CNNs: Classification and Verification

Recitation 6

Zhefan Xu, Advait Gadhikar

Today we will talk about

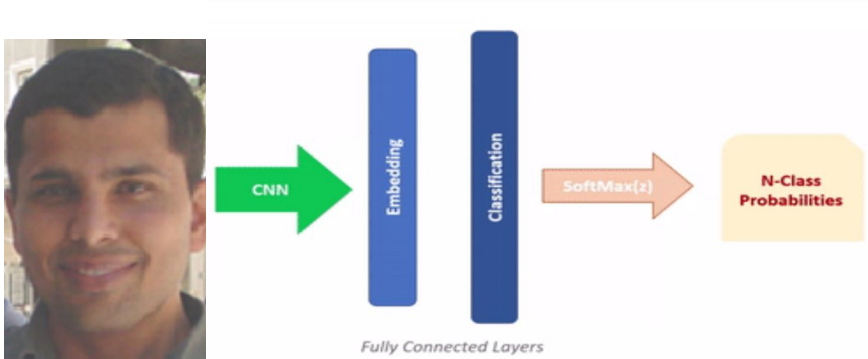
- Problem Statement for Part 2
  - Classification and Verification
  - Transfer Learning
- Model Architectures
- Types of Losses
- Dataset and custom Dataloader for PyTorch(in the Notebook)

Problem Statement

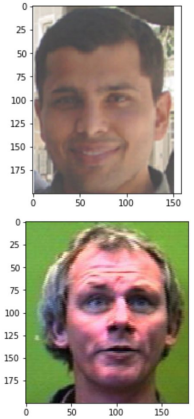
- Face Classification:
  - Classifying the person(ID) based on the image of the person’s face
- Face Verification:
  - How would you use the same network you trained for Classification to do face verification, Ideas??
  - You identify the most important features in the image which capture the identity of the face
  - The extracted features will be represented by a fixed length vector, known as an embedding
  - In order to do verification, we need to identify if a given embedding is similar to a reference embedding of that person using a distance metric like the Cosine Distance

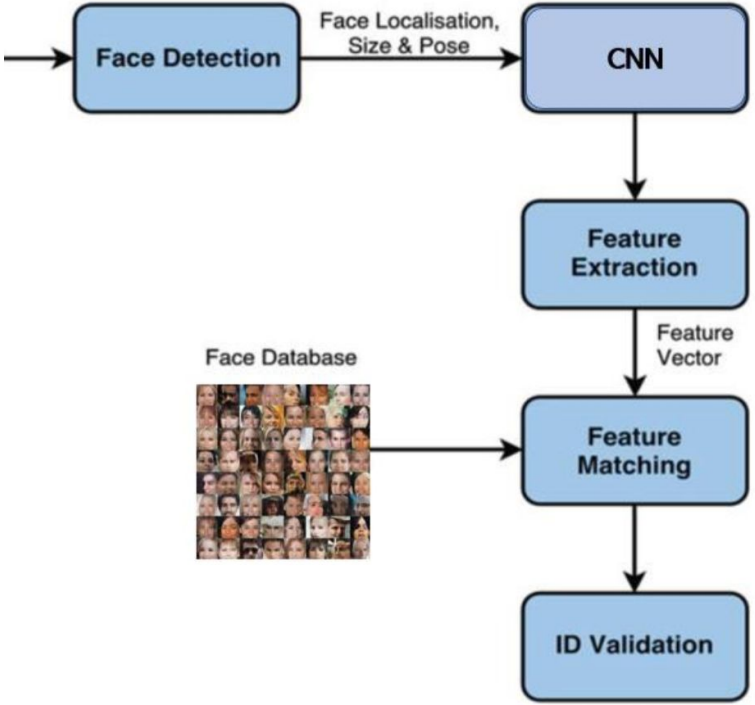
Classification vs Verification

- Classification
  - An N way classification task, predicting from a fixed set of possible output classes
- Verification
  - It is a matching operation, where you match the given sample to the closest sample from a reference of N other samples
  - Can also be a 1 to 1 task, where we want to verify if the two embeddings are similar (belong to the same class)
  - N way classification using cross entropy loss



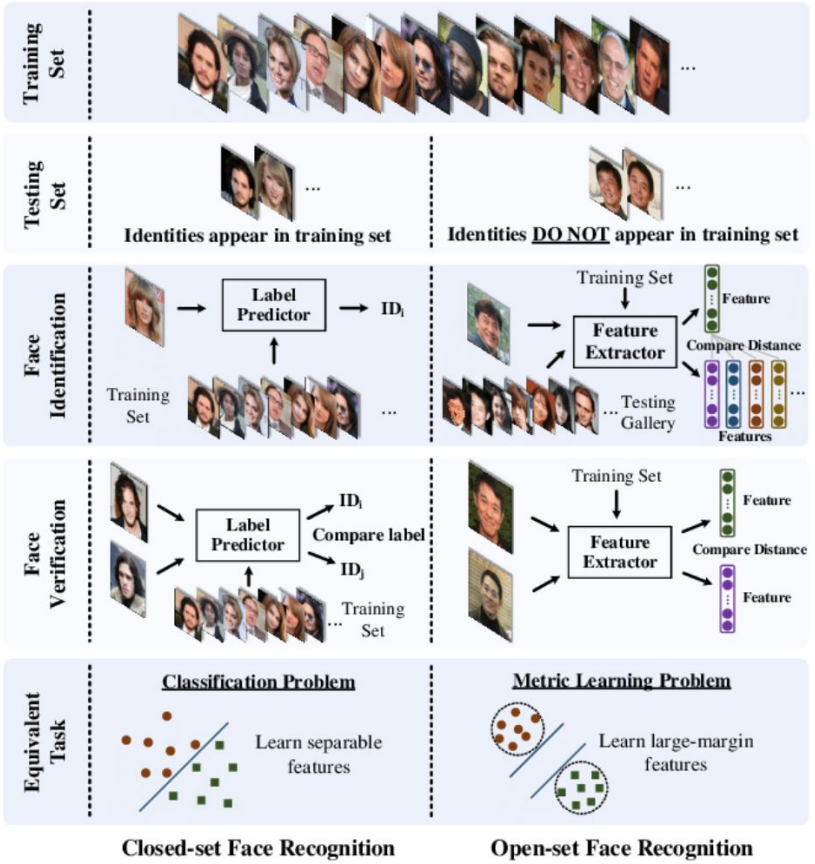
Verification





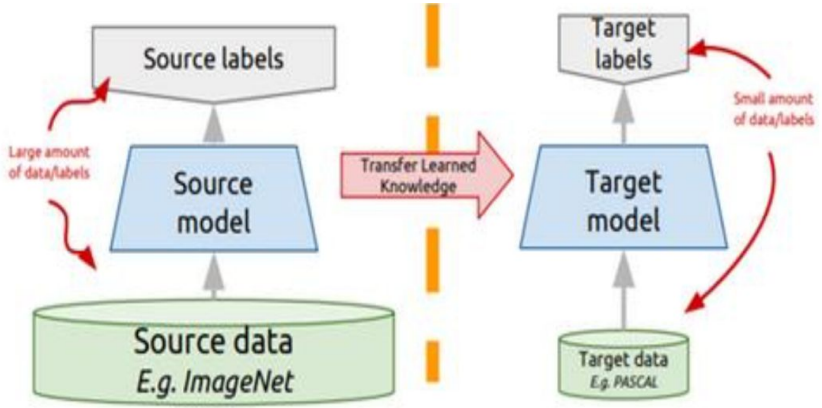
Open set vs Closed Set

Classification versus Verification & Open versus Closed set for the facial recognition problem.



Transfer Learning

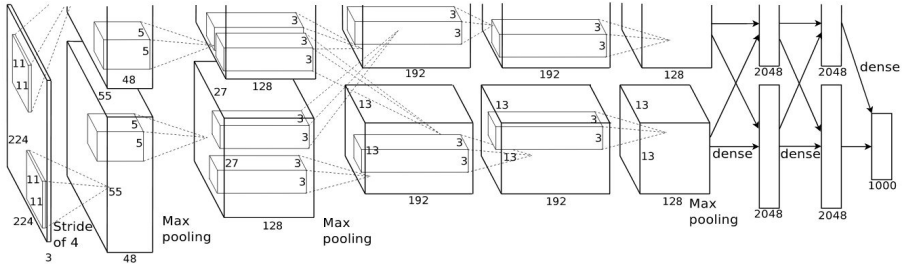
- Transfer Learning is a method where a model developed for a task is reused as the starting point for a model on another task
- Two ways to do Transfer Learning
  - Use trained model weights to initialize the network and train the entire network again
  - Freeze the weights of the network and fine tune the last few layers on the target dataset



CNN Architectures: AlexNet

Published in 2012 and was the winner of the Imagenet LSVRC-2012

- Uses ReLU as it makes training faster
- Implements dropout between layers
- Uses data augmentation methods (Random Crop, Random Flip in PyTorch)
- The network is trained using SGD with momentum

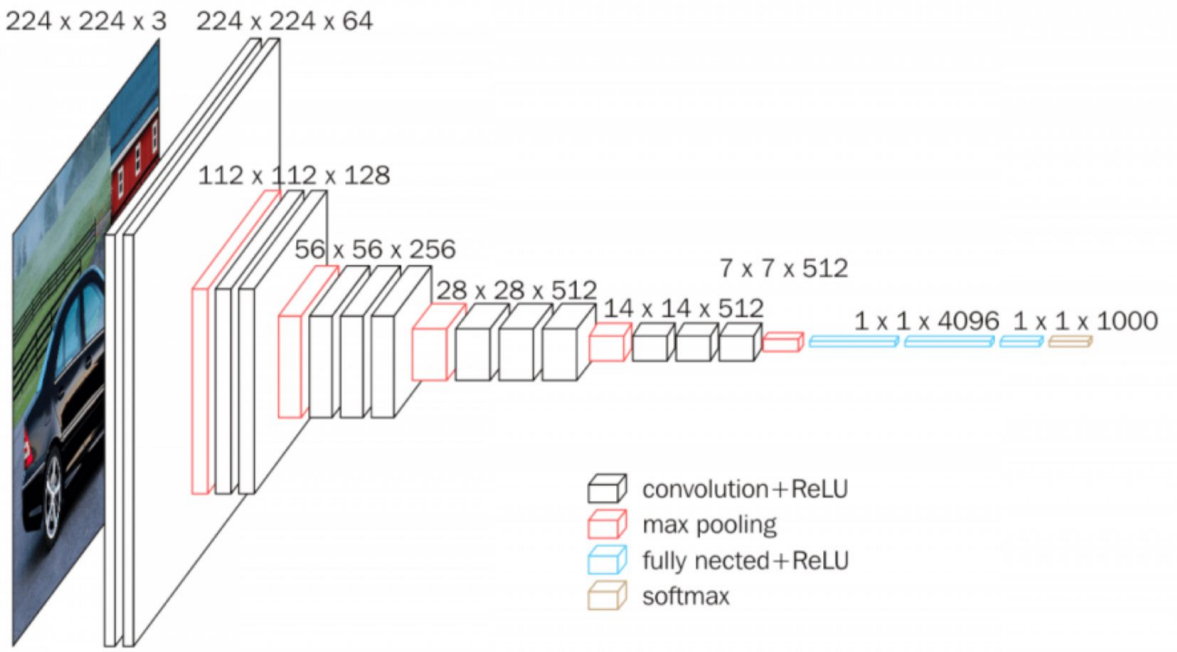


https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

VGG Net

VGG (2014) architecture reduces the size of each layer yet increases the overall depth of it. reinforces the idea that CNNs must be deep in order to work well on visual data.

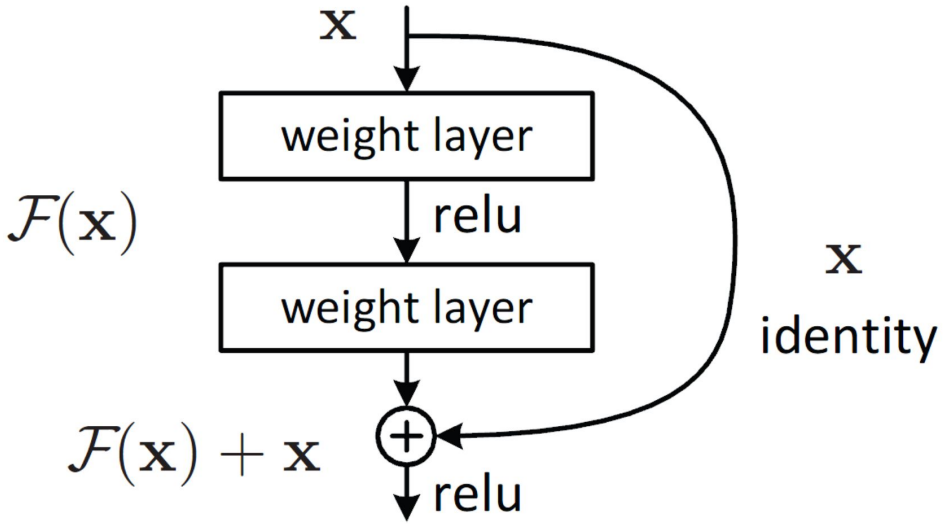
- Conv filters (3*3)
- Max Pool (2*2)



https://arxiv.org/pdf/1409.1556.pdf

ResNet

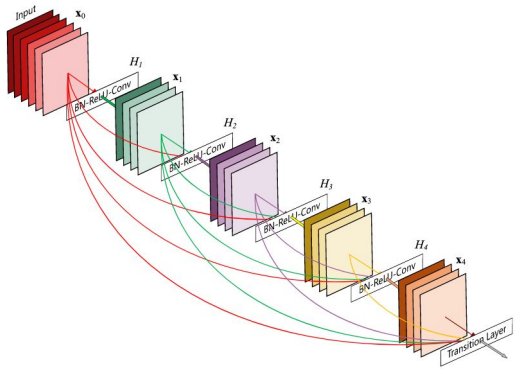
- Introduced in 2015, utilizes bottleneck architectures efficiently and learns them as residual functions
- Easier to optimize and can gain accuracy from increased depth due to skip connections

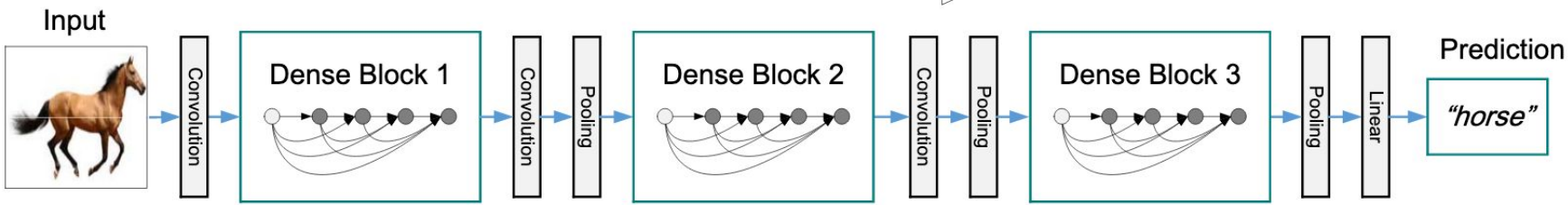


https://arxiv.org/pdf/1512.03385.pdf

Dense Nets

- It includes a stack of dense blocks followed by a transition layers
- Strengthen feature propagation, encourage feature reuse and decrease the number of parameters





https://arxiv.org/pdf/1608.06993.pdf

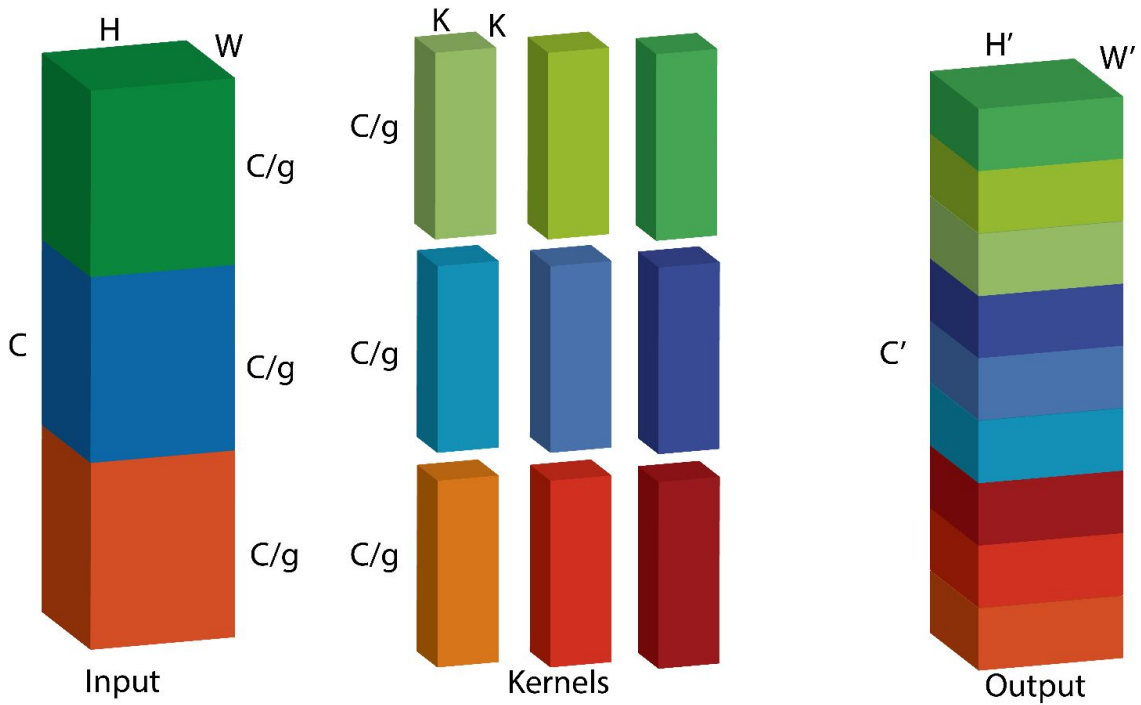
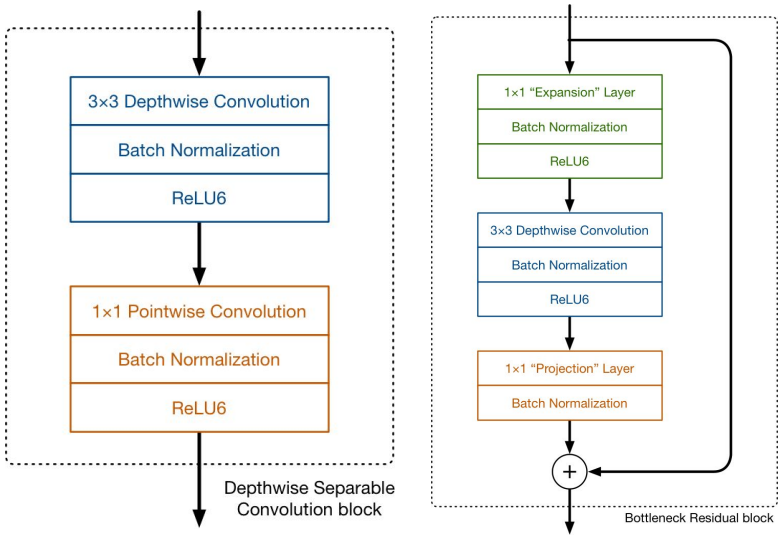


Figure 0: Grouped Convolutions with N_i = C input Channels, g groups, and N_o = C' output Channels

MobileNet

- Introduced in 2017, intended to run neural networks efficiently on mobile devices

  - V1 introduces Depth wise separable convolution
  - V2 introduces Inverted Residual block



https://arxiv.org/pdf/1704.04861.pdf

Residual and Inverted Residual block

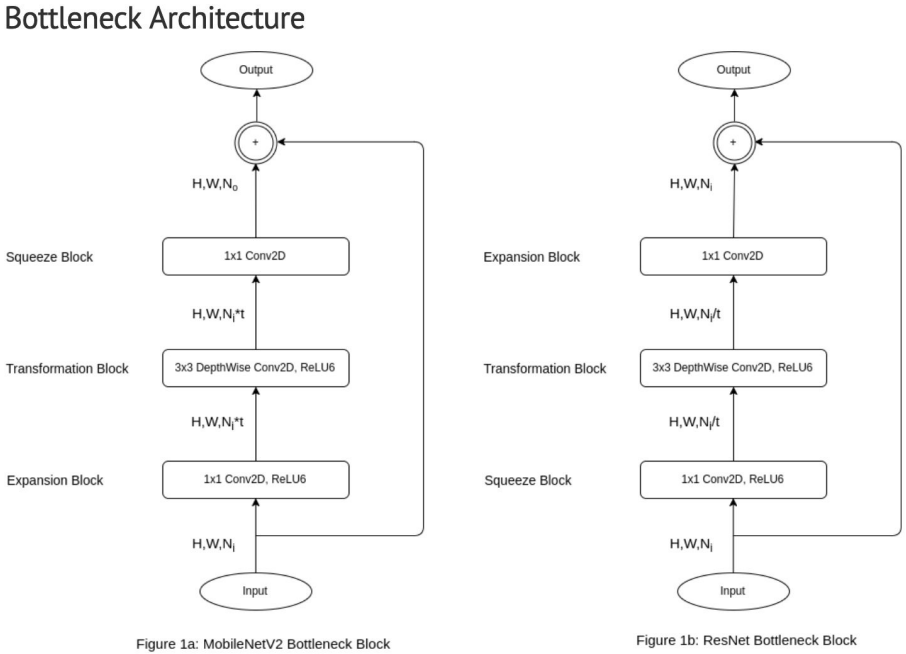
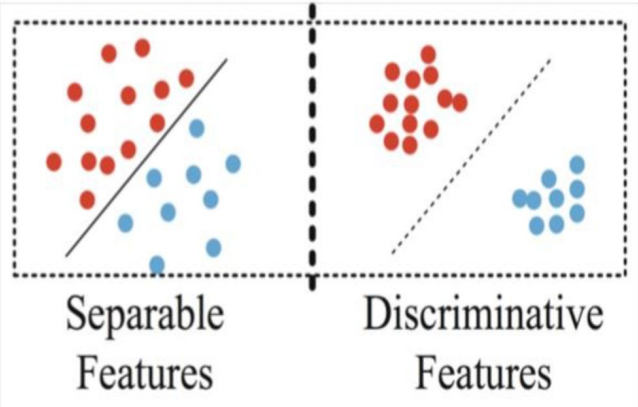


Figure 1: MobileNetV2 Architecture, BN is used after every conv and is omitted for brevity

Discriminative Features

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

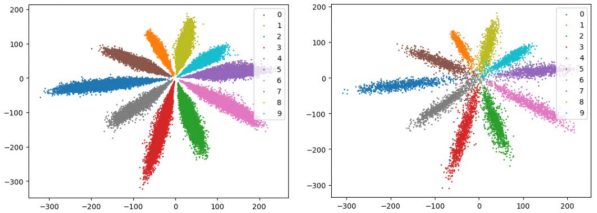


Center Loss

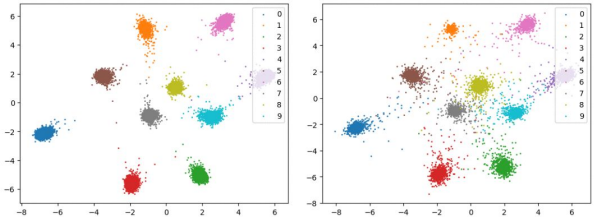
- Tries to minimize the intra class distance by adding a euclidean distance loss term

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

Softmax only. Left: training set. Right: test set.

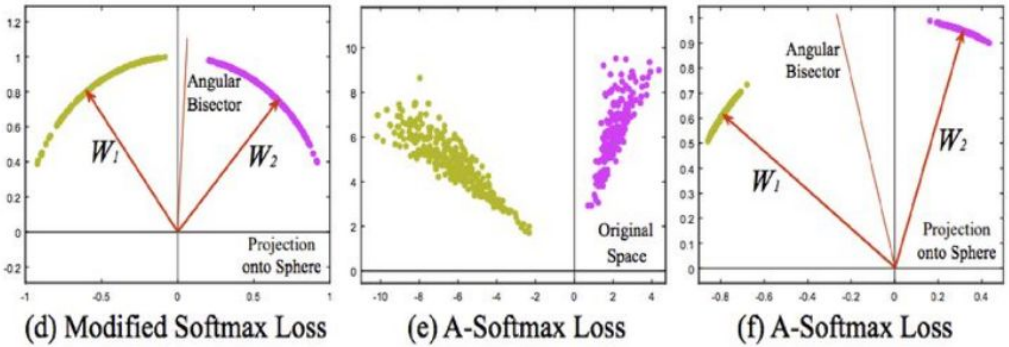


Softmax + center loss. Left: training set. Right: test set.



Other types of Losses

- Contrastive Loss (maintain margin between classes)
- Triplet Loss (motivated from nearest neighbour classification)
- Pair Wise Loss (separate distributions of similarity scores)
- Angular Softmax Loss



(d) Modified Softmax Loss (e) A-Softmax Loss (f) A-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-deep-convolutional-neural-networks.pdf