CNNs: Classification and Verification

Recitation 6

Jinhyung Park

Today we will talk about

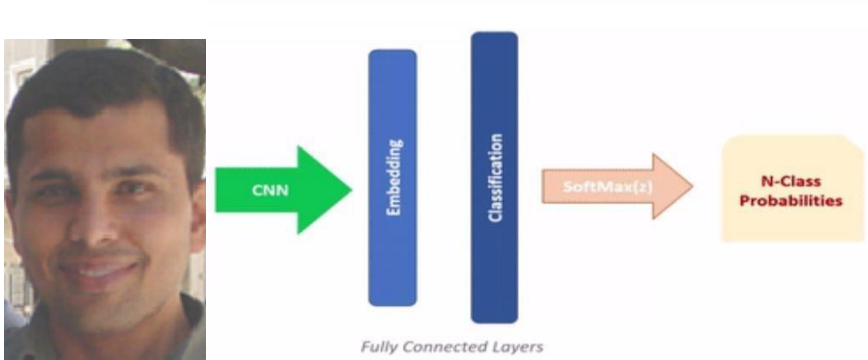
- Problem Statement for Part 2
  - Classification and Verification
  - Transfer Learning
- Model Architectures
- Types of Losses
- Run through the starter notebook

Problem Statement

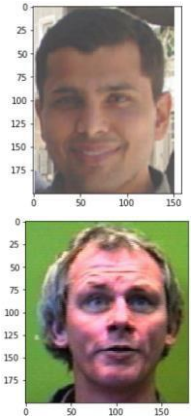
- Face Classification:
  - Classifying the person(ID) based on the image of the person’s face
- Face Verification:
  - How would you use the same network you trained for Classification to do face verification, Ideas??
  - You identify the most important features in the image which capture the identity 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

