM & KNN
- Conclusion

Paper Description

- Layout of Paper
- Previous Approaches
- What Does FaceNet Offer?
- Background Concepts

Layout of Paper

- Aim
  - To present a system that directly learns a mapping from face images to a compact Euclidean space.
  - Distances in the Euclidean space correspond to a measure of face similarity.
  - Tasks such as face recognition, verification and clustering can be easily implemented with this.
- Methodology
  - Use a deep convolutional network trained to directly optimize the embedding instead of an intermediate bottleneck layer.
  - To train, use triplets of roughly aligned matching / non-matching face patches which are generated using a novel online triplet mining method.

Layout of Paper

- Final outcome
  - System gives a new record accuracy of 99.63% on the widely used Labeled Faces in the Wild (LFW) dataset.
  - On YouTube Faces DB it achieves 95.12%. This system cuts the error rate in comparison to the best published result* by 30% on both datasets.

* Reference: (Y. Sun, X. Wang, and X. Tang. Deeply learned face representations are sparse, selective, and robust. CoRR, abs/1412.1265, 2014. 1, 2, 5, 8)

Previous Approaches

Approach:

- Deep networks are implemented which use a classification layer.
- These networks are trained over a set of known face identities.
- Intermediate bottle-neck layer is used to generalize recognition even for the set of identities that are not used in training.

Disadvantages:

- These have indirectness and inefficiency in terms of generalization to new faces through bottle-neck representation.
- If we add a new face to the data, we'll have to retrain the whole model by adding an extra node in the softmax layer.

What Does FaceNet Offer?

Approach:

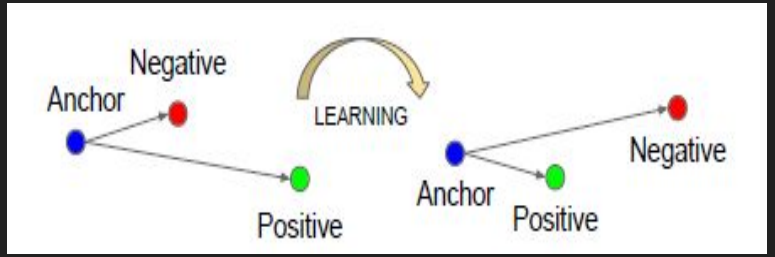
- Learning a Euclidean embedding per image using a deep convolutional network.
- The network is trained such that the squared L2 distances in the embedding space directly correspond to face similarity, using a special triplet loss.
- Face verification: Thresholding distance between the two embeddings
- Recognition: Clustering of embeddings / classification using SVM or KNN

Advantages:

- Network trained to directly optimize embedding itself, rather than an intermediate bottleneck layer as in previous deep learning approaches.
- Greater representational efficiency: State-of-the-art face recognition performance can be achieved using only 128-bytes per face.

Background Concepts – Triplet Loss

- The embedding is represented by $f(x) \in R^d$. It embeds an image x into a d-dimensional Euclidean space (here d = 128).
- Here we want to ensure that an image x_i^a (anchor) of a specific person is closer to all other images x_i^p (positive) of the same person than it is to any image x_i^n (negative) of any other person.



Triplet loss minimizes distance between anchor and positive while maximizing distance between anchor and negative

Background Concepts – Triplet Loss

- Therefore, we want:

$$|| f(x_i^a) - f(x_i^p) ||_2^2 + \alpha < || f(x_i^a) - f(x_i^n) ||_2^2 , \forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in T \qquad (1)$$

- α is a margin that is enforced between positive and negative pairs. T is the set of all possible triplets in the training set and has cardinality N.
- The loss that is being minimized is then

$$L = \sum_{i=1}^{N} [|| f(x_i^a) - f(x_i^p) ||_2^2 - || f(x_i^a) - f(x_i^n) ||_2^2 + \alpha]_+ \qquad (2)$$

Background Concepts – Triplet Selection

- In order to ensure fast convergence it is crucial to select triplets that violate the triplet constraint in Eq. (1)
- This means that given x_i^a, we want to select
  - x_i^p (hard positive) =

$$\text{argmax} \; || f(x_i^a) - f(x_i^p) ||_2^2$$

  - x_i^n (hard negative) =

$$\text{argmin} \; || f(x_i^a) - f(x_i^n) ||_2^2$$

Background Concepts – Triplet Selection

- In short, the above equations mean
  - Select a positive that is most like a negative (hard/difficult positive)
  - Select a negative that is most like a positive (hard/difficult negative)
- This helps the network learn faster because we are feeding it triplets that actually require significant distinguishing (hence more room to learn)
- Triplets are generated online and using large mini-batches.
  - Small mini-batches will not contain hard positives / hard negatives that are hard / difficult enough
  - Large mini-batches take too many epochs to train
  - We used a batch size of 200 which worked well – Tradeoff between difficulty and time for training

Background Concepts – Triplet Selection

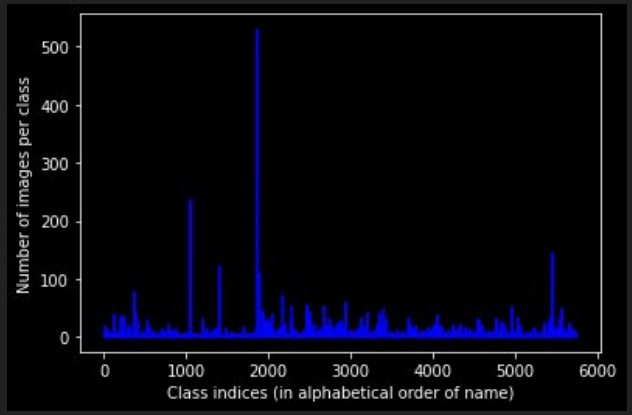
- Computation of the argmin and argmax is done within a mini-batch
  - Our mini-batches had 200 images each
  - Training set contains approx. 6000 images
  - $\therefore$ It is infeasible to compute argmin and argmax across the whole training set
- To have a meaningful representation of the anchor positive distances
  - There should be enough number of images of one individual in a mini-batch
  - This is why we chose only individuals who had 3 or more images

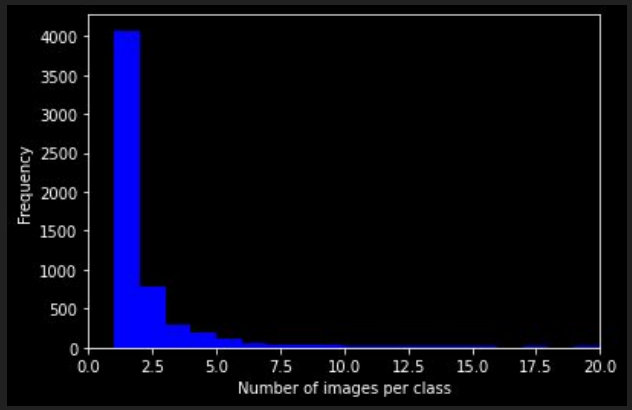
Looking at the Data

- Labeled Faces in the Wild
- LFW-a Dataset
- Using a Subset of the Data

Labelled Faces in the Wild

- Used Labeled Faces in the Wild Dataset
- Contains approx. 13k close-cropped images of 5749 unique individuals
  - Most individuals had only a single image
  - Mean number of images per class = 2.3
  - Median number of images per class = 1
  - Only the top 5% of individuals had 6 images or more
  - Weird anomaly: George W. Bush had 500+ images (note the spike in the first image to the right)
- Faces detected using Viola-Jones detector





Labelled Faces in the Wild – Subset

- Most classes have 2 or 3 images each (ref. prev. slide)
- Decided to use only classes with 3 or more images for training
  - 6352 images
  - 901 unique individuals
- Used a version of the dataset with aligned faces (LFW-a dataset*)

* Reference: Lior Wolf, Tal Hassner, and Yaniv Taigman, *Effective Face Recognition by Combining Multiple Descriptors and Learned Background Statistics*, IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI), 33(10), Oct. 2011

Network Architectures

- Zeiler & Fergus (Z&F) Arch
- Z&F Arch – Modified
- Inception Arch
- Inception Arch – Modified

Z&F Architecture

- Six sets of convolutional layers
  - Same padding
  - ReLU activation
  - Interspersed with max-pooling layers
- Sets 2 to 6 have two conv layers each:
  - Conv A layer: 1 x 1 convolution
  - Conv layer: 3 x 3 convolution
- Flattening layer after all convolutional sets
- L2 normalization so that output embedding is on unit n-sphere (n=128)

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

Modified version of the ZF architecture used by Schroff et. al.
Reference: Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815-823).

Z&F Architecture – Modified

- Changed input shape of first layer to (96, 96, 1) from (220, 220, 1) to allow for faster training
- Replaced same padding with valid padding to reduce size of axis=0 and axis=1 of tensors in hidden conv layers
- Replaced rnorm (local response normalization) with batch norm as rnorm is considered outdated*
- Replaced concat layer with a flatten layer

* Refer Francois Chollet's statement here: https://groups.google.com/g/keras-users/c/8Ncd0dpuPNE

Z&F Architecture – Modified

Model: "zeiler_fergus"

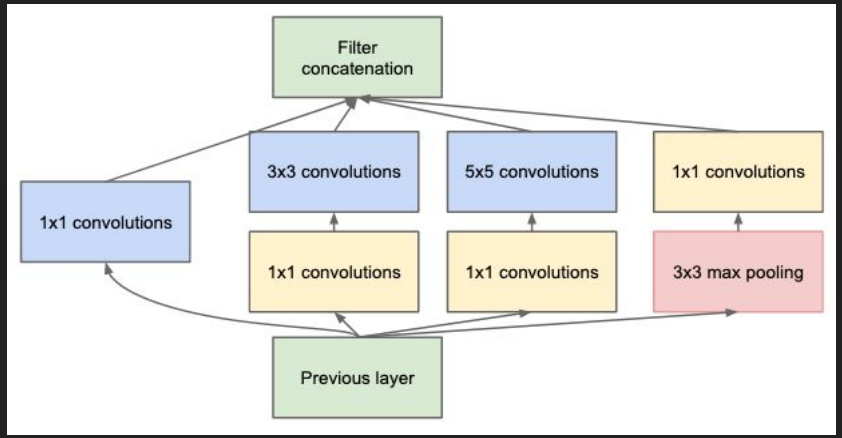
Layer (type)	Output Shape	Param #
conv1 (Conv2D)	(None, 48, 48, 64)	3200
pool1 (MaxPooling2D)	(None, 24, 24, 64)	0
norm1 (BatchNormalization)	(None, 24, 24, 64)	256
conv2a (Conv2D)	(None, 24, 24, 64)	4160
conv2 (Conv2D)	(None, 24, 24, 192)	110784
norm2 (BatchNormalization)	(None, 24, 24, 192)	768
pool2 (MaxPooling2D)	(None, 12, 12, 192)	0
conv3a (Conv2D)	(None, 12, 12, 192)	37056
conv3 (Conv2D)	(None, 12, 12, 384)	663936
pool3 (MaxPooling2D)	(None, 6, 6, 384)	0
conv4a (Conv2D)	(None, 6, 6, 384)	147840
conv4 (Conv2D)	(None, 6, 6, 256)	884992
conv5a (Conv2D)	(None, 6, 6, 256)	65792
conv5 (Conv2D)	(None, 6, 6, 256)	590080
conv6a (Conv2D)	(None, 6, 6, 256)	65792
conv6 (Conv2D)	(None, 6, 6, 256)	590080
pool4 (MaxPooling2D)	(None, 3, 3, 256)	0
flatten (Flatten)	(None, 2304)	0
fc1 (Dense)	(None, 1024)	2360320
drop1 (Dropout)	(None, 1024)	0
fc2 (Dense)	(None, 1024)	1049600
drop2 (Dropout)	(None, 1024)	0
fc3 (Dense)	(None, 128)	131200
l2 (Lambda)	(None, 128)	0

Total params: 6,705,856
Trainable params: 6,705,344
Non-trainable params: 512

Inception Architecture

- 11 inception blocks, each block contains in parallel:
  - 1 x 1 convolution
  - 3 x 3 convolution
  - 5 x 5 convolution
  - 3 x 3 max-pooling
- Parallel layer outputs are concatenated
- L2 pooling replaced by max pooling
- L2 normalization so that output embedding is on unit n-sphere (n=128)
- We made no modifications apart from changing input shape to (96, 96, 1) from (220, 220, 1) to allow for faster training

type	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj (p)	params	FLOPS
conv1 (7×7×3, 2)	112×112×64	1							9K	119M
max pool + norm	56×56×64	0						m 3×3, 2		
inception (2)	56×56×192	2		64	192				115K	360M
norm + max pool	28×28×192	0						m 3×3, 2		
inception (3a)	28×28×256	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	28×28×320	2	64	96	128	32	64	L_2, 64p	228K	179M
inception (3c)	14×14×640	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	14×14×640	2	256	96	192	32	64	L_2, 128p	545K	107M
inception (4b)	14×14×640	2	224	112	224	32	64	L_2, 128p	595K	117M
inception (4c)	14×14×640	2	192	128	256	32	64	L_2, 128p	654K	128M
inception (4d)	14×14×640	2	160	144	288	32	64	L_2, 128p	722K	142M
inception (4e)	7×7×1024	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	7×7×1024	2	384	192	384	48	128	L_2, 128p	1.6M	78M
inception (5b)	7×7×1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1×1×1024	0								
fully conn	1×1×128	1							131K	0.1M
L2 normalization	1×1×128	0								
total									7.5M	1.6B



Inception block
Reference: Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

Code Structure

- Dataset generation
  - Used `ImageDataGenerator` to generate and augment training dataset.
- Model training
  - Trained Z&F model and Inception model to generate 128D embeddings.
- Classification
  - Fit the training embeddings to make clusters of classes using KNN and SVM.
- Prediction / Verification
  - Given a new image generate its embeddings and classify the 128D point using KNN or SVM to perform prediction and verification.

Our Results

- Zeiler & Fergus (Z&F)
- Inception
- SVM
- KNN

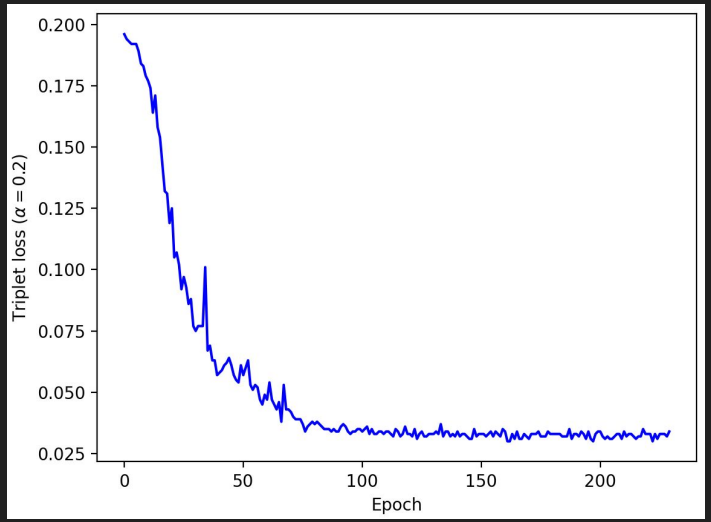
Z&F – Hyperparameters

- Loss: Triplet semi-hard loss
- Optimizer: SGD
- Learning rate: 0.05 – 0.00015 (reduced on loss plateauing)
- Batch size: 200
- Epochs: 230

Z&F – Learning/Loss vs Epoch

- Used a margin $\alpha = 0.2$
- Triplet loss was approx. 0.2 at the start (network perfectly unable to distinguish)
- Trained for 230 epochs

Epoch	Loss
1	0.196
10	0.179
25	0.092
50	0.061
100	0.031
230	0.034

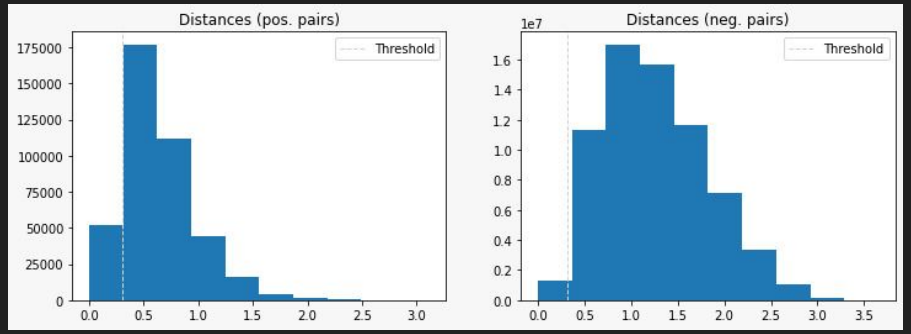


Z&F Architecture – Visualizing Embeddings

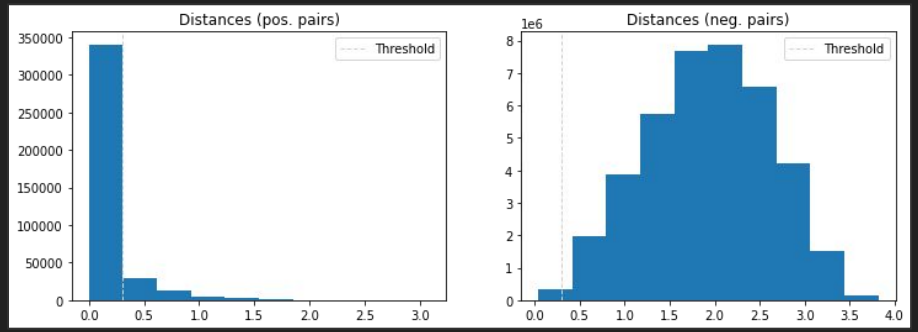
- 128D training embeddings projected to 2D using UMAP
- Some cluster formation is visible
- However, given the constraints of time and hardware, we were unable to separate the embeddings into more clusters
- The image on the right is for 901 classes, however on performing a similar dimensionality on 3 classes, clear cluster formation is visible (upcoming slides)

Z&F – Histograms of ++ and +- Pairs

- The histograms on the right show the distribution of distance between:
  - Anchor & positive (++) pairs (left fig.)
  - Anchor & negative (+-) pairs (right fig.)
- The white dotted line parallel to the y-axis is the threshold
- We want:
  - ++ distance to be small (peak at lower x)
  - +- distance to be large (peak at higher x)
- As the model learns to distinguish:
  - ++ peak shifts leftwards
  - +- peak shift rightwards



After 72 epochs (++ peak is still above threshold)



After 230 epochs (++ peak has moved below threshold)

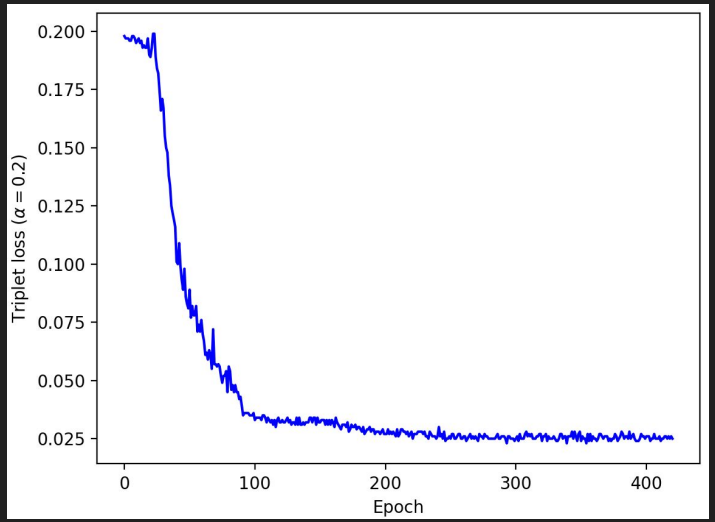
Inception – Hyperparameters

- Loss: Triplet semi-hard loss
- Optimizer: SGD
- Learning rate: 0.01 – 0.00003 (reduced on loss plateauing)
- Batch size: 200
- Epochs: 421

Inception – Learning/Loss vs Epoch

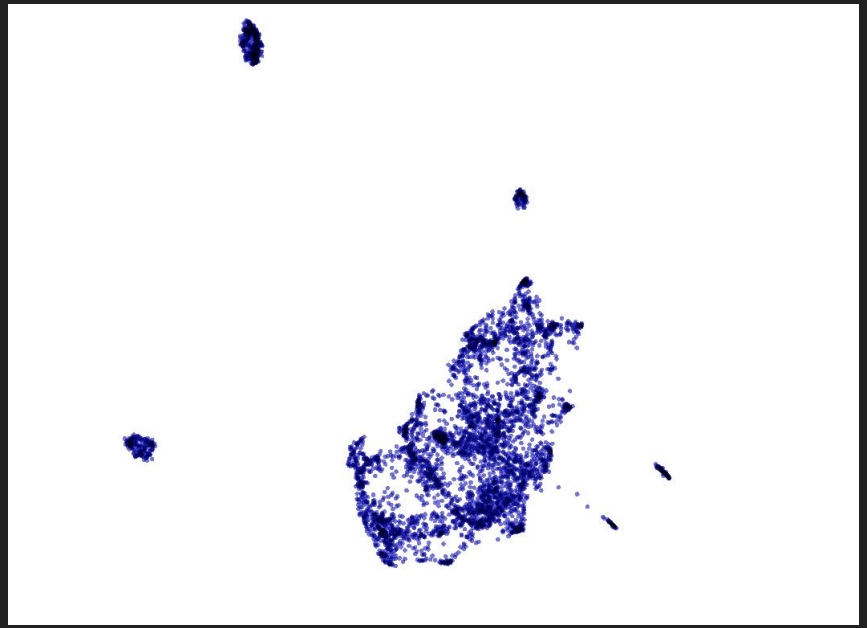
- Used a margin $\alpha = 0.2$
- Triplet loss was approx. 0.2 at the start (network perfectly unable to distinguish)
- Trained for 421 epochs

Epoch	Loss
1	0.198
50	0.081
100	0.036
200	0.027
300	0.024
421	0.025



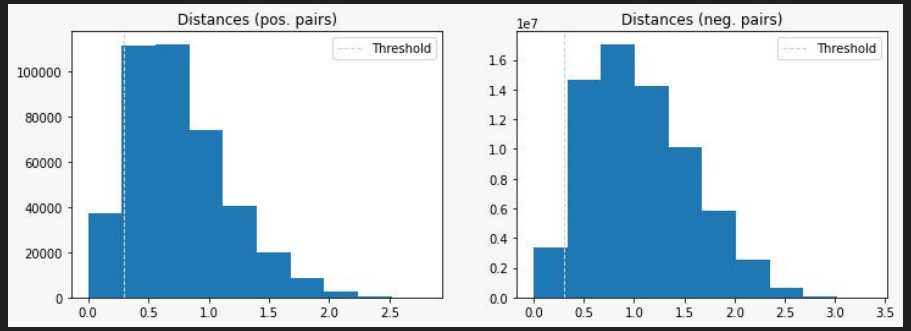
Inception Architecture – Visualizing Embeddings

- 128D training embeddings projected to 2D using UMAP
- Some cluster formation is visible
- However, given the constraints of time and hardware, we were unable to separate the embeddings into more clusters
- The image on the right is for 901 classes, however on performing a similar dimensionality on 3 classes, clear cluster formation is visible (upcoming slides)

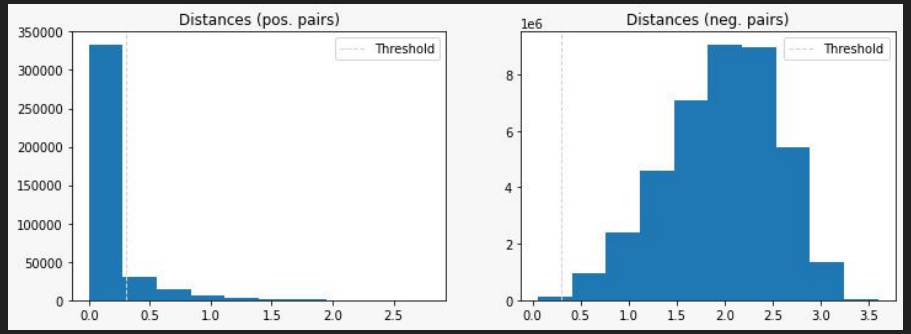


Inception – Histograms of ++ and +- Pairs

- The histograms on the right show the distribution of distance between:
  - Anchor & positive (++) pairs (left fig.)
  - Anchor & negative (+-) pairs (right fig.)
- The white dotted line parallel to the y-axis is the threshold
- We want:
  - ++ distance to be small (peak at lower x)
  - +- distance to be large (peak at higher x)
- As the model learns to distinguish:
  - ++ peak shifts leftwards
  - +- peak shift rightwards



After 90 epochs (++ peak still above threshold)



After 400 epochs (++ peak has moved below threshold)

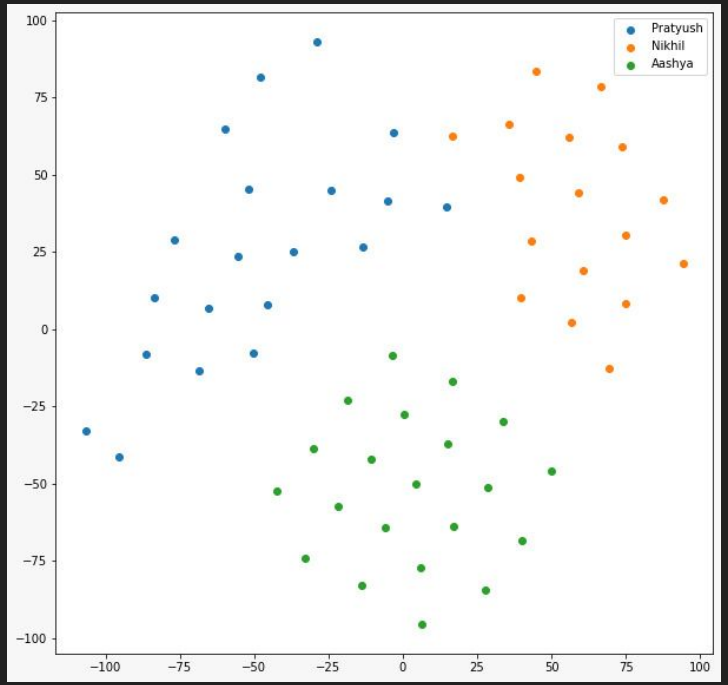
SVM

- Trained an SVM using the embeddings generated from the above networks
- Results on 10-fold cross-validation on reduced LFW-a (6352 images):
  - Accuracy = 49.83% using Z&F (input shape = (96, 96)) 128D embeddings
  - Accuracy = 51.64% using Inception (input shape = (96, 96)) 128D embeddings
- Results are comparable to the NNS2 model (accuracy = 51.9%, input shape=(140, 116)) in the FaceNet paper
- Used default values of hyperparameters
  - Gave reasonably good results
  - Scope for improvement on performing grid search

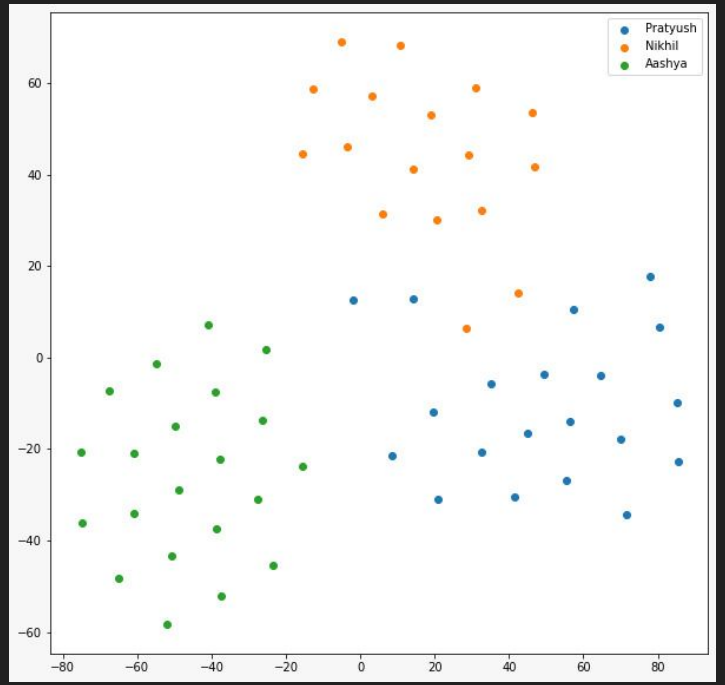
KNN

- Trained a KNN model using the embeddings generated from the above networks
- Results on 10-fold cross-validation on reduced LFW-a (6352 images):
  - Accuracy = 42.24% (k = 5) using Z&F (input shape = (96, 96)) 128D embeddings
  - Accuracy = 48.64% (k = 1) using Inception (input shape = (96, 96)) 128D embeddings
- Results are comparable to the NNS2 model (accuracy = 51.9%, input shape=(140, 116)) in the FaceNet paper
- Used default values of hyperparameters
  - Gave reasonably good results
  - Scope for improvement on performing grid search

Results on Our Images – tSNE (Cluster Formation)



Projecting 128D Z&F embeddings to 2D
Clusters: Nikhil, Pratyush, Aashya



Projecting 128D Inception embeddings to 2D
Clusters: Nikhil, Pratyush, Aashya

Results on Our Images – tSNE (Cluster Formation)

- Z&F and Inception generate 128D embeddings
- tSNE (t-distributed stochastic neighbour embedding) is a method of dimensionality reduction
- We use tSNE to convert the 128D embeddings to 2D
- Explanation of images on prev. slide:
  - Generated 128D embeddings of our (Nikhil, Pratyush, Aashya) images
  - Reduced 128D embeddings to 2D
  - Clear cluster formation of embeddings of each individual
- Generalizing well: Results on the prev. Slide are for images the network has not seen before

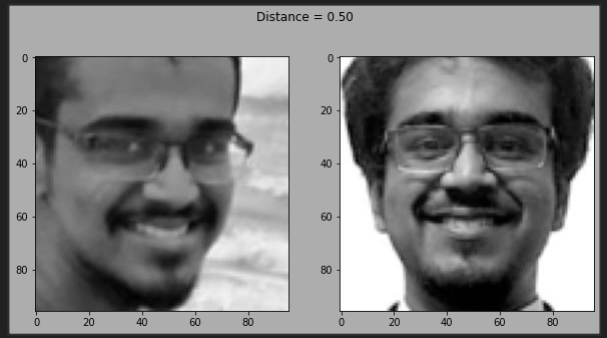
Results on Our Images – Classifier Confidence

- Classifier able to distinguish between images it has not been trained on (fig. on right)
- Able to recognize in an image with multiple individuals (Nikhil, Pratyush)
  - Nikhil with 0.48 confidence
  - Pratyush with 0.49 confidence

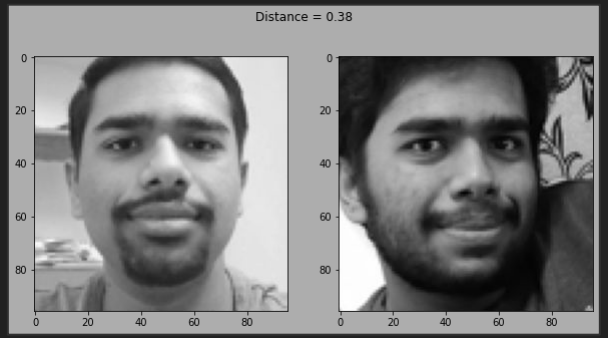
Results on Our Images – Distances

Distance between embeddings of same individual

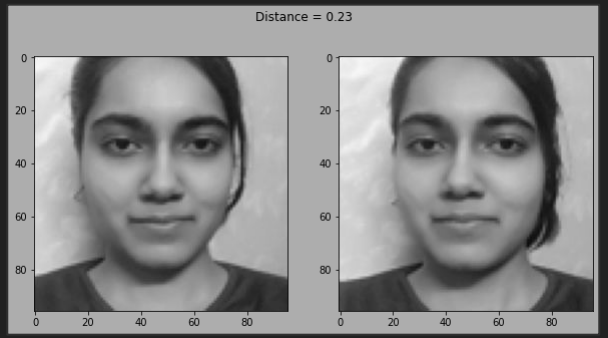
Nikhil – Nikhil (distance = 0.5)



Pratyush – Pratyush (distance = 0.38)



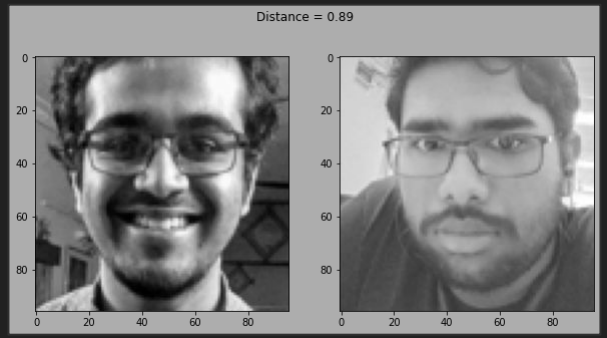
Aashya – Aashya (distance = 0.23)



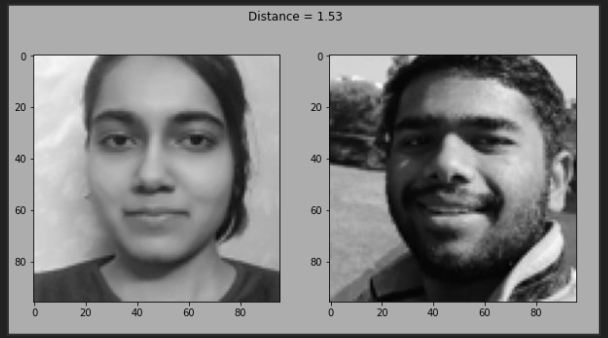
Results on Our Images – Distances

Distance between embeddings of different individuals

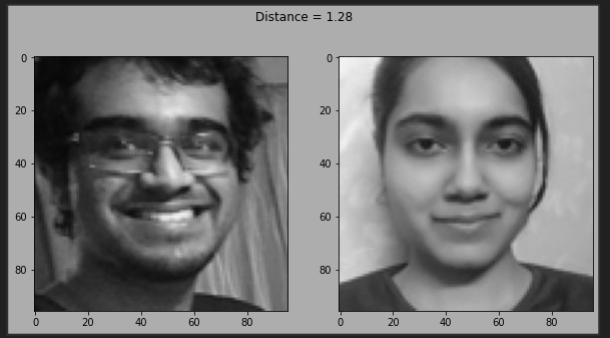
Nikhil – Pratyush (distance = 0.89)



Aashya – Pratyush (distance = 1.53)



Nikhil – Aashya (distance = 1.28)



Results on Our Images – Distances

Summary matrix of distances between embeddings of individuals

	Nikhil	**Pratyush**	**Aashya**
Nikhil	0.50	0.89	1.28
Pratyush	0.89	0.38	1.53
Aashya	1.28	1.53	0.23

Note how the distances between embeddings of the same individual are smaller than distances between embeddings of different individuals in all cases.

Conclusion

- Challenges Faced
- Future Scope
- Learning Outcomes

Challenges Faced

- Reduced input dimension of (96, 96) didn't generalize well on the LFW-a dataset with 901 classes. However, it did perform well on our dataset with 3 classes.
- Can be solved using:
  - More epochs of training
  - A larger dataset (we used a subset of the original LFW dataset)
- Took approx. 24 hours to train 20 epochs using (220, 220). Due to hardware constraints, had to adopt some simplifications in the model, described in prev. Slides.
- Loss started stagnating after a few epochs. Solved using the following approaches:
  - Learning rate scheduling for Z&F architecture.
  - Reducing learning rate on loss plateauing for Inception architecture

Future Scope

- With more powerful hardware, it might be possible to use the entire training set to generate hard positives and hard negatives (instead of constraining the triplet mining to the mini-batch).
- Serialize the embeddings of faces (128 bytes each) instead of storing people's faces in a database
    - Saves space as images take up kilobytes of data per image
    - Addresses privacy concerns (no need to store images once embedding has been generated)

Learning Outcomes

- Implementing a research paper in code.
- Learning new loss functions (triplet loss).
- Bridging the gap between theory learnt in class and implementing models in code by reading documentation.
- Learned how to do prediction and verification.

Thank You!