HW2P2 Bootcamp

Face Classification and Verification

Logistics

- HW2P2 has two parts and their corresponding Kaggle competitions. They are both due on 21st October, 2021 11:59 EDT.
- HW2P2 is not as easy as HW1P2 so please start early. Model training and convergence itself will take a lot of time.
- The public leaderboard is based only on **30%** of the test data unlike HW1P2 which was based on **70%** of the test data.
- Ensure that your models are not overfit because a high score on public leaderboard might not necessarily contribute to a high score on the private leaderboard.
- The baseline architecture is already on Piazza and ***it will not*** necessarily help you cross B cutoff.
- You will not be provided with a base notebook to edit, we will provide only some code snippets to help you implement the model.


TIRED.
JUST TIRED.

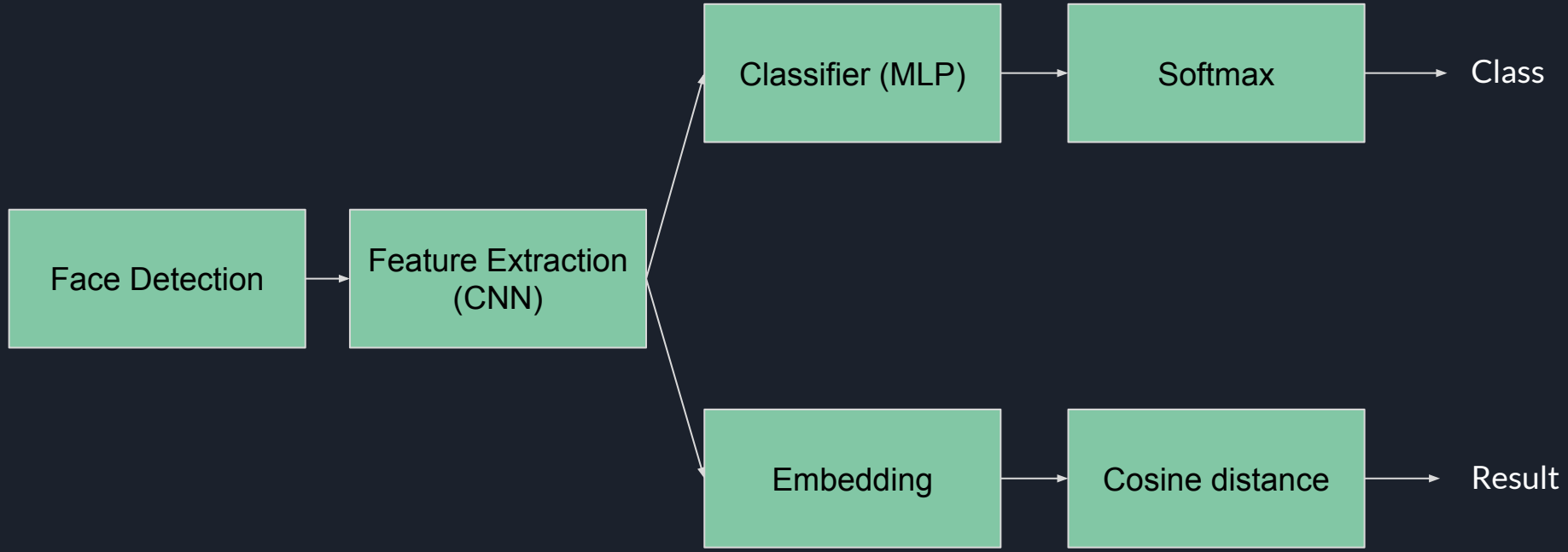

People with no idea about AI, telling me my AI will destroy the world
Me wondering why my neural network is classifying a cat as a dog..
Dog

Problem Statement

Convolutional networks are very good feature extractors. We use them for extracting facial features which can then be fed to any other classification network.

1. ***Face Classification***:
   - Extract features from image of the face of a person
   - Develop a network to use these features to classify the image into classes (people in our case)
2. ***Face Verification***:
   - You can use the network developed earlier to do Face Verification. But how?
   - Identify the most important features which capture the unique identity of a person
   - These features form a fixed-length vector called the Embedding
   - In order to do verification, we only need to identify if 2 embeddings are similar using a metric like cosine distance

Overall Workflow



Difference between 2 problem statements

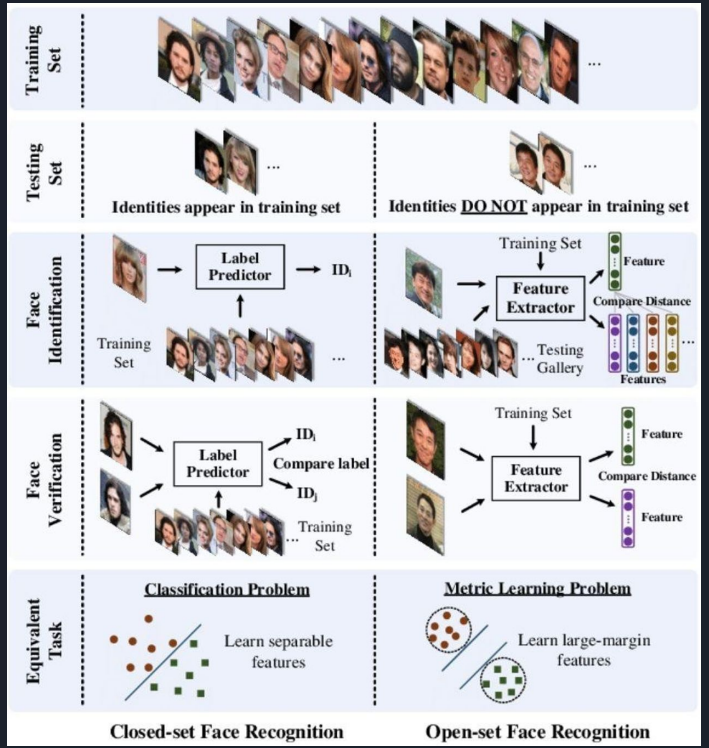
Both problems fundamentally differ in 1 key aspect. Any guesses? It's already on the slides.

Classification is closed set whereas verification is open set.

Closed set meaning the test instances would come from the same classes as the train and validation data.

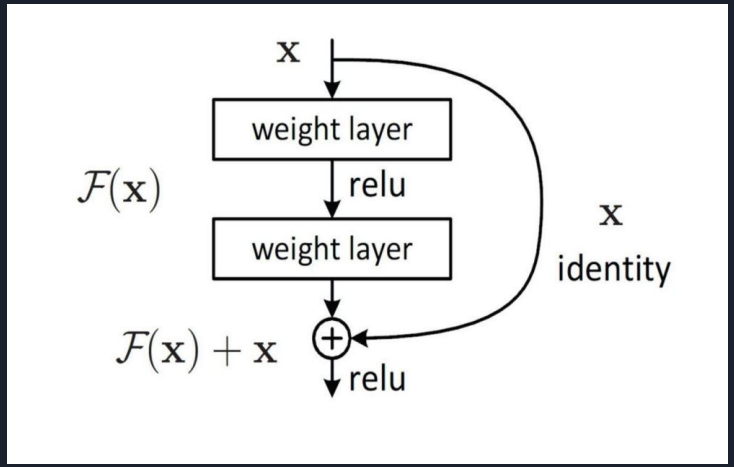
This may not be the case in verification as the model should be able to ascertain if 2 faces belong to the same person or not.

So, what changes?



ResNet

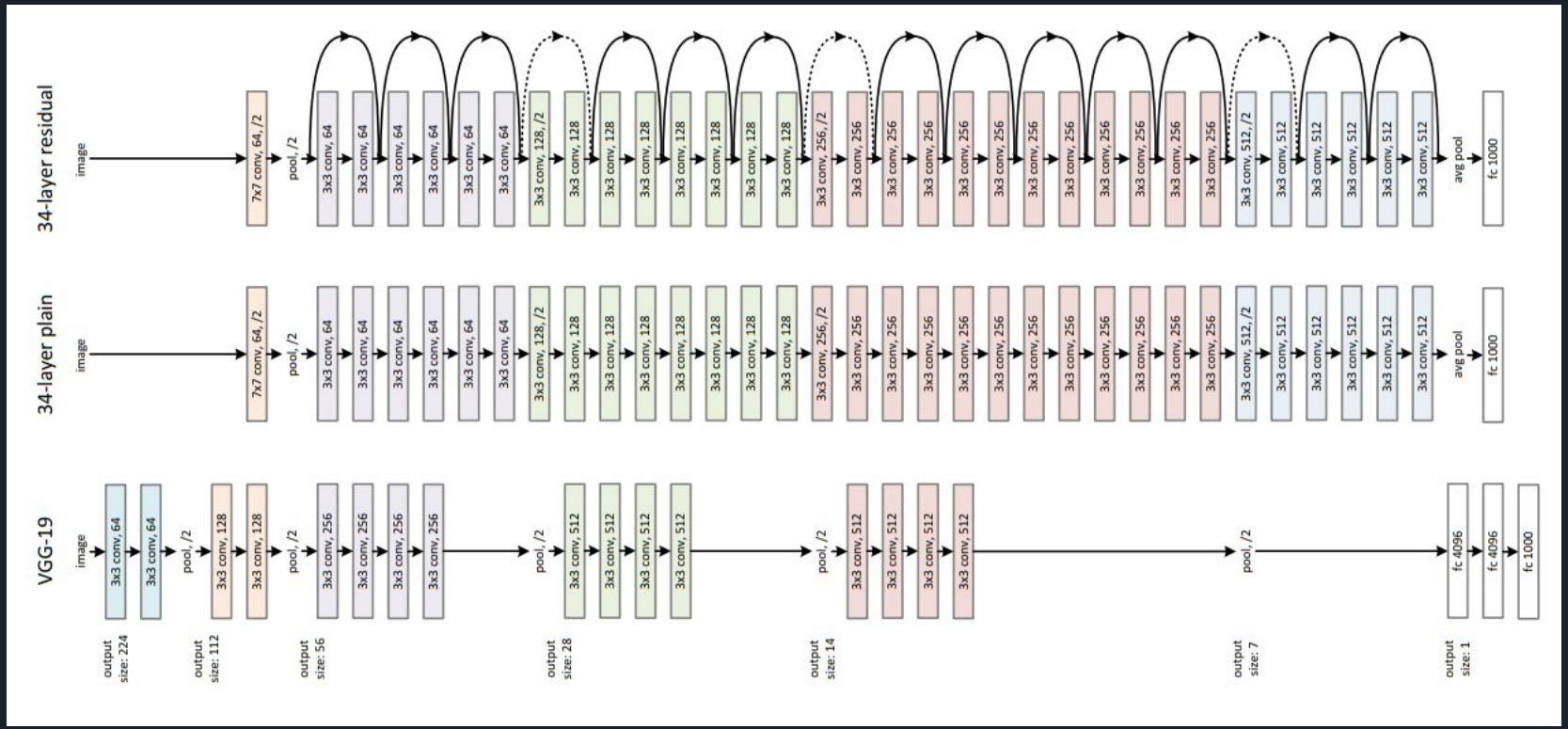
- Introduced in 2015, utilizes bottleneck architectures efficiently and learns them as residual functions
- Easier to optimize and can gain accuracy from increased depth due to skip connections



https://arxiv.org/pdf/1512.03385.pdf

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

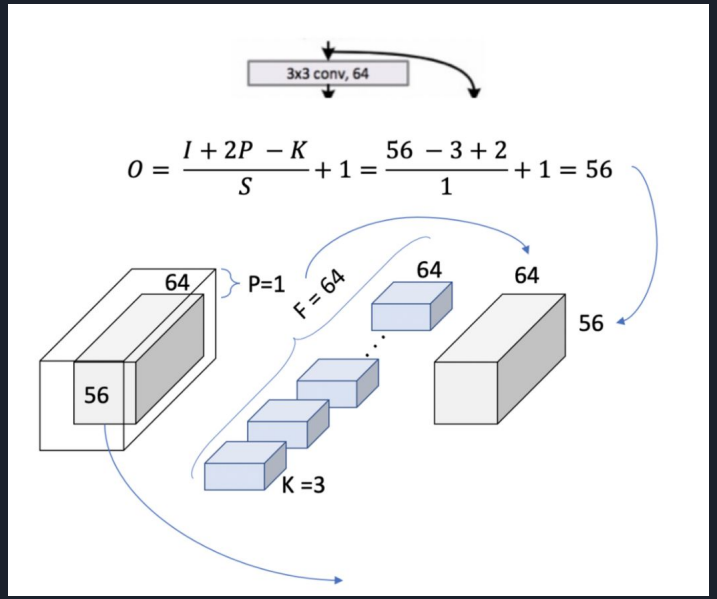
ResNet Architectures

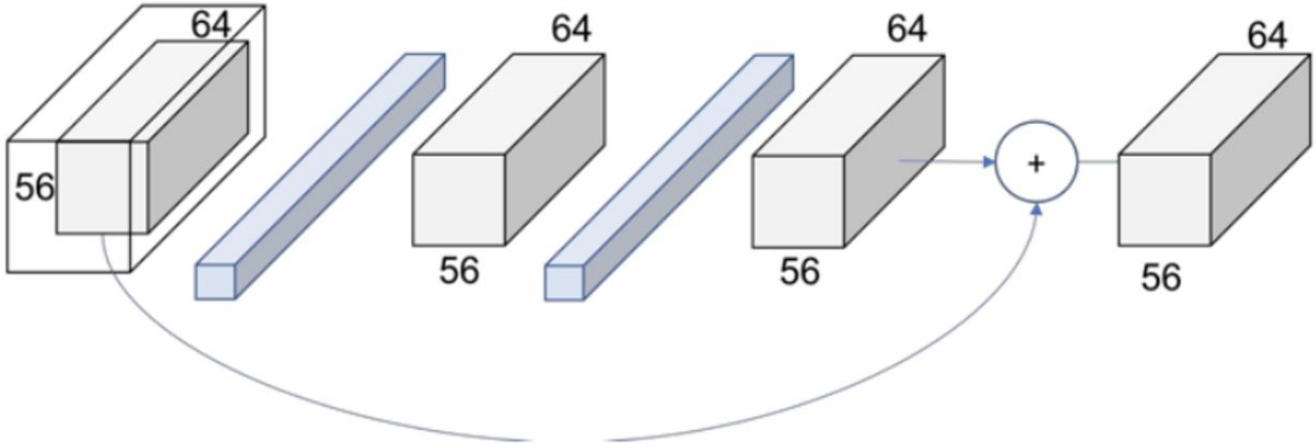


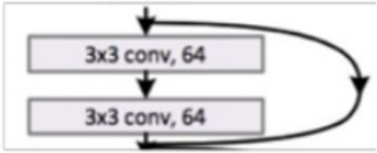
34-Layer ResNet with Skip/Shortcut Connection (Top), 34-Layer Plain Network (Middle), 19-Layer VGG-19 (Bottom)

Block 1: Convolution

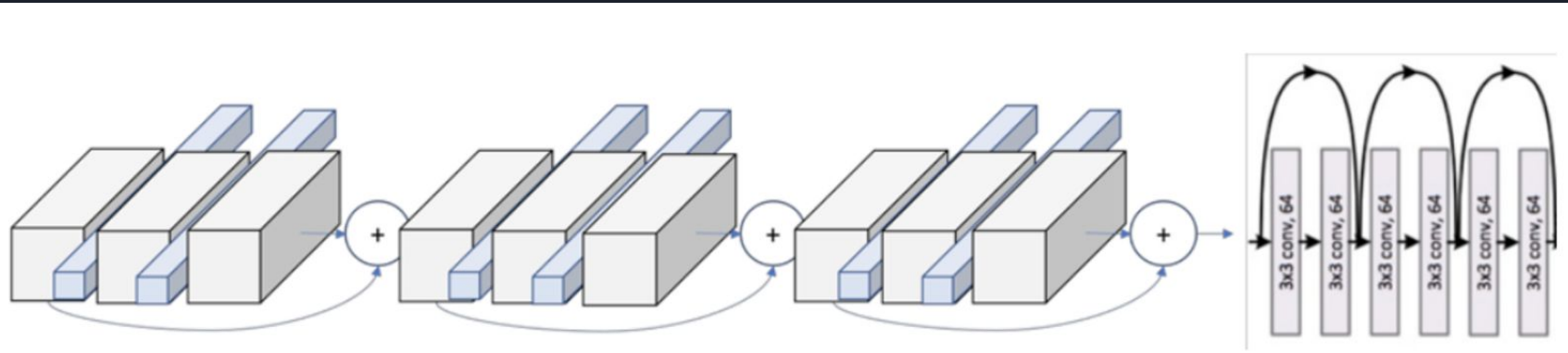
We are replicating the simplified operation for every layer on the paper



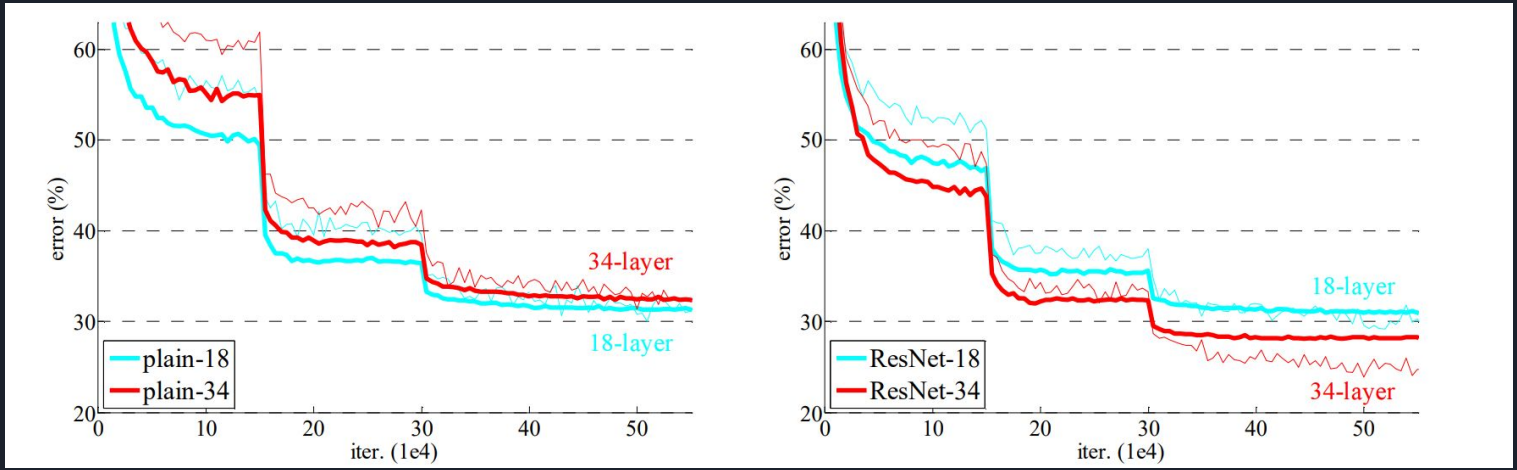

64
56
64
56
64
56
64
56
+


3x3 conv, 64
3x3 conv, 64

We can see how we have the [3 x 3, 64] x 3 times within the layer

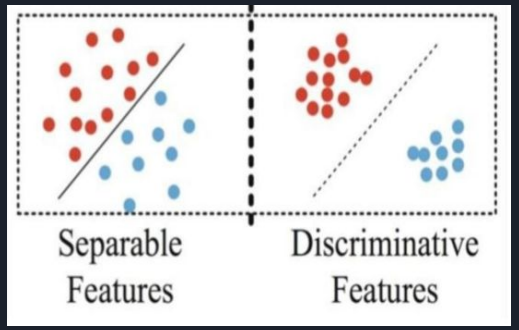


Plain Network v.s. ResNet



Validation Error: 18-Layer and 34-Layer Plain Network (Left), 18-Layer and 34-Layer ResNet (Right)

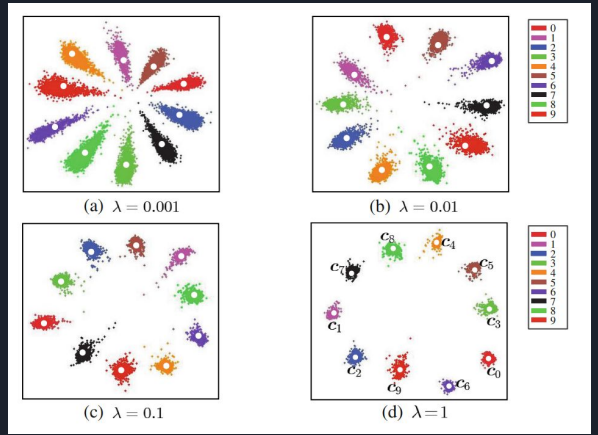
Discriminative Features

- Classification optimizes learning separable features
- Optimally we wish to learn discriminative features
  - Maximum inter class distance
  - 

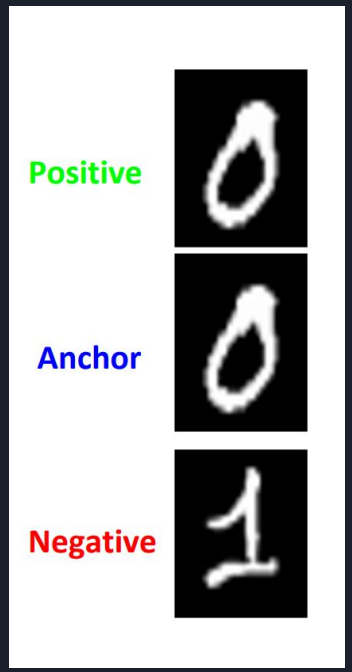
Center Loss

- Tries to minimize the intra class distance by adding a euclidean distance loss term
- If you use this, YOU MUST USE CENTER LOSS FROM THE BEGINNING OF TRAINING CLASSIFICATION!

$$\mathcal{L} = \mathcal{L}_S + \lambda \mathcal{L}_C$$
$$= -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}} + \frac{\lambda}{2} \sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2$$

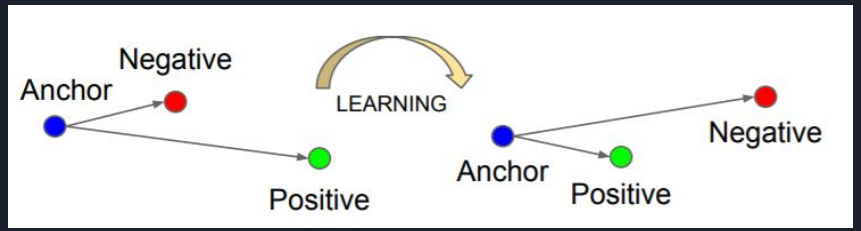


Triplet Loss

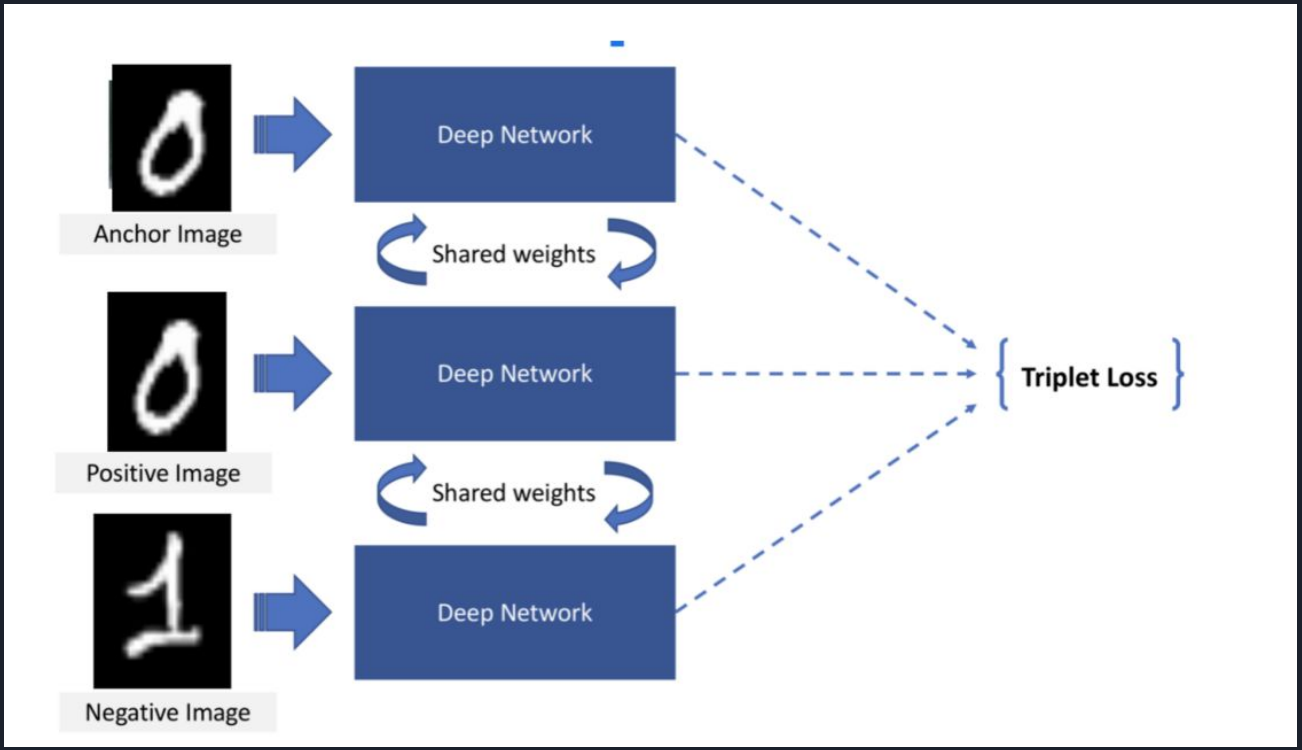


$$\sum_{i}^{N} \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+$$

- Minimizing first term → distance between Anchor and Positive image
- Maximizing second term → distance between Anchor and Negative image



Training with Triplet Loss



Siamese Network

This network does not classify the images into certain categories or labels, rather it only finds out the distance between any two given images.

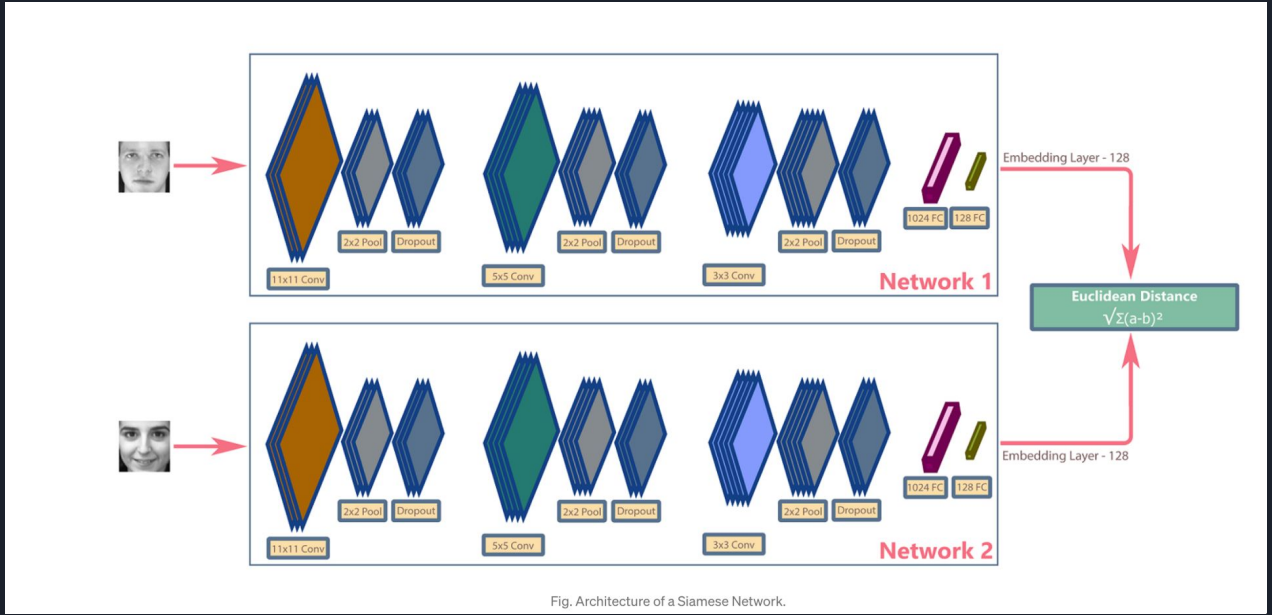


Fig. Architecture of a Siamese Network.

Contrastive Loss

- Contrastive loss is a metric learning loss, which operates on the data points produced by network and their positions relative to each other.
- The model can learn any features regardless of whether similar data points would be located closely to each other or not after the transformation.
- Y term here specifies, whether the two given data points (X_1 and X_2) are similar (Y=0) or dissimilar (Y=1)
- So Ls (loss for similar data points) is just Dw, distance between them, if two data points are labeled as similar,
- we will minimize the euclidean distance between them. Ld, (loss for dissimilar data points) on the other hand, needs some explanation. One may think that for two dissimilar data points we just need to maximize distance between them but with a margin

$$L(W, Y, \vec{X_1}, \vec{X_2}) = (1 - Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2$$

$$D_W(\vec{X_1}, \vec{X_2}) = \|G_W(\vec{X_1}) - G_W(\vec{X_2})\|_2$$

Understanding the Contrastive Loss Function

- In the first figure, we would naturally like to pull black dots closer to the blue dots and push white dots farther away from it.
    - Specifically, we would like to minimize the intra-class distances (blue arrows) and maximize the inter-class distances (red arrows)
- In the second figure, what we would like to achieve is to make sure that for each class/group of similar points (in case of Face Recognition task it would be all the photos of the same person) the maximum intra-class distance is smaller than the minimum
    - This means is that if we define some radius/margin m, all the black dots should fall inside of this margin, and all the white dots — outside
    - This way we would be able to use a nearest neighbour algorithm for new data — if a new data point lies within m distance from other, they are similar/belong to same group/class. Inter-class distance.
    - If Dw is ≥ m, the {m - Dw} expression is negative and the whole right part of the loss function is thus 0 due to max() operation — and the gradient is also 0, i.e. we don't force the dissimilar points farther away than necessary.

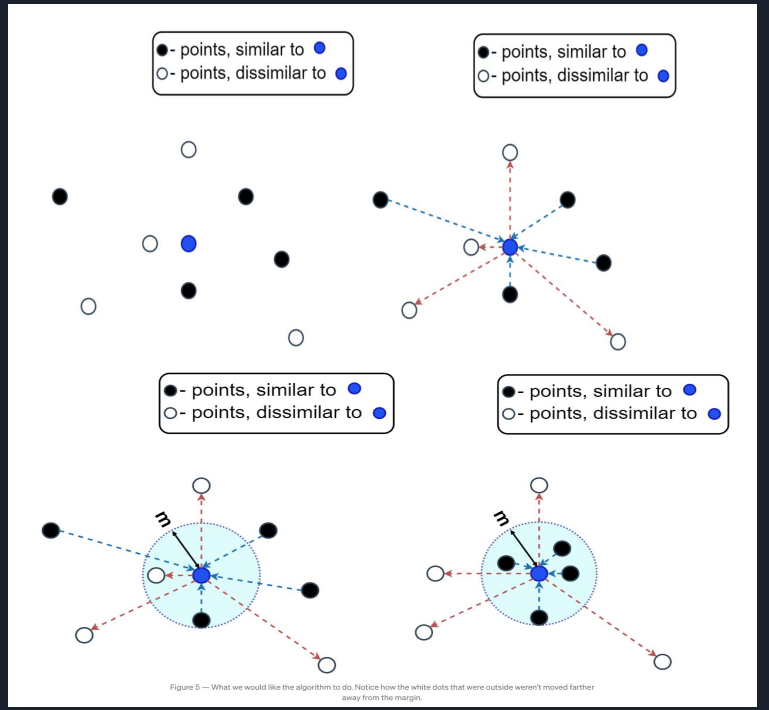
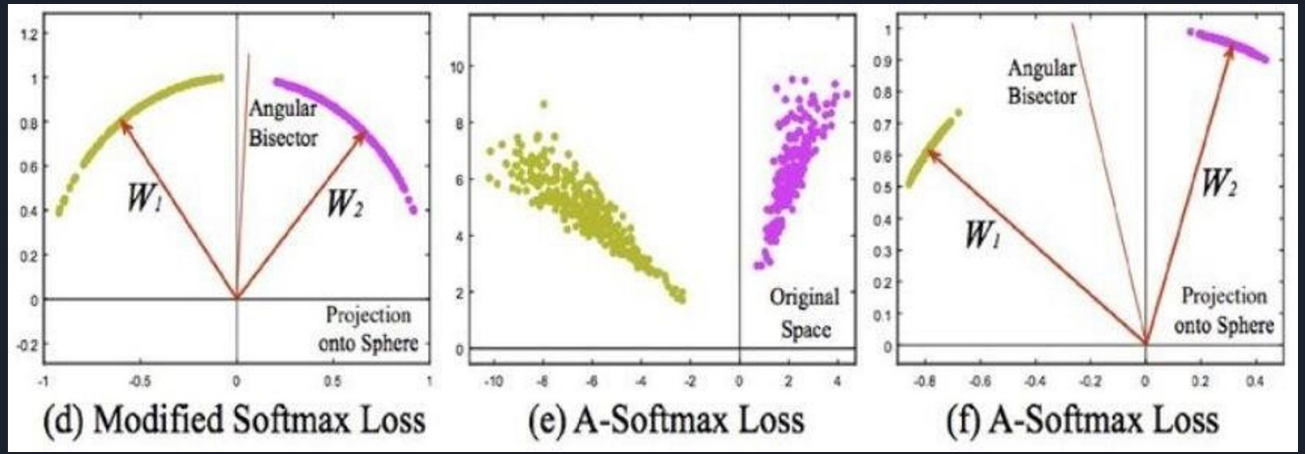


Figure 5 — What we would like the algorithm to do. Notice how the white dots that were outside weren't moved farther away from the margin.

Other types of Losses

- Pair-wise Loss (separate distributions of similarity scores)
- Angular Softmax Loss



References

- https://arxiv.org/pdf/1512.03385.pdf
- https://arxiv.org/pdf/1608.06993v3.pdf
- https://arxiv.org/pdf/1409.1556.pdf
- https://arxiv.org/pdf/1704.08063.pdf
- https://arxiv.org/pdf/1503.03832v3.pdf
- http://ydwen.github.io/papers/WenECCV16.pdf
- https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf
- https://towardsdatascience.com/densenet-2810936aeebb
- http://yann.lecun.com/exdb/publis/pdf/hadsell-chopra-lecun-06.pdf
- https://papers.nips.cc/paper/4824-imagenet-classification-with-deepconvolutional-neural-networks.pdf