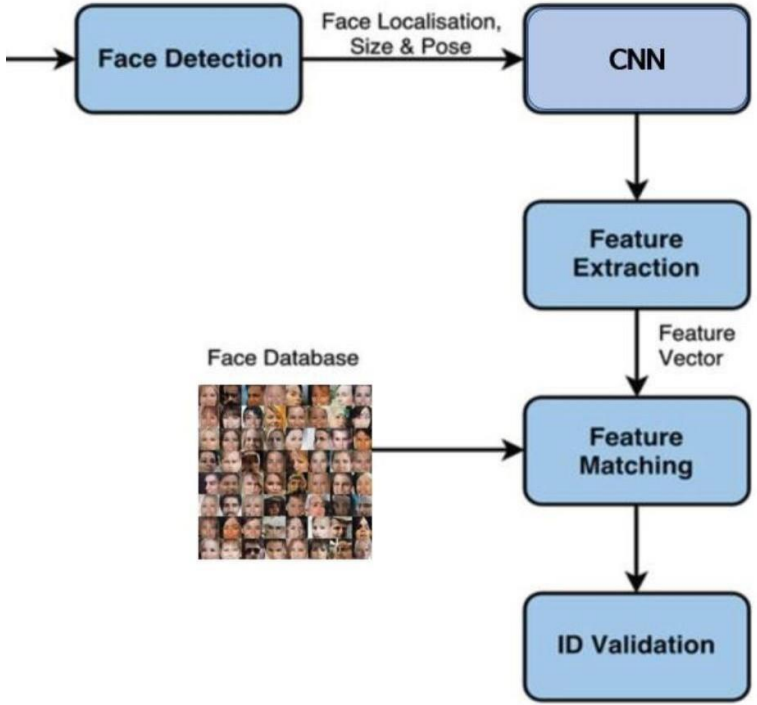
Classification vs Verification

- 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



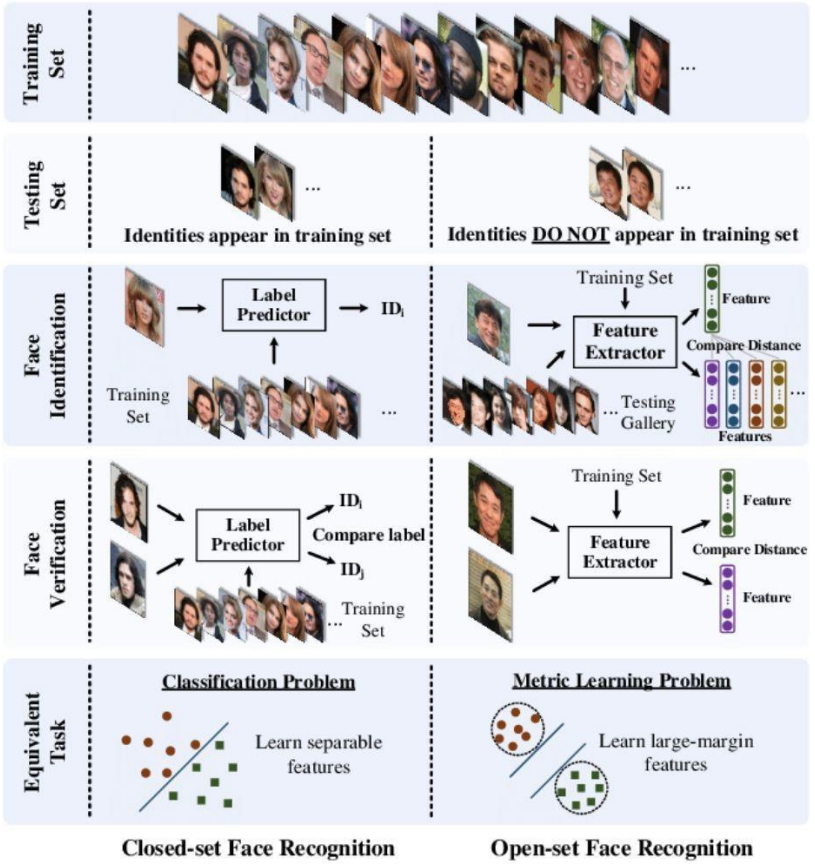
Verification



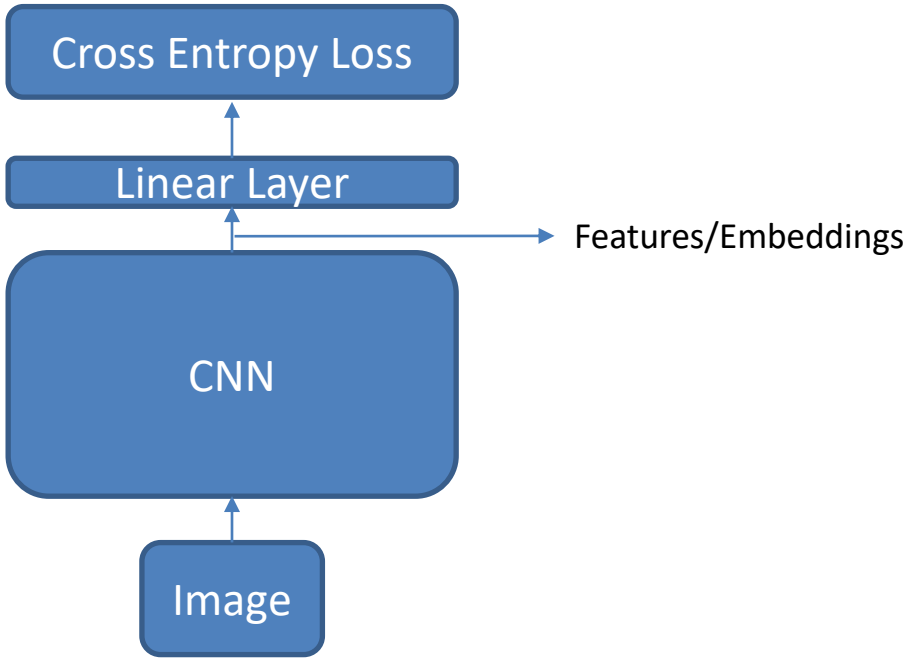


Open set vs Closed Set

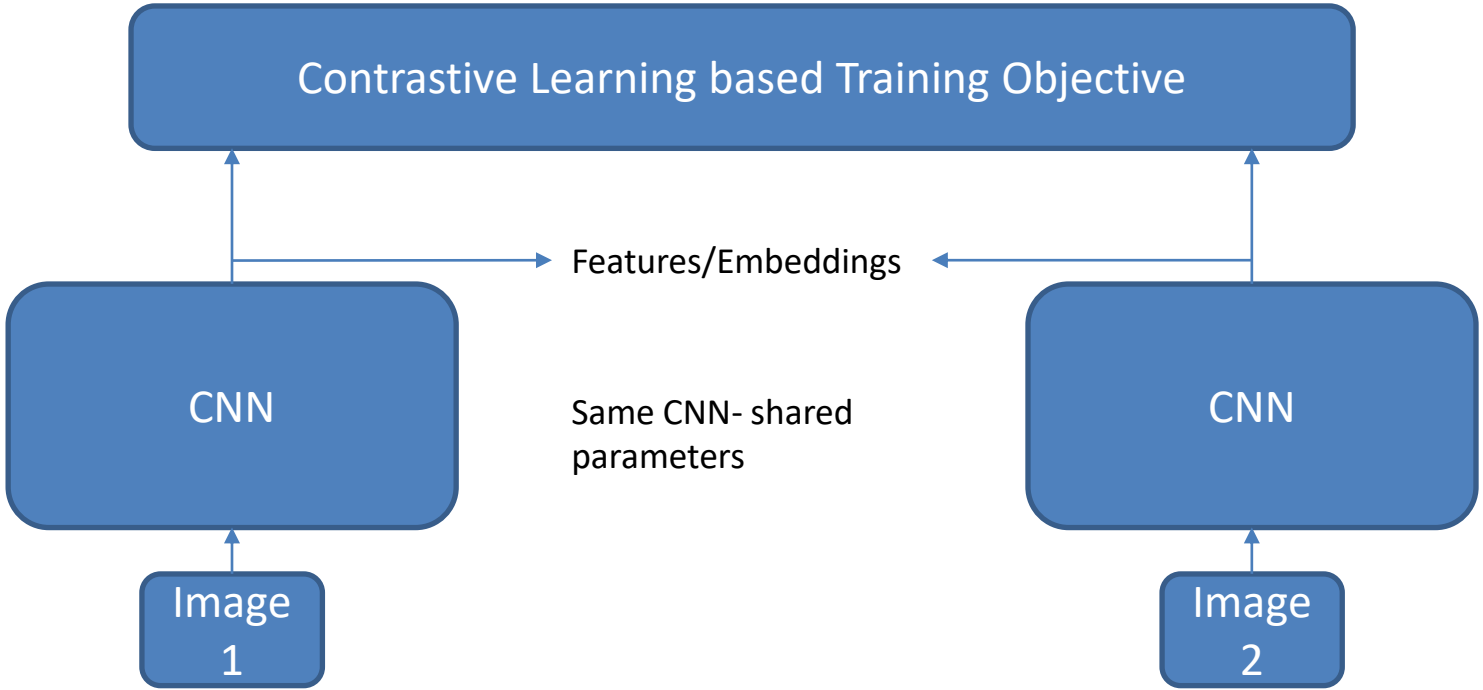
Classification versus Verification & Open versus Closed set for the facial recognition problem.



Training for Classification task

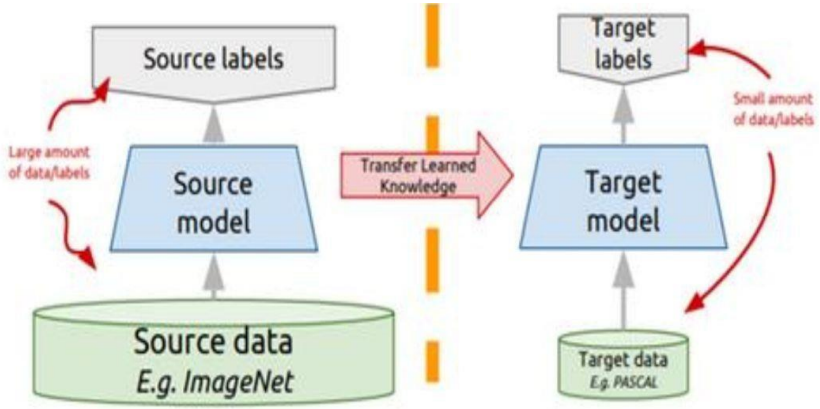


Training for Verification task



Transfer Learning

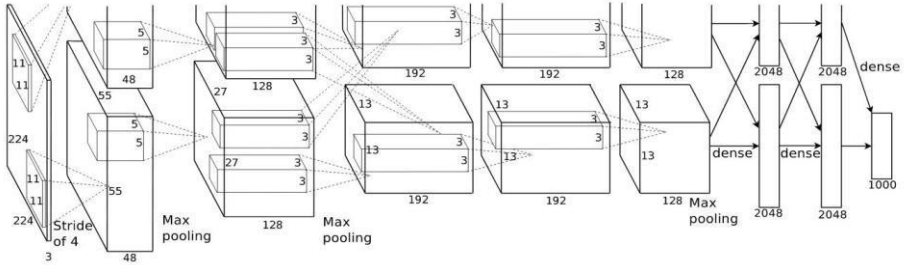
- Transfer Learning is a method where a model developed for a task is reused as the starting point for a model on another task
- Two ways to do Transfer Learning
  - Use trained model weights to initialize the network and train the entire network again
  - Freeze the weights of the network and fine tune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



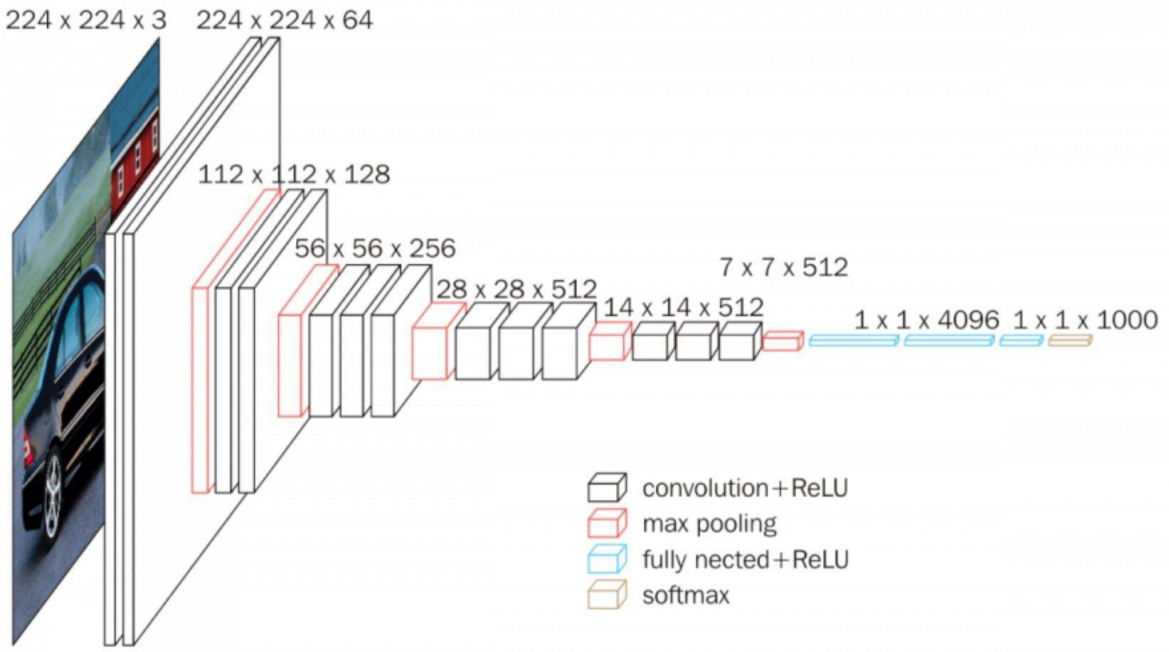
https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

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.

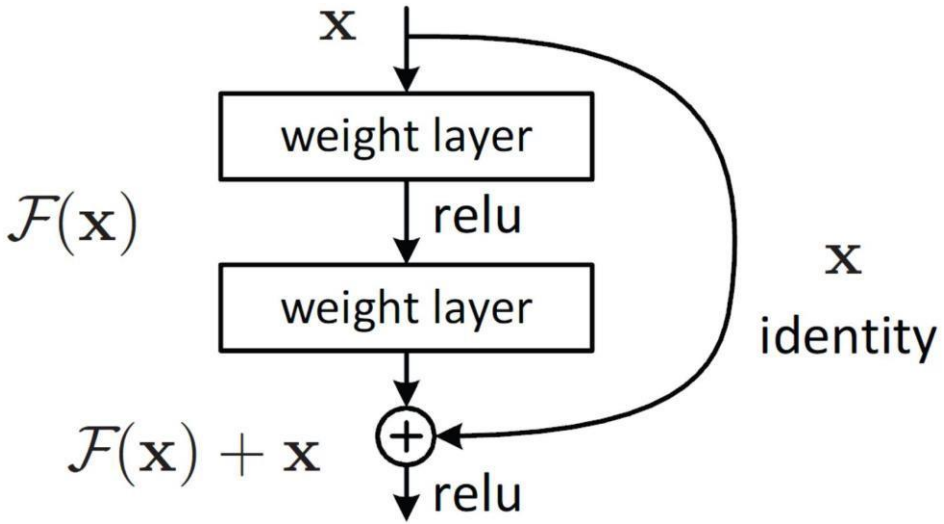
- Conv filters (3*3)
- Max Pool (2*2)

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



ResNet

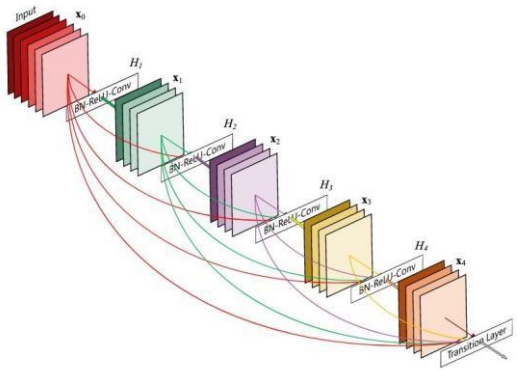
- Introduced in 2015, utilizes bottleneck architectures efficiently and learns them as residual functions
- Easier to optimize and can gain accuracy from increased depth due to skip connections

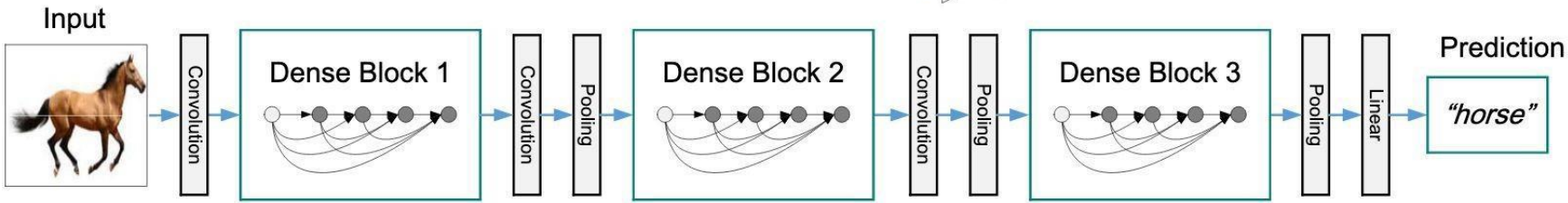


https://arxiv.org/pdf/1512.03385.pdf

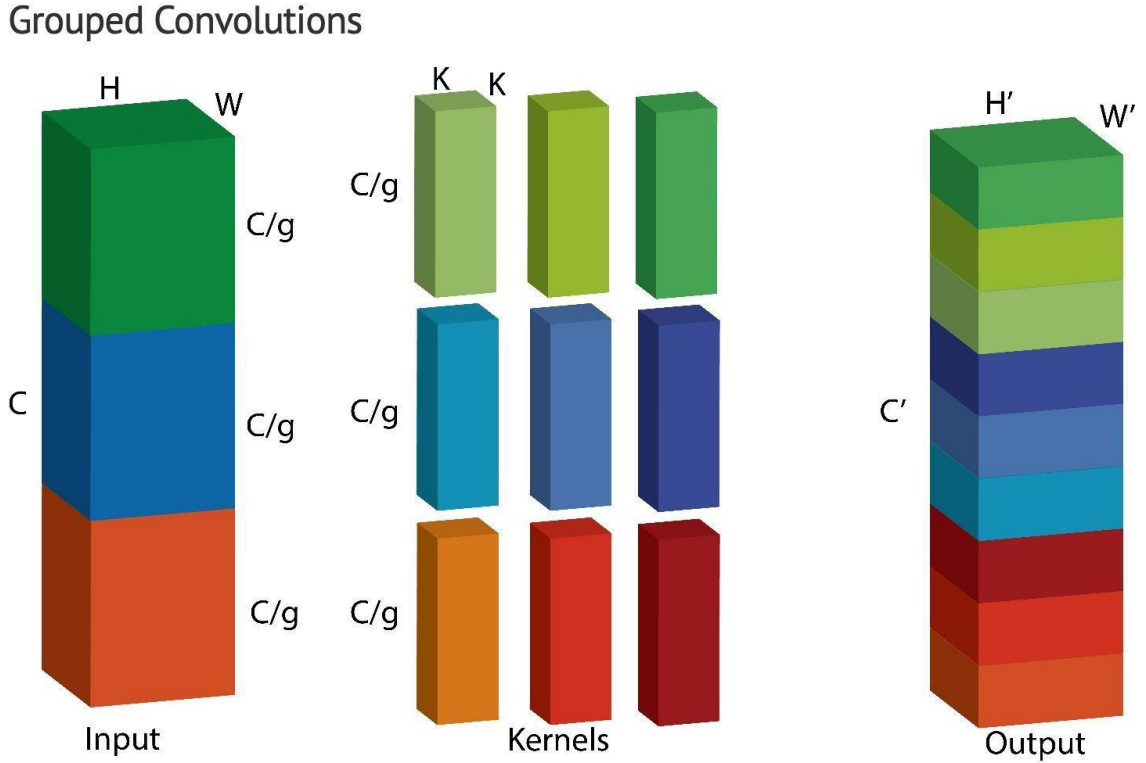
Dense Nets

- 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



Depthwise Separable Convolution is a special case of Grouped Convolutions where g = 1

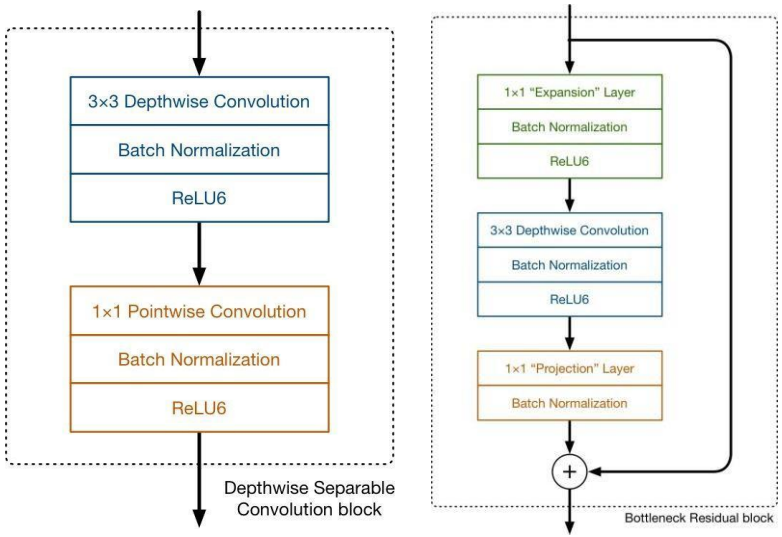
ShuffleNet is an extension to the Group Convolution which shuffles the output channels C’

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

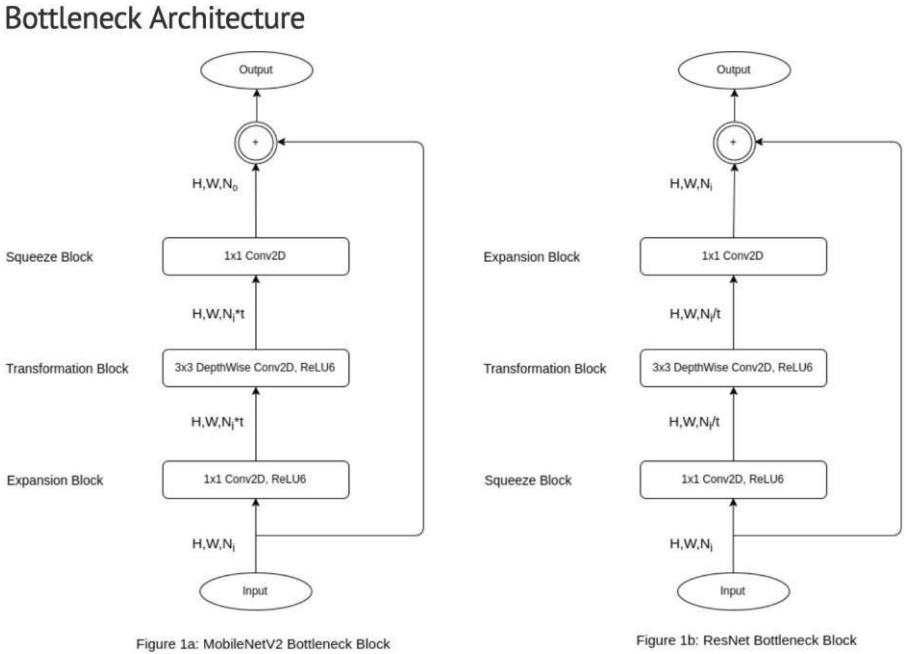
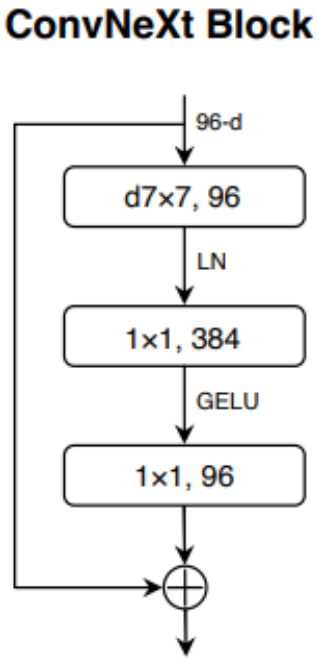


Figure 1a: MobileNetV2 Bottleneck Block

Figure 1b: ResNet Bottleneck Block

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

ConvNext



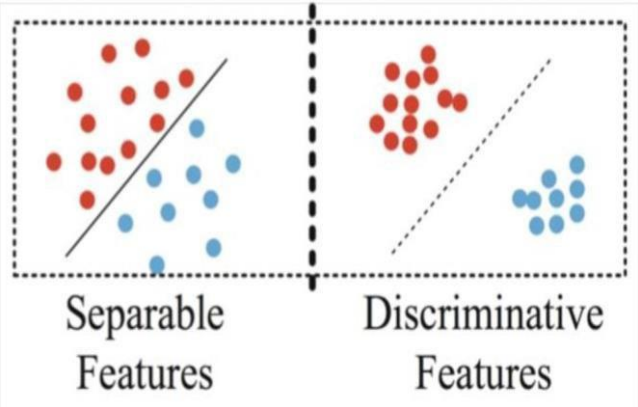
- A 7x7 depth-wise convolution.
- A point-wise convolution increasing # of channels
- A point-wise convolution decreasing # of channels
- Residual Connection

Other Interesting Papers

- ResNeXt (2016)
  - https://arxiv.org/pdf/1611.05431.pdf
  - Generally a strict improvement to ResNet, but slower. It's like 3 lines of code changed.
- SENet (2017)
  - https://arxiv.org/pdf/1709.01507.pdf
  - Channel-wise attention in CNNs. It's like 20 lines of code.
- EfficientNet (2019)
  - https://arxiv.org/pdf/1905.11946.pdf
  - Optimized model scaling. Probably can hard code this with some effort.
- RegNet (2020)
  - https://arxiv.org/pdf/2003.13678.pdf
  - ResNet with optimized layer sizes. It's probably… 10 lines changed?
- ResNeSt (2020)
  - https://arxiv.org/pdf/2004.08955.pdf
  - ResNeXt on steroids + attention. I (we?) will be really impressed ☺
- NFNet (2021)
  - https://arxiv.org/pdf/2102.06171v1.pdf
  - Quite doable actually

Discriminative Features

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



Center Loss

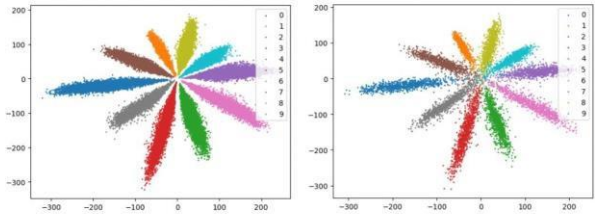
- Tries to minimize the intra class distance by adding a euclidean distance loss term
- If you use this, **YOU MUST USE CENTER LOSS FROM THE BEGINNING OF TRAINING CLASSIFICATION!**

$$\mathcal{L} = \mathcal{L}_S + \lambda \mathcal{L}_C$$

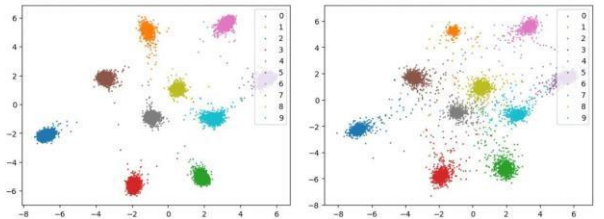
$$= -\sum_{i=1}^{m} \log \frac{e^{W_{y_i}^T \boldsymbol{x}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \boldsymbol{x}_i + b_j}} + \frac{\lambda}{2} \sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2$$

https://github.com/KaiyangZhou/pytorch-center-loss

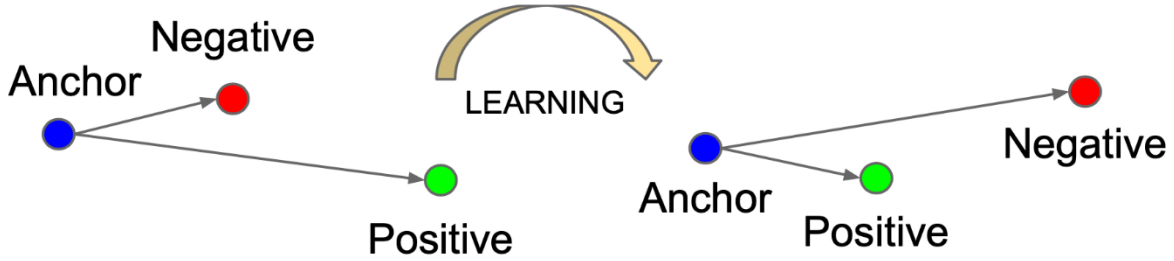
Softmax only. Left: training set. Right: test set.



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



Triplet Loss

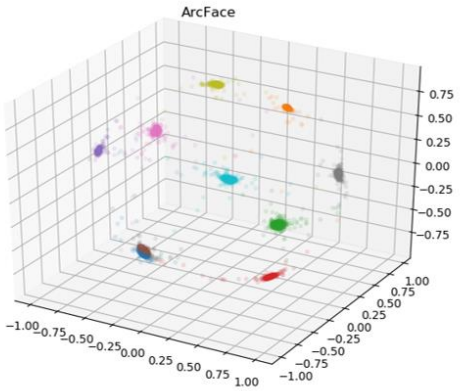


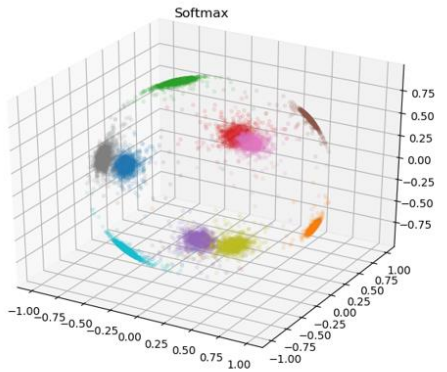
$$\mathcal{L}_{\text{triplet}}(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) = \sum_{\mathbf{x} \in \mathcal{X}} \max\left(0, \|f(\mathbf{x}) - f(\mathbf{x}^+)\|_2^2 - \|f(\mathbf{x}) - f(\mathbf{x}^-)\|_2^2 + \epsilon\right)$$

Here, x is the anchor image, x+ is the positive sample and x- is the negative sample. It is crucial to select challenging/hard negative samples to truly improve the model, known as **Hard Negative Mining**.

ArcFace

ArcFace calculates the geodesic distance on a hypersphere as opposed to Euclidean distance and tries to maximize the angular margin between two classes.



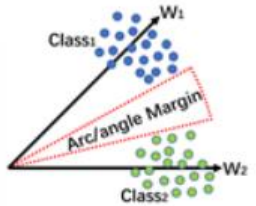


$$-\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s*(\cos(\theta_{y_i}+m))}}{e^{s*(\cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{n} e^{s*cos\theta_j}}$$

where θ_j is the angle between the weight W_j and the feature x_i

s - feature scale, the hypersphere radius

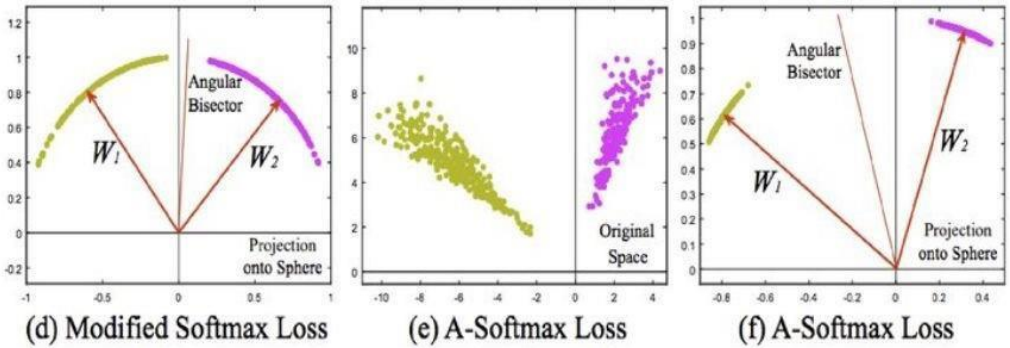
m - angular margin penalty



https://arxiv.org/pdf/1801.07698.pdf

Other types of Losses

- Contrastive Loss (maintain margin between classes)
- Pair Wise Loss (separate distributions of similarity scores)
- Angular Softmax Loss



(d) Modified Softmax Loss (e) A-Softmax Loss (f) A-Softmax Loss

https://github.com/adambielski/siamese-triplet/blob/master/losses.py

Regularization Techniques

- Label Smoothing
- DropBlock

Label Smoothing

- Deep Learning Models undergo a problem of Overfitting and Overconfidence.
- Label Smoothing is a technique that can help us solve the problem of Overconfidence.

What is Overconfidence?
For each sample, the model predicts outcomes with higher probabilities than the accuracy over the entire dataset.
This is a poorly calibrated model.
For example, it may predict 0.9 for inputs where the accuracy is only 0.6.

Example

Without Label Smoothing

Suppose we have $K = 3$ classes, and our label belongs to the 1st class.

Logit Vector z = [a, b, c]

Label vector y = [1, 0, 0] (one-hot encoded)

Gradient of Loss = softmax(z) – y

Our model will make $a \gg b$ and $a \gg c$

z = [10, 0, 0]

softmax(z) = [0.9999, 0, 0]

Label Smoothing

$y\_ls = (1 - \alpha) * y\_hot + (\alpha / K)$

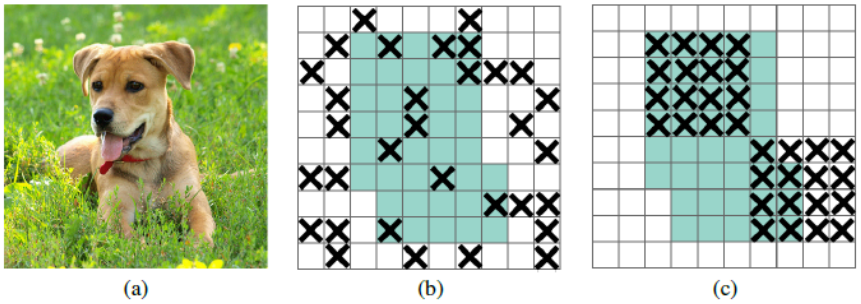
Example: With Label Smoothing

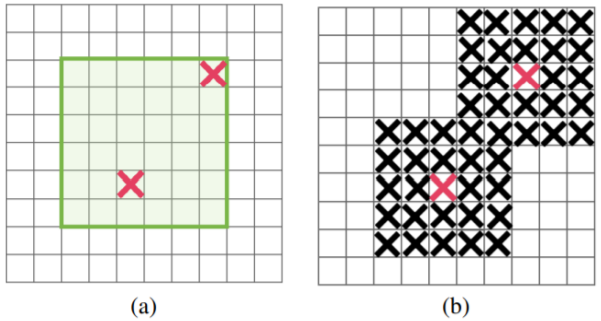
$\alpha = 0.1$
y_ls = [0.9333, 0.0333, 0.0333]
This would result into the logits z = [3.3332, 0, 0]
softmax(z) = [0.9333, 0.0333, 0.0333]

https://arxiv.org/pdf/1812.01187.pdf
https://arxiv.org/abs/1812.01187
https://github.com/ankandrew/online-label-smoothing-pt

DropBlock

- Dropout usually works better with Fully Connected Networks
- Dropout has not proven to be useful in CNNs because of the spatial correlation between the activation outputs
- DropBlock is a regularization technique that has proven to be useful for CNNs
- It is a structured form of dropout that drops contiguous regions and not just random pixels





https://arxiv.org/pdf/1810.12890.pdf

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://towardsdatascience.com/what-is-label-smoothing-108debd7ef06
- https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf