HW2P2 Bootcamp

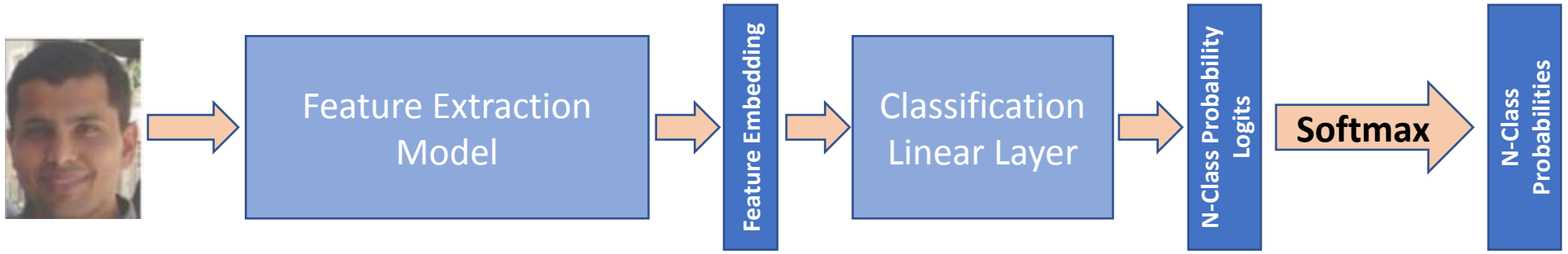
Logistics

- Early Submission is due **in 4 days** October 10th, 11:59 PM EST
  - Make sure to do the early Kaggle submission & the Canvas MCQ.
  - You need at least a 10% in classification, and 7.5% accuracy in verification
- The on-time submission deadline is October 26th, 11:59 PM EST.
- HW2P2 is **significantly harder** than HW1P2. Models will be harder to develop, train, and converge. Please start early!
- Models must be written yourself and trained from scratch.

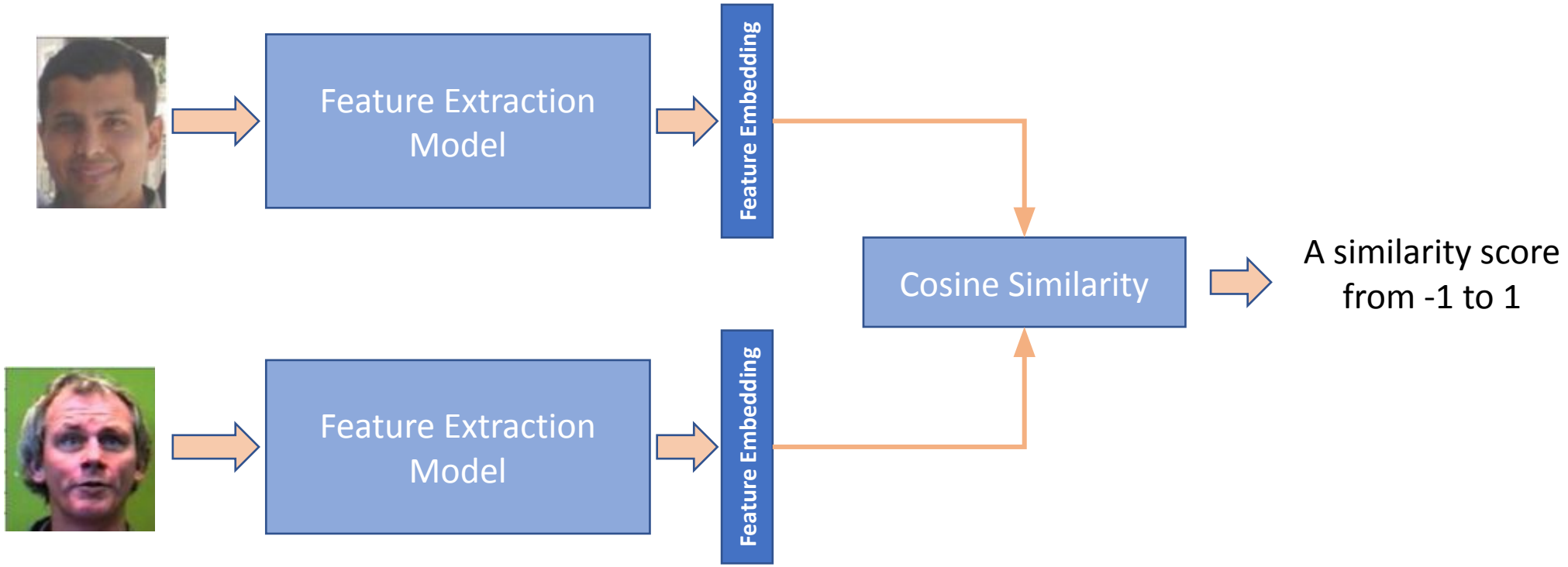
Problem Statement

- **Face Classification**
  - Given an image, figure out which person it is.
- **Face Verification**
  - Given a set of images, figure out if they are the same person or not.

Face Classification



Face Verification



Face Verification

Unknown image

Known images

Feature Extraction Model

Feature Extraction Model

Feature Embedding

Feature Embedding

Cosine Similarity for each known image

Take argmax

A similarity score from -1 to 1

CLASS `torch.nn.CosineSimilarity(dim=1, eps=1e-08)` [SOURCE]

Returns cosine similarity between x_1 and x_2, computed along *dim*.

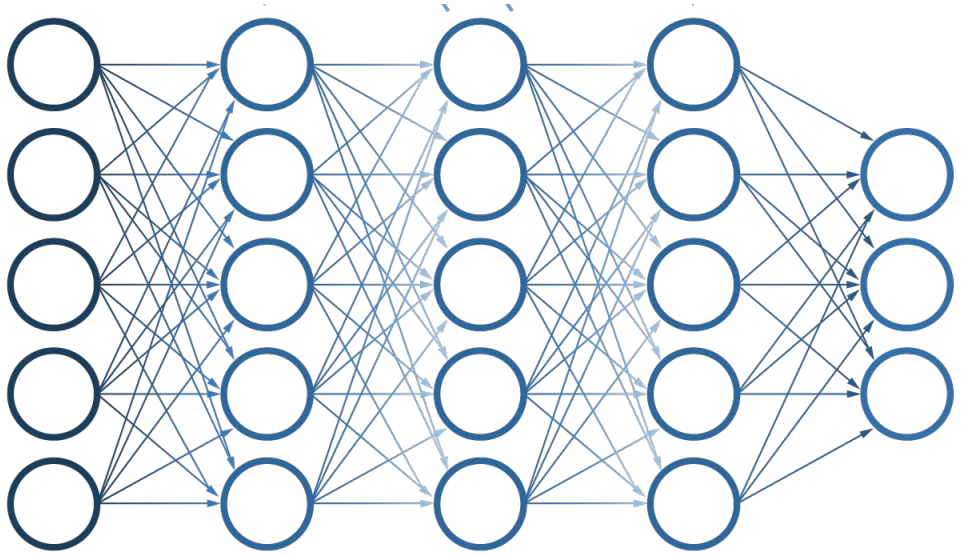
$$\text{similarity} = \frac{x_1 \cdot x_2}{\max\left(\|x_1\|_2 \cdot \|x_2\|_2, \epsilon\right)}.$$

Workflow

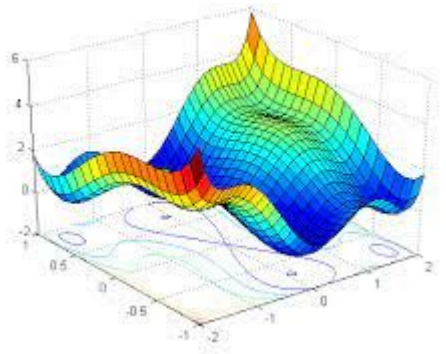
- First train a strong classification model for the classification task.
- Then, for the verification task, use the model trained on classification.
  - take the penultimate features as feature embeddings of each image.
- You should additionally train verification-specific losses such as ArcFace, Triplet Loss to improve performance.

Building Blocks

Input Image +
Transformations



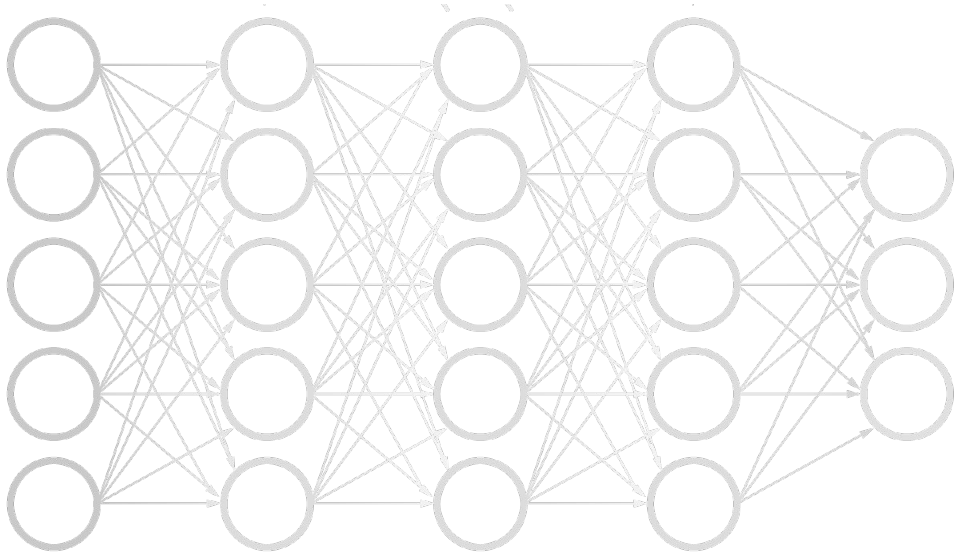
Choice of Model



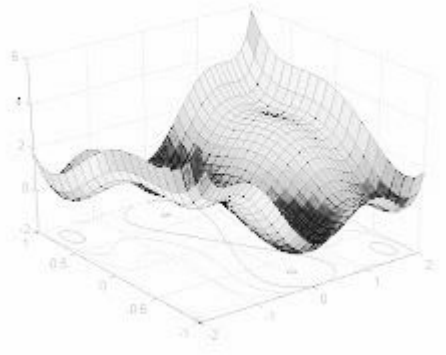
Training the model

Building Blocks

Input Image +
Transformations

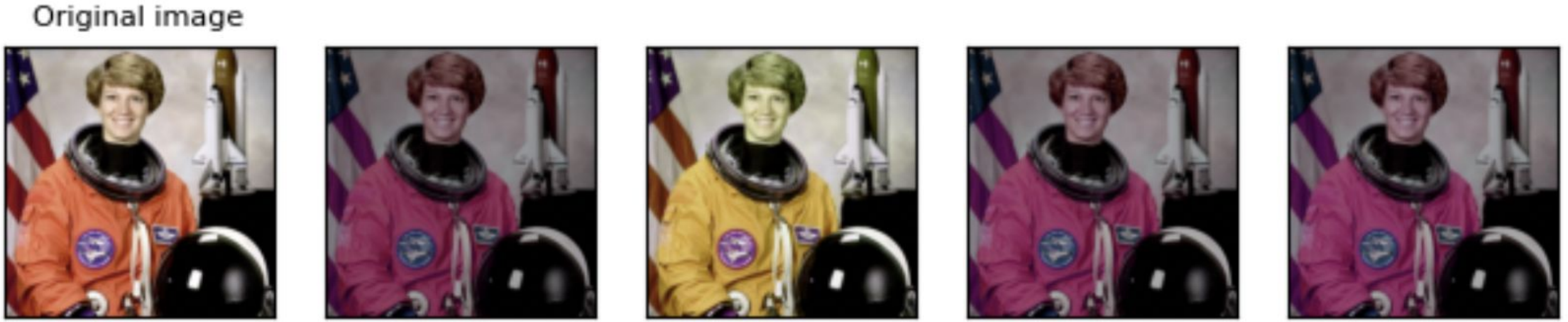


Choice of Model

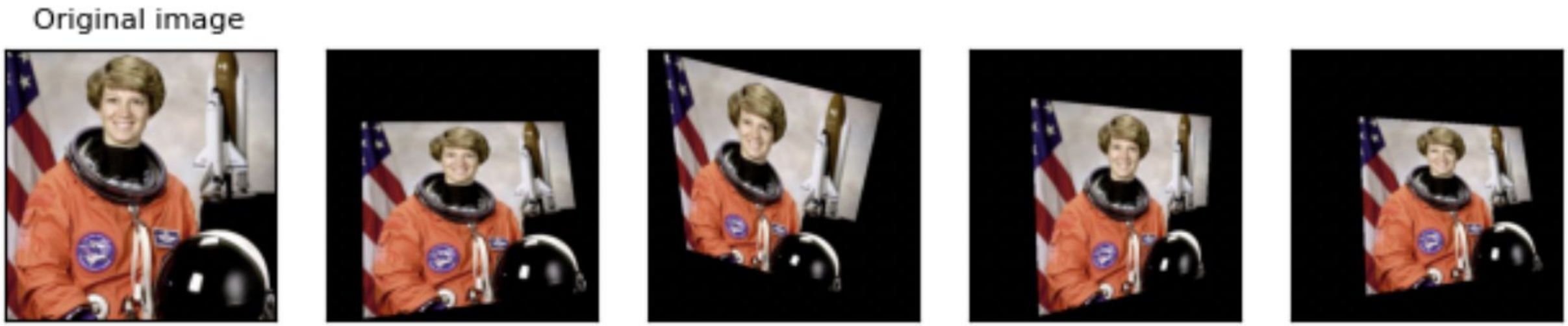


Training the model

Color Jitter



Random Perspective



Random Vertical Flip

Original image

Transformation Guide

URL:
https://pytorch.org/vision/stable/auto_examples/plot_transforms.html#sphx-glr-auto-examples-plot-transforms-py

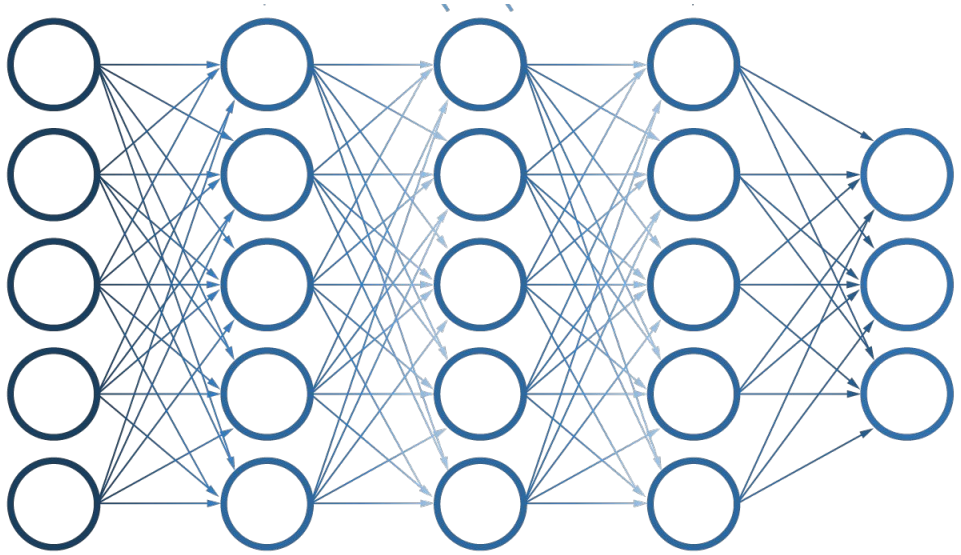
Common Issue:

```
TypeError: Input tensor should be a torch tensor. Got <class 'PIL.Image.Image'>.
```

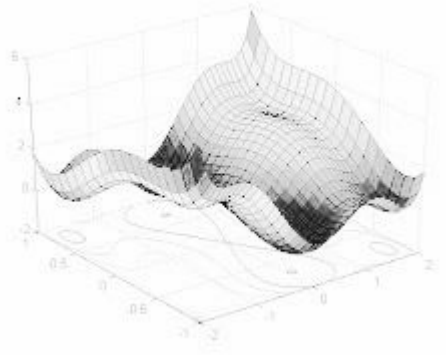
—> Please check the sequencing of your transforms. Read the documentation and verify the kind of input required.

Building Blocks

Input Image +
Transformations



Choice of Model



Training the model

Architectures

- At this point, you should have basic familiarity with convolutions as taught in lecture.
- Now, how do we take convolutions and assemble them into a strong architecture?
  - Layers? Channel size? Stride? Kernel Size? Etc.
- We'll cover three architectures:
  - MobileNetV2 – A fast, parameter-efficient model.
  - ResNet – The "go-to" for CNNs.
  - ConvNeXt – The state-of-the-art model.

General Architecture Flow

- CNN architectures are divided into stages, which are divided into blocks.
  - Each “stage” consists of (almost) equivalent “blocks”
  - Each “block” consists of a few CNN layers, BN, and ReLUs.
- To understand an architecture, we mostly need to understand its **blocks**.
- All that changes for blocks in different stages is the base # of channels

General Architecture Flow

- However, you do need to piece these blocks together into a final model.
- The general flow is like this:
  - Stem
  - Stage 1
  - Stage 2
  - …
  - Stage n
  - Classification Layer

General Architecture Flow

- The stem usually downsamples the input by 4x.
- Some stages do downsample. If they do, generally, the first convolution in the stage downsample by 2x.
- When you downsample by 2x, you usually increase channel dimension by 2x.
  - So, later stages have smaller spatial resolution, higher # of channels

MobileNetV2

- The goal of MobileNetV2 is to be parameter efficient.
- They do so by making extensive use of **depth-wise convolutions** and **point-wise convolutions**

A Normal Convolution

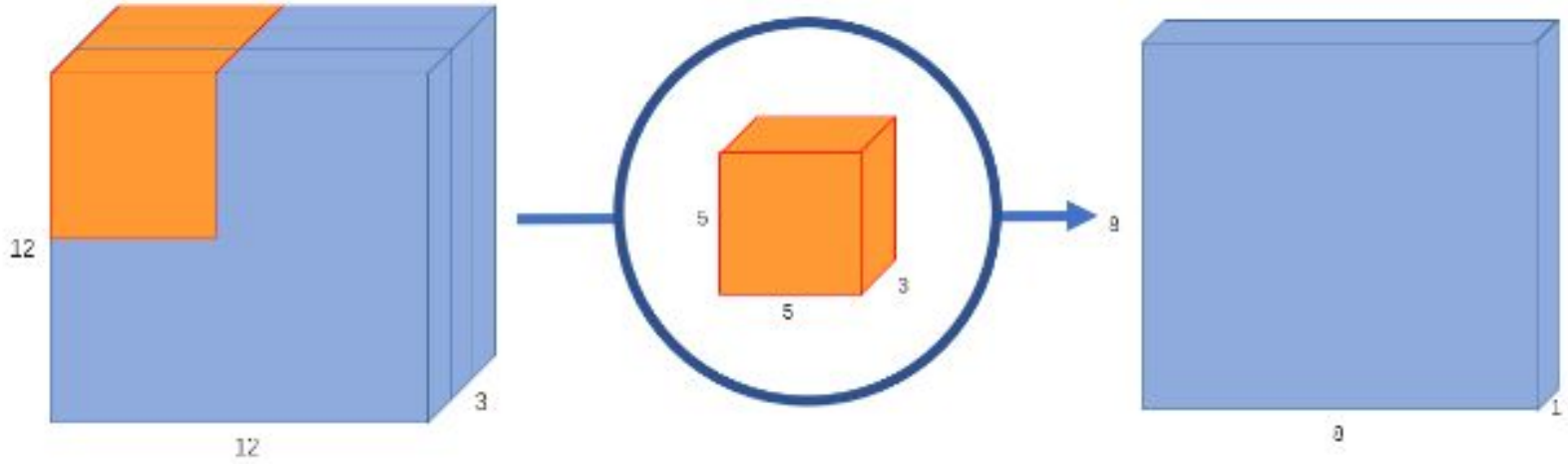


Image 4: A normal convolution with 8×8×1 output

- Considering just a single output channel

A Normal Convolution (Another Diagram)

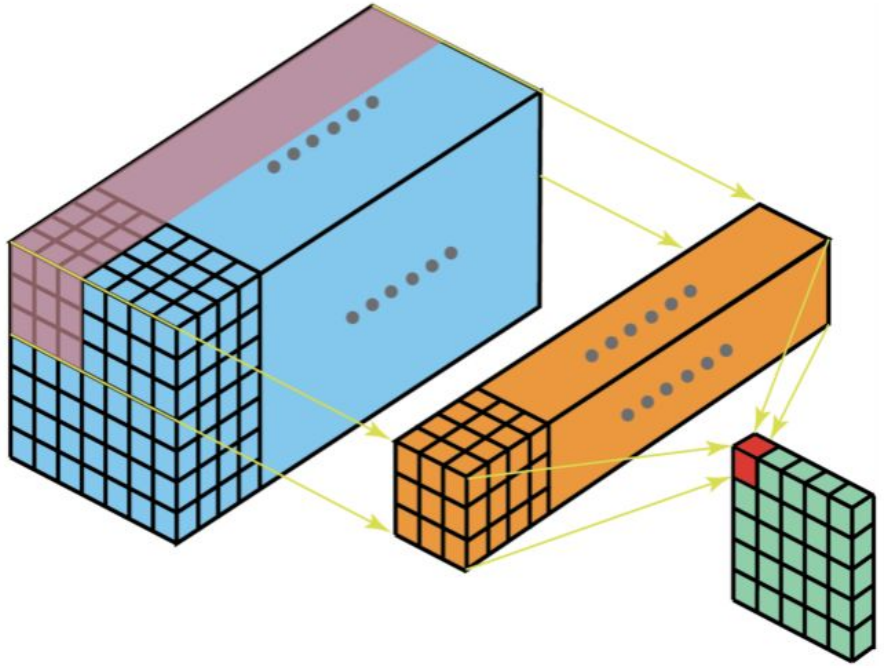


Fig. 2: Functional interpretation of 2D convolution (source)

- Considering a single output channel

A Normal Convolution

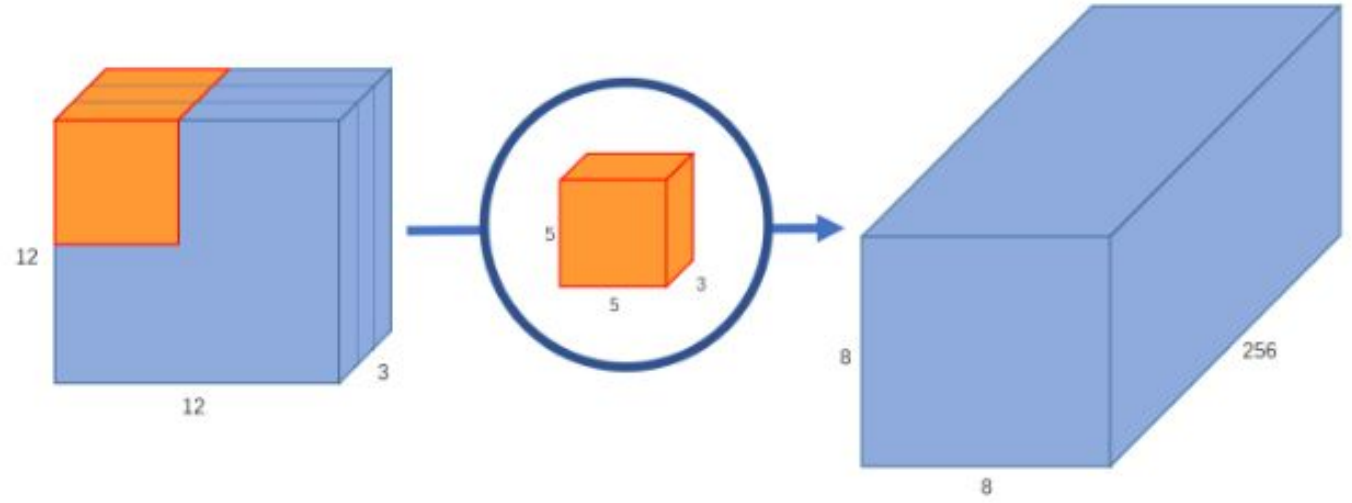
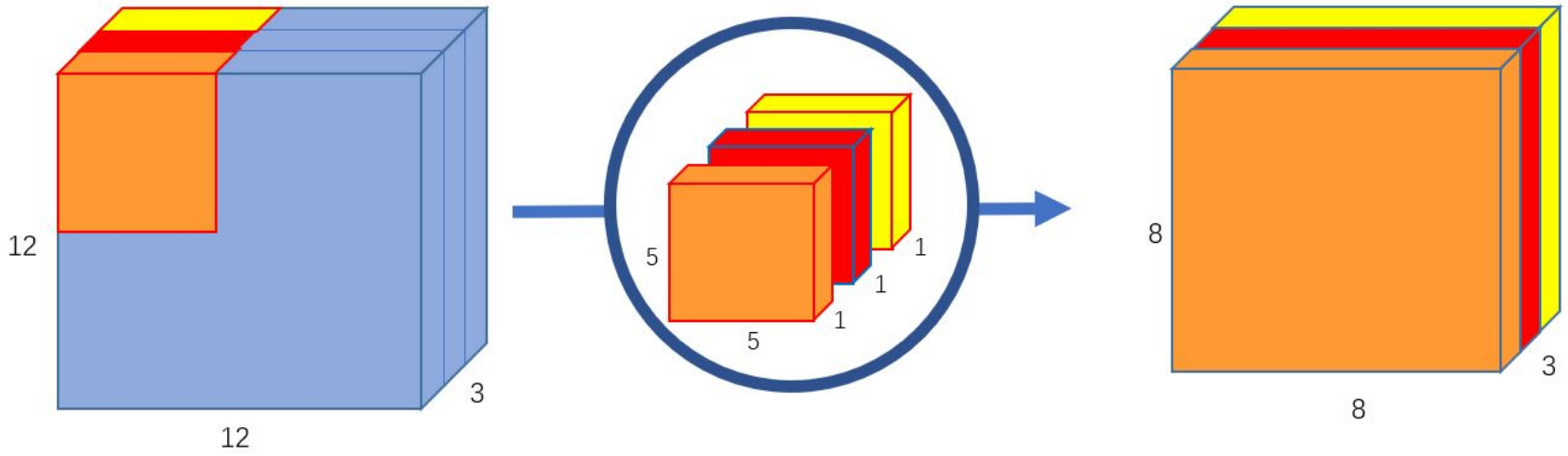


Image 5: A normal convolution with 8×8×256 output

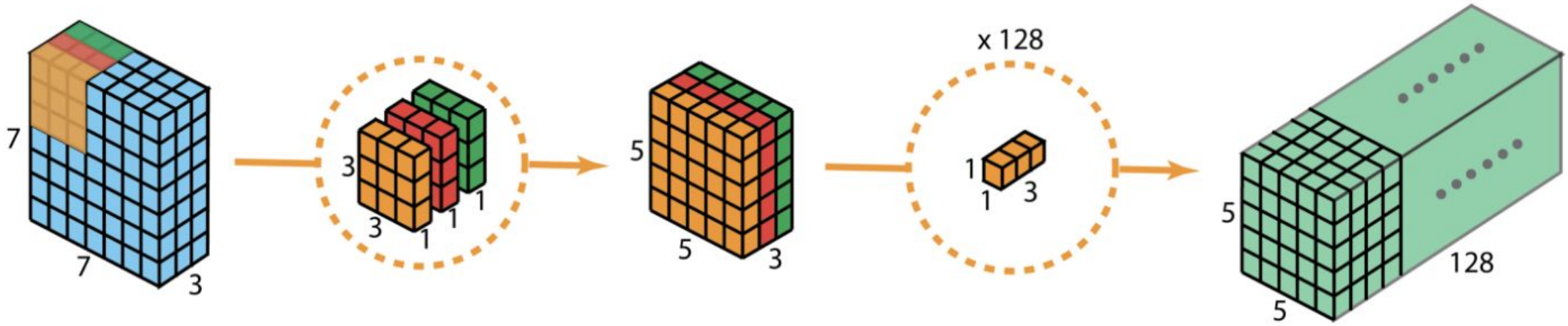
- Considering all output channels

Depth-wise Convolutions

- Shorthand for “Depth-wise separable convolutions”
- “Depth”-wise separable, because considering channels as “depth”, perform convolutions on them **independently**



Depth-wise Convolutions (Another Diagram)



Point-wise Convolutions

- “Point”-wise convolutions because **each pixel is considered independently**
- Considering just a single output channel:

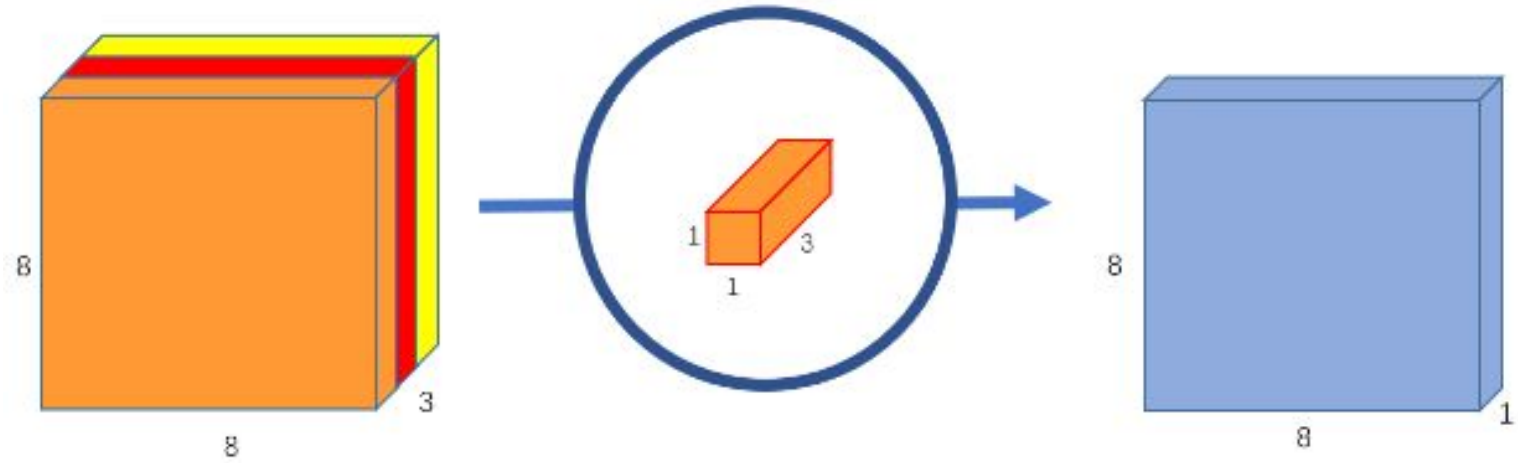


Image 7: Pointwise convolution, transforms an image of 3 channels to an image of 1 channel

Point-wise Convolutions

- “Point”-wise convolutions because **each pixel is considered independently**
- Considering all output channels:

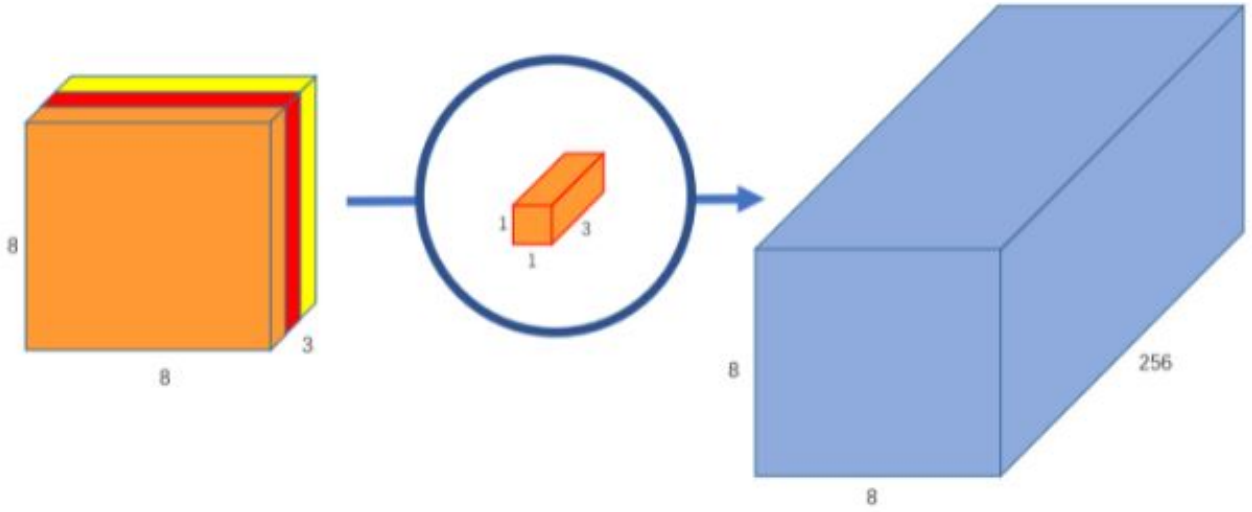


Image 8: Pointwise convolution with 256 kernels, outputting an image with 256 channels

Summary

- A normal convolution mixes information from both **different channels and different spatial locations (pixels)**
- A depth-wise convolution only mixes information **over spatial locations**
  - Different channels do not interact.
- A point-wise convolution only mixes information **over different channels**
  - Different spatial locations do not interact

MobileNetV2

- Again, to understand an architecture, we mostly need to understand its **blocks**.
- All that changes for blocks in different stages is the base # of channels

MobileNetV2

- The core block of MobileNetV2 has three steps:
  - Feature Mixing
  - Spatial Mixing
  - Bottlenecking Channels

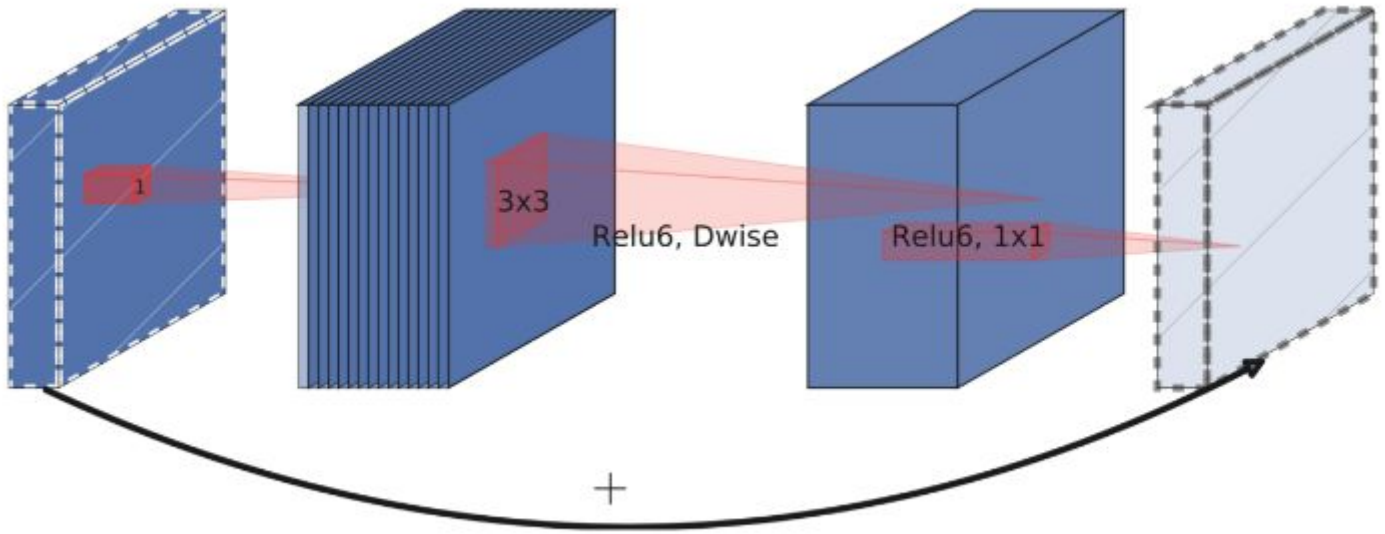


Fig. 6: Visualization of the intermediate feature maps in the inverted residual layer (source)

MobileNetV2: Feature Mixing

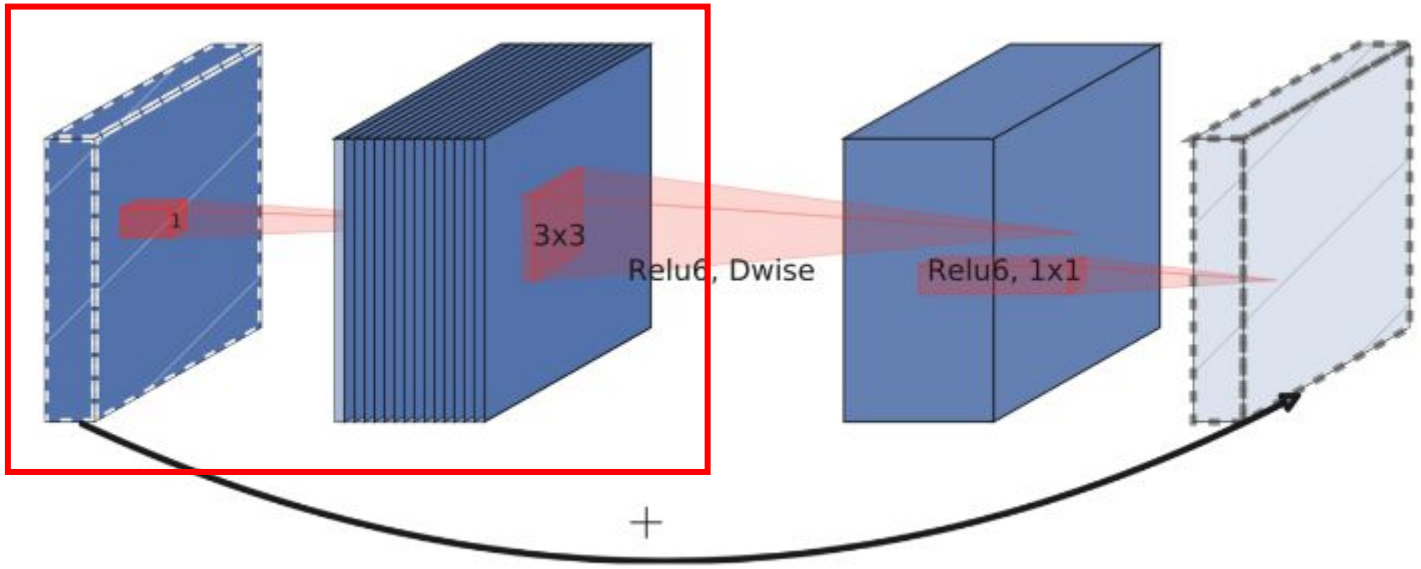


Fig. 6: Visualization of the intermediate feature maps in the inverted residual layer (source)

- A point-wise convolution that *increases the channel dimension* by an "expansion ratio"

MobileNetV2: Spatial Mixing

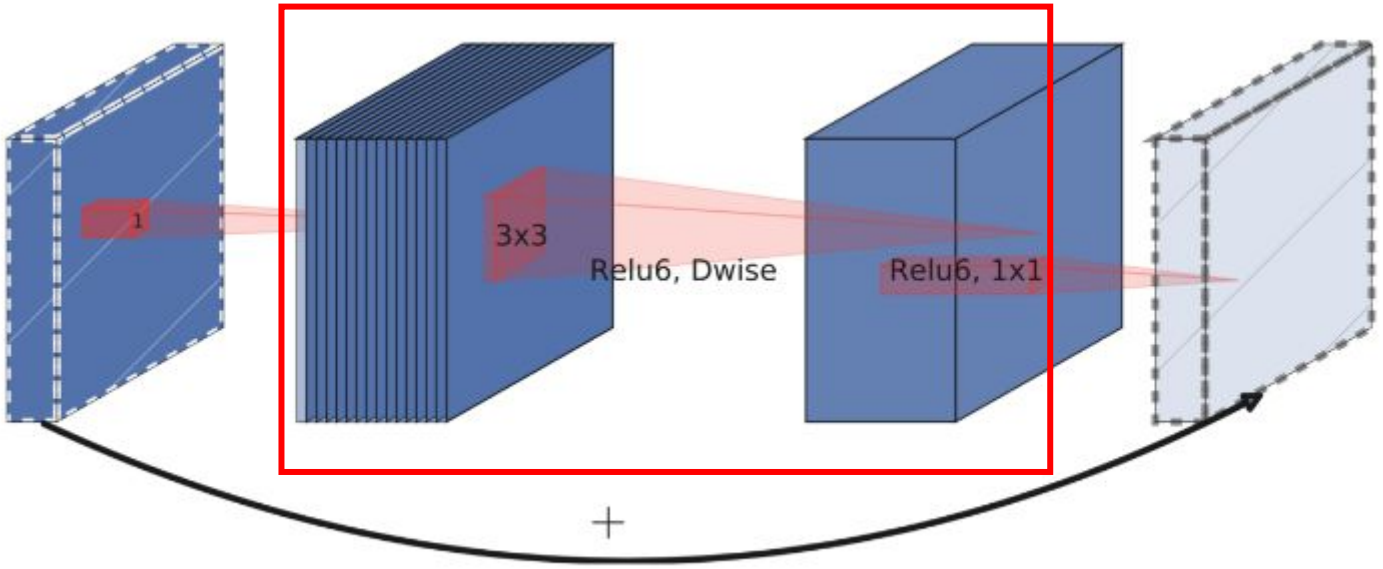


Fig. 6: Visualization of the intermediate feature maps in the inverted residual layer (source)

- A depth-wise convolution that communicates information over different spatial locations.

MobileNetV2: Bottlenecking Channels

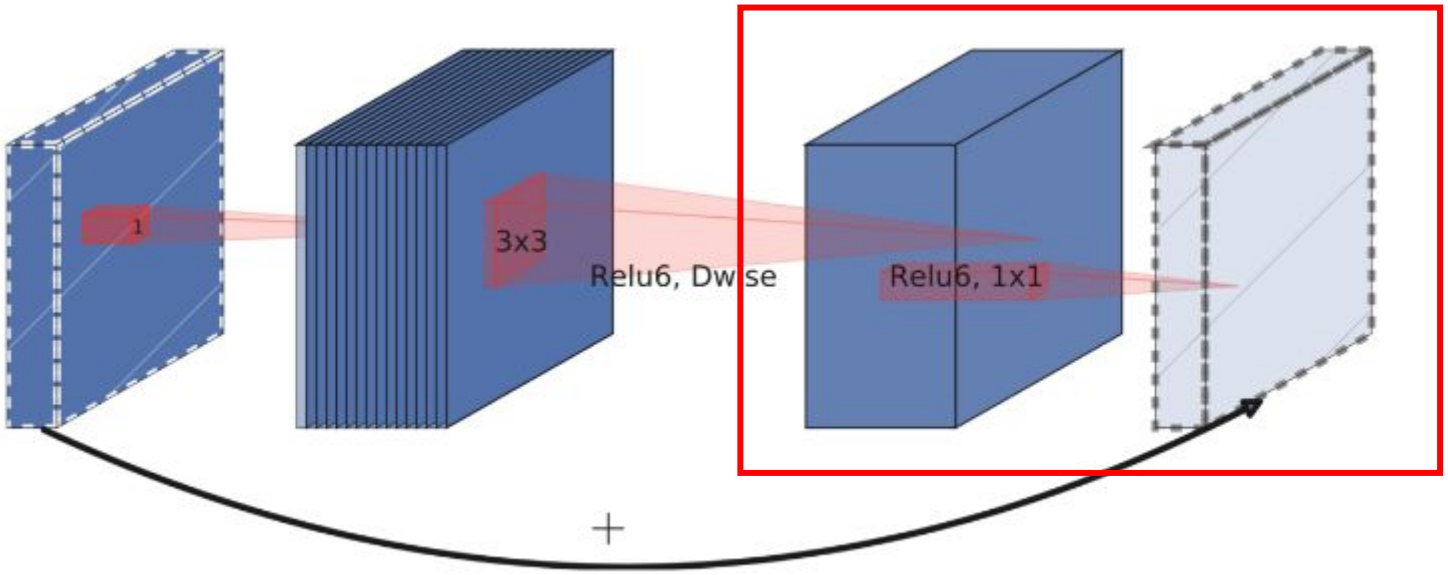


Fig. 6: Visualization of the intermediate feature maps in the inverted residual layer (source)

- Point-wise convolution to reduce channel dimension by the same expansion ratio.

ResNet

- Again, remember that to understand a paper, we just really need to understand its **blocks**.
- ResNet proposes 2 blocks: BasicBlock & BottleneckBlock
- The key point is residual connection
  - Actually, ResNet is older than MobileNetV2, so MobileNetV2 has this already

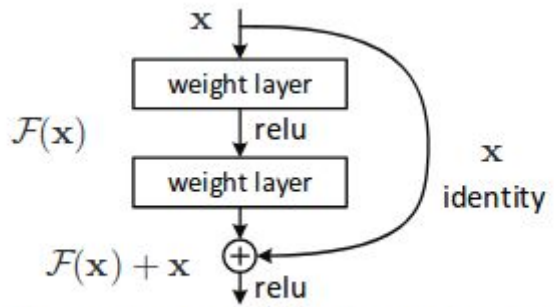
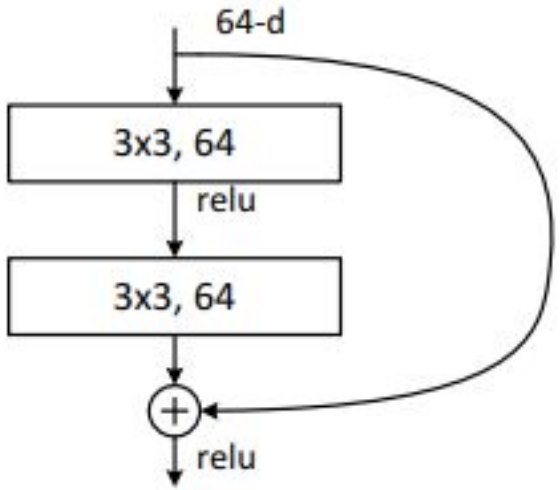


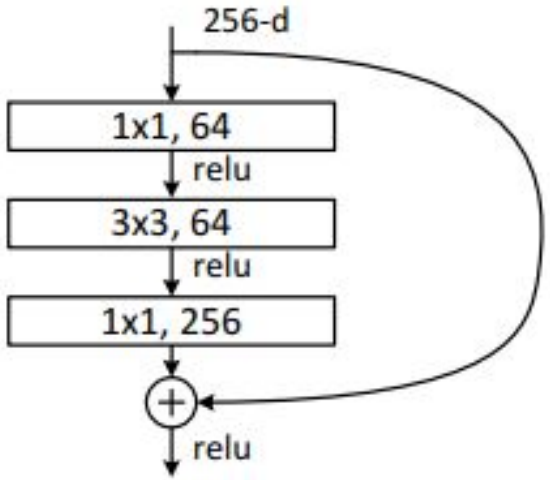
Figure 2. Residual learning: a building block.

ResNet: BasicBlock



- It’s just a regular 3x3 convolution (then BN, ReLU), another 3x3 convolution (then BN).
- Then, a skip connection adding input and output, then ReLU.

ResNet: BottleneckBlock



- A bit more involved.
- A 256-channel input goes through a point-wise convolution, reducing channels to 64.
- Then, a 3x3 regular convolution maintains channels at 64.
- Then, a point-wise convolution expands channels back to 256.
- Finally, the residual connection.

ResNet: Overall Architecture

Stem

Stage 1

Stage 2

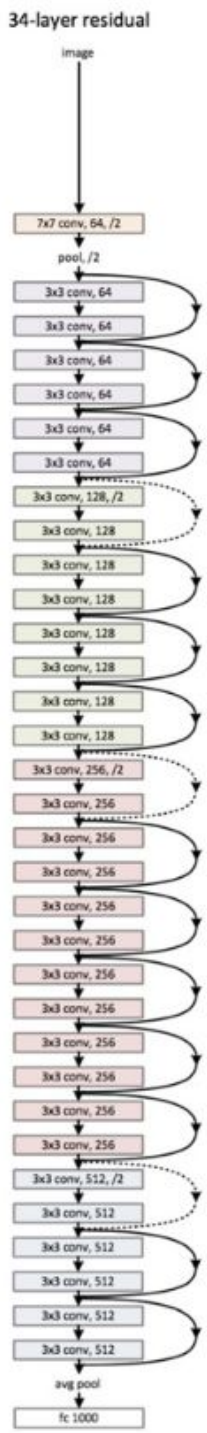
Stage 3

Stage 4

Classification Layer

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

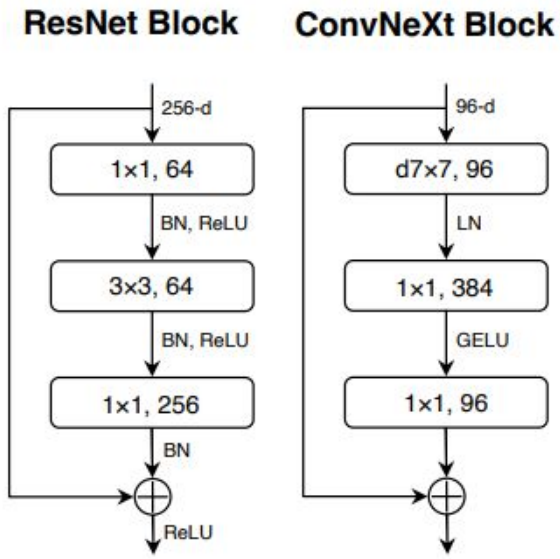
Figure 2. Sizes of outputs and convolutional kernels for ResNet 34



ConvNeXt

- This is a very new paper, a state-of-the-art architecture.
- However, **its intuitions are very similar to MobileNetV2.**
- Again, remember that to understand a paper, we just really need to understand its **blocks**.
- Just a single block type for ConvNeXt
- **Read the paper** for details on stages/channel sizes, etc.
  - We recommend **ConvNeXt-T size** which has less than 35M parameters.

ConvNeXt: Block



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

ConvNeXt vs MobileNetV2

ConvNeXt

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

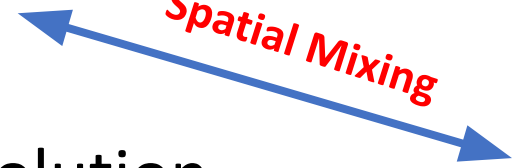
MobileNetV2

- A point-wise convolution increasing # of channels
- A 3x3 depth-wise convolution.
- A point-wise convolution decreasing # of channels
- Residual Connection

ConvNeXt vs MobileNetV2

ConvNeXt

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



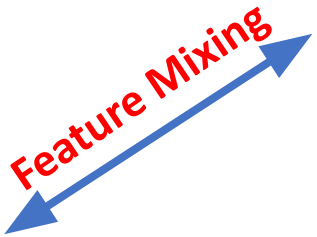
MobileNetV2

- A point-wise convolution increasing # of channels
- A 3x3 depth-wise convolution.
- A point-wise convolution decreasing # of channels
- Residual Connection

ConvNeXt vs MobileNetV2

ConvNeXt

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



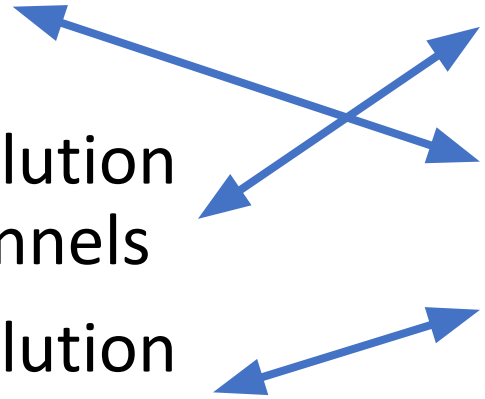
MobileNetV2

- A point-wise convolution increasing # of channels
- A 3x3 depth-wise convolution.
- A point-wise convolution decreasing # of channels
- Residual Connection

ConvNeXt vs MobileNetV2

ConvNeXt

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



MobileNetV2

- A point-wise convolution increasing # of channels
- A 3x3 depth-wise convolution.
- A point-wise convolution decreasing # of channels
- Residual Connection

Extremely Similar!

ConvNeXt vs MobileNetV2: Differences

- So what changed? Some things did change.
- The depth-wise convolution in ConvNeXt is larger kernel size (7x7).

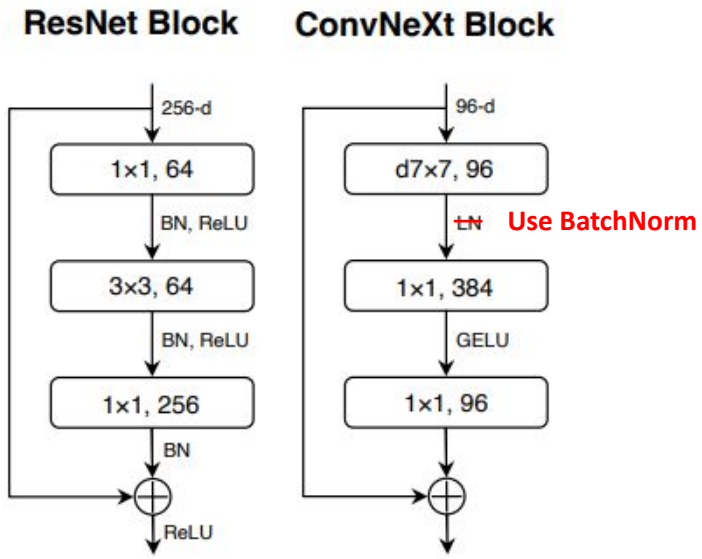
ConvNeXt vs MobileNetV2: Differences

- So what changed? Some things did change.
- The depth-wise convolution in ConvNeXt is larger kernel size (7x7).
- The order of spatial mixing & feature mixing are flipped.
  - In ConvNeXt, depth-wise convolution operates on lower # of channels.
  - In MobileNetV2, operates on higher # of channels.
- Channel Expansion Ratio in ConvNeXt is 4, MobileNetV2 is 6.

ConvNeXt vs MobileNetV2: Differences

- So what changed? Some things did change.
- The depth-wise convolution in ConvNeXt is larger kernel size (7x7).
- The order of spatial mixing & feature mixing are flipped.
  - In ConvNeXt, depth-wise convolution operates on lower # of channels.
  - In MobileNetV2, operates on higher # of channels.
- Channel Expansion Ratio in ConvNeXt is 4, MobileNetV2 is 6.
- ConvNeXt uses LayerNorm, MobileNetV2 uses BatchNorm.
  - **Note: We recommend using BatchNorm for this homework regardless.**
- ConvNeXt recommends training via AdamW, MobileNetV2 recommends SGD
  - **Note: We recommend using SGD for this homework.**

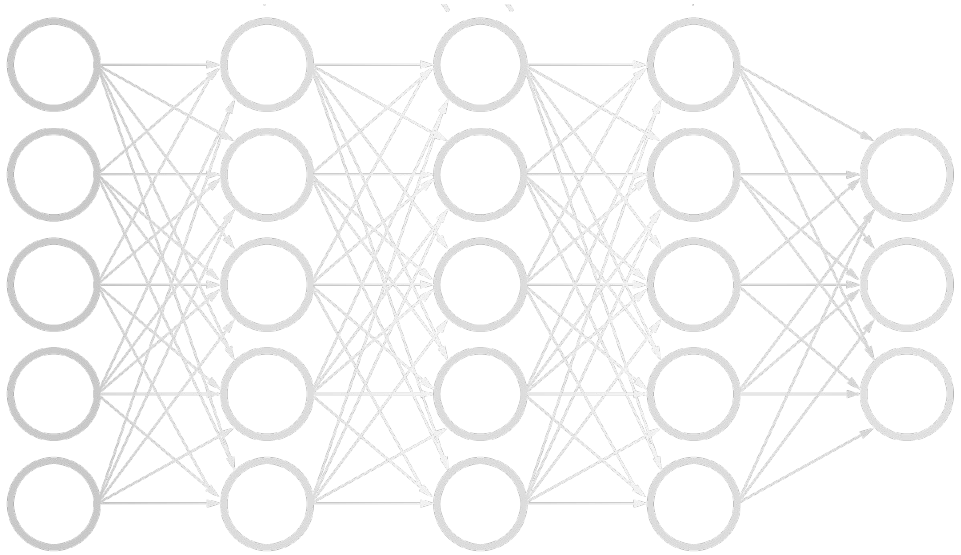
ConvNeXt vs MobileNetV2: Differences



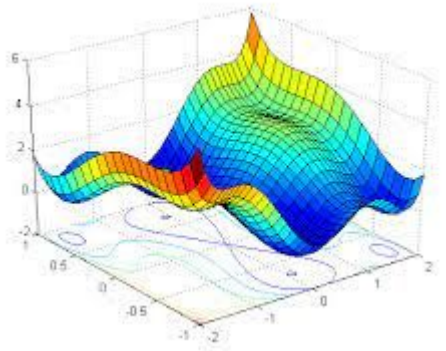
- Note that ConvNeXt has fewer BN/ReLU
  - GELU is just more advanced ReLU

Building Blocks

Input Image +
Transformations



Choice of Model



Training the model

The easy bit first….

Monitoring Training vs Validation Acc

- The standard intuition of “overfitting” is – if the training & validation gap is too large, you should stop training as it’s overfitting.
- However, in modern DL, this intuition is not as relevant.
- XELoss != Accuracy
  - Model can keep improving after training accuracy hits 100%.
  - There is recent research that finds that on some problems, training accuracy hits 100% at epoch 10 while validation accuracy is <50%. Then, on epoch 1000, validation hits 100%.
- Of course, we can’t train for that long, but train until validation stops improving.
  - Or just set a standard LR schedule/setup like “CosineAnnealingLR for 50 epochs” and just let it run. what I prefer to do.

How to tackle overfitting?

- There are *a lot* of different tricks to improving your CNN model.
- From the recent ConvNeXt paper:

(pre-)training config	ConvNeXt-T/S/B/L ImageNet-1K 224^2
optimizer	AdamW
base learning rate	4e-3
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.999
batch size	4096
training epochs	300
learning rate schedule	cosine decay
warmup epochs	20
warmup schedule	linear
layer-wise lr decay [6, 10]	None
randaugment [12]	(9, 0.5)
label smoothing [65]	0.1
mixup [85]	0.8
cutmix [84]	1.0
stochastic depth [34]	0.1/0.4/0.5/0.5
layer scale [69]	1e-6
gradient clip	None
exp. mov. avg. (EMA) [48]	0.9999

How to tackle overfitting?

- There are *a lot* of different trick to improving your CNN model.
- From the recent ConvNeXt paper
- What we recommend trying first:
  - Label Smoothing (huge boost)
  - Stochastic Depth
  - EMA
  - DropBlock (paper)
  - Dropout before final classification layer
- Then you can try the others
- Check out “Bag of Tricks for Image Classification with Convolutional Neural Networks”
  - https://arxiv.org/abs/1812.01187

(pre-)training config	ConvNeXt-T/S/B/L ImageNet-1K 224^2
optimizer	AdamW
base learning rate	4e-3
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.999
batch size	4096
training epochs	300
learning rate schedule	cosine decay
warmup epochs	20
warmup schedule	linear
layer-wise lr decay [6, 10]	None
randaugment [12]	(9, 0.5)
label smoothing [65]	0.1
mixup [85]	0.8
cutmix [84]	1.0
stochastic depth [34]	0.1/0.4/0.5/0.5
layer scale [69]	1e-6
gradient clip	None
exp. mov. avg. (EMA) [48]	0.9999

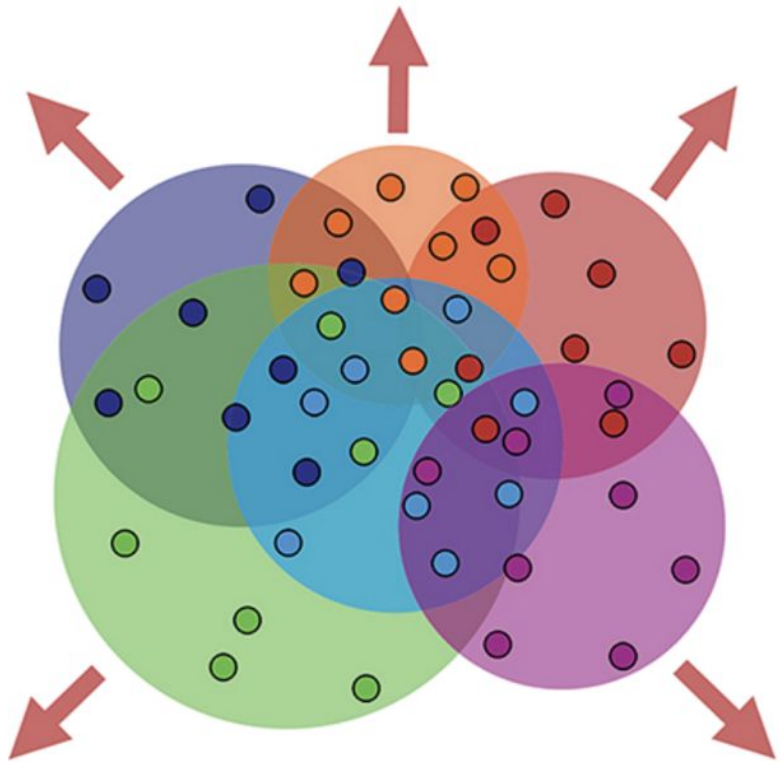
Let’s get real now….

Loss Functions

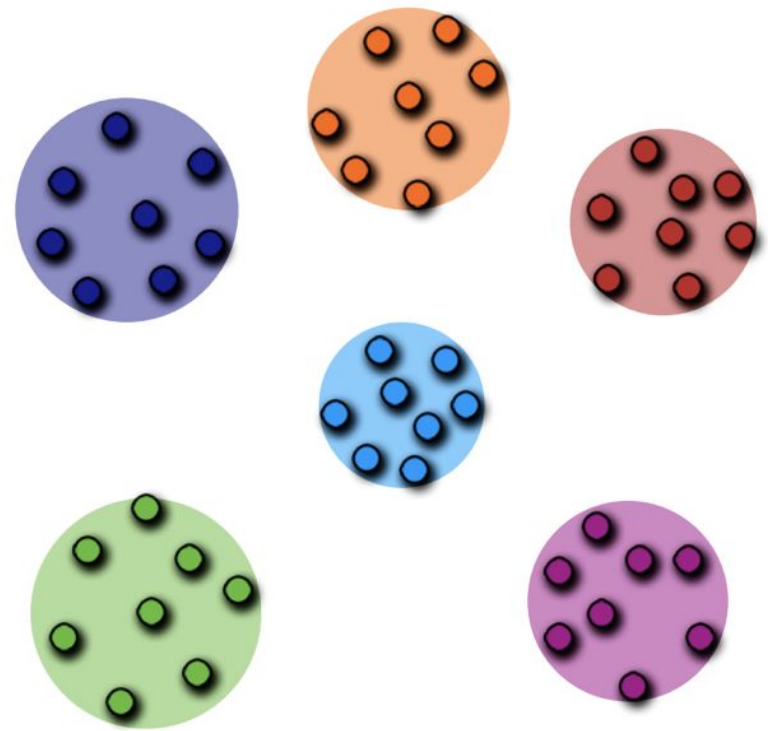
Face Verification + Face Recognition tasks (VGG Face2)

Types of Loss functions

Non Contrastive loss functions

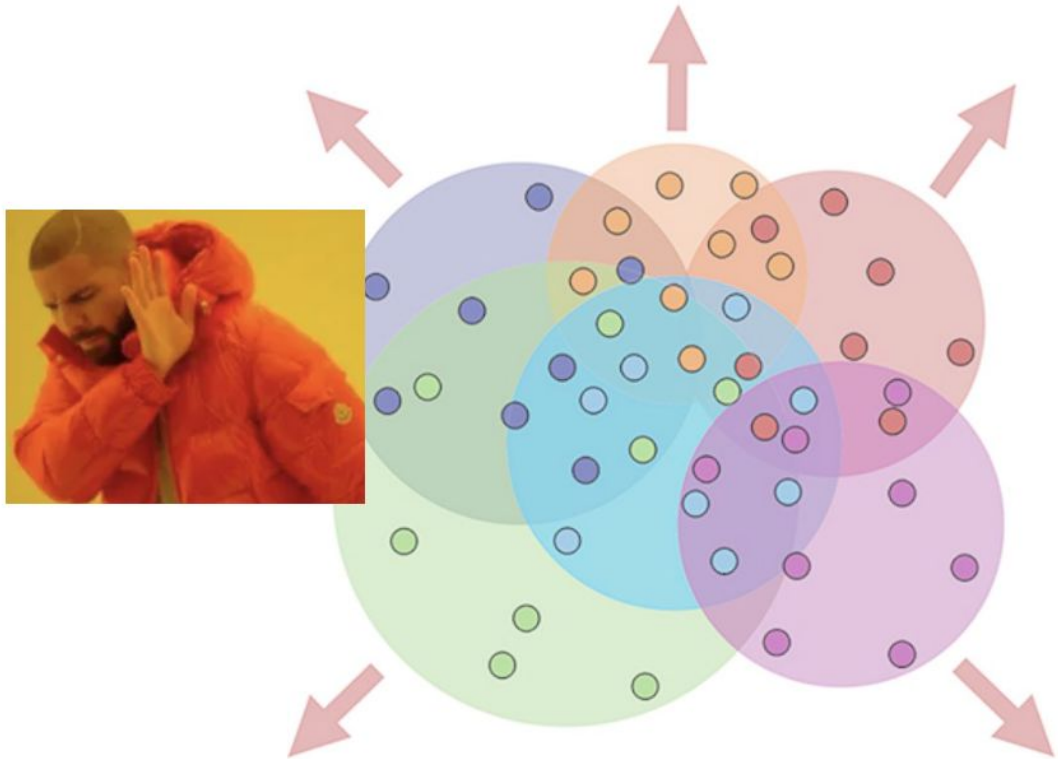


Contrastive-Losses



Types of Loss functions

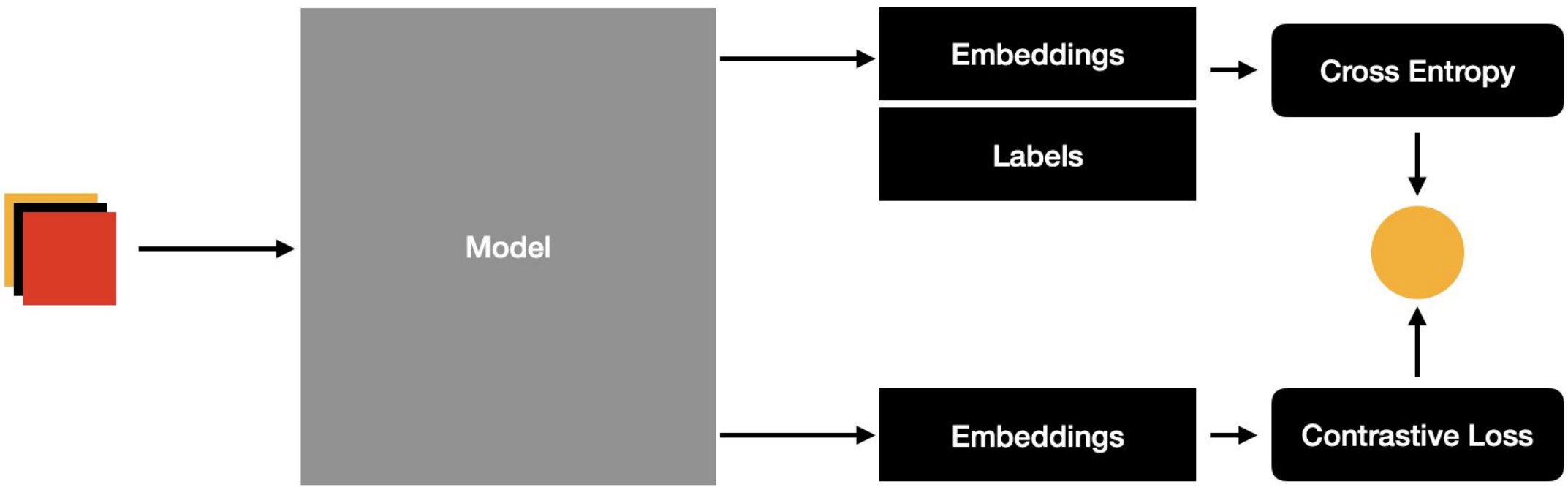
Non Contrastive loss functions



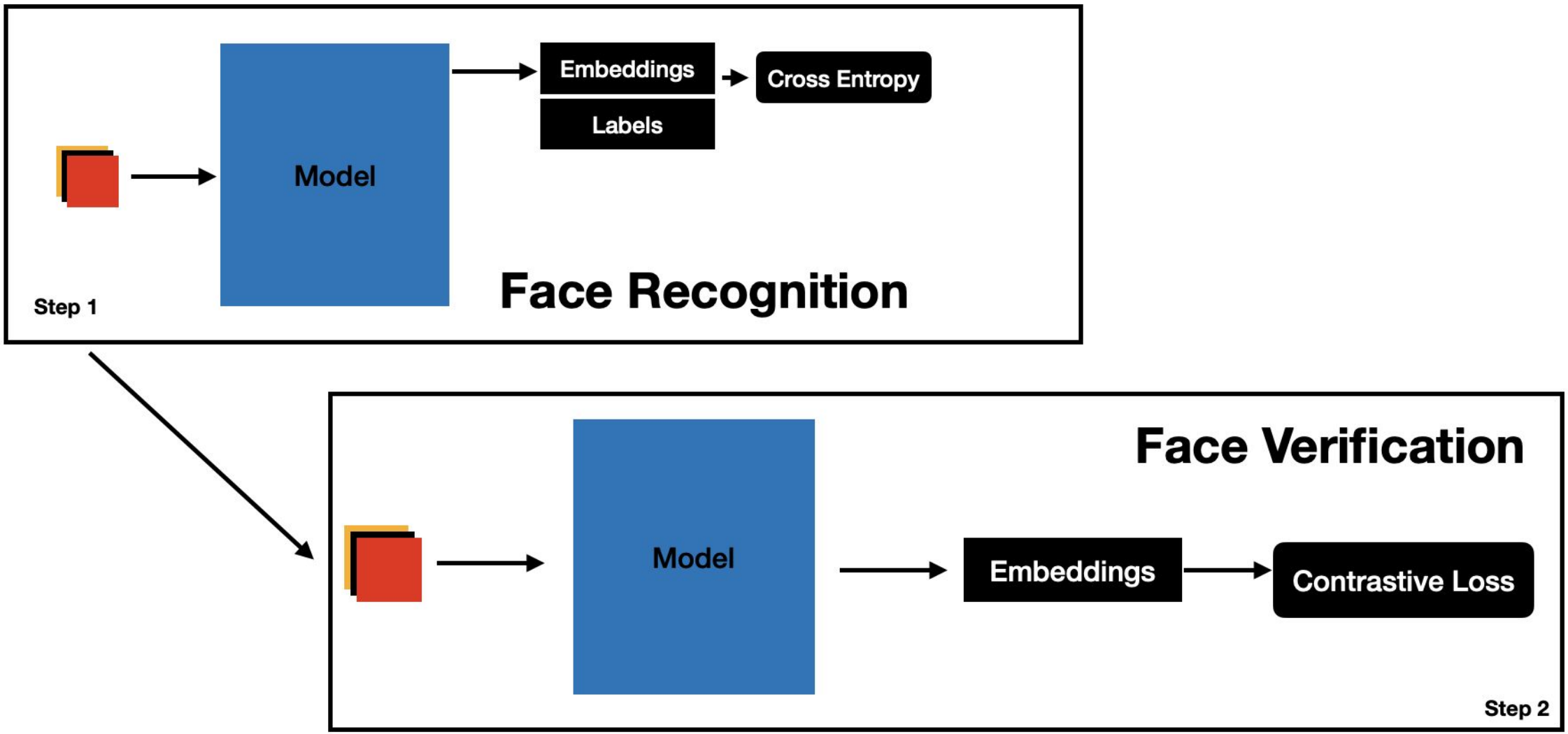
Contrastive-Losses

How to train models with such loss functions ?

Approach 1 - Joint Loss Optimization



Approach 2 - Sequential (Fine-tuning)



Types of Contrastive Losses

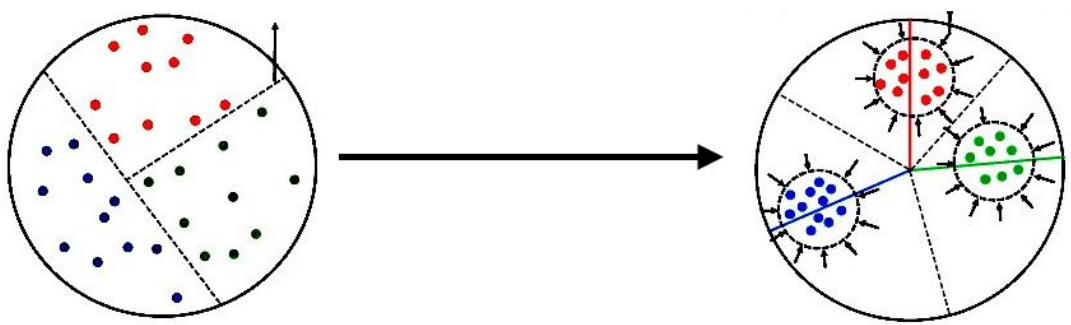
1. Centre Loss
2. Triplet Loss
3. Sphere Face (Angular Softmax)
4. CosFace Loss
5. ArcFace

Centre Loss

$$\mathcal{L}_C = \frac{1}{2}\sum_{i=1}^{m} \|\boldsymbol{x}_i - \boldsymbol{c}_{y_i}\|_2^2$$

- Increases the disparity between classes using softmax
- Increases inter-class distance by reducing intra-class Euclidean distance by assigning centers to each class.

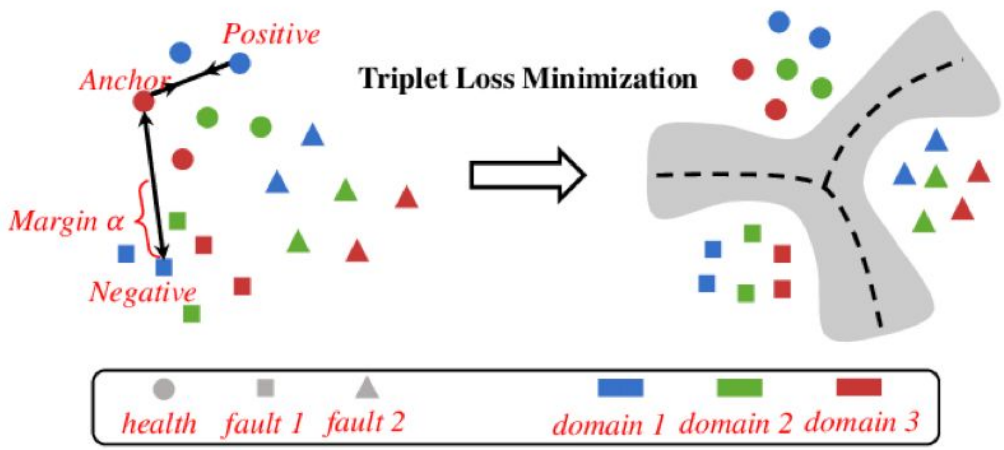
- Calculating the centre for each class, is difficult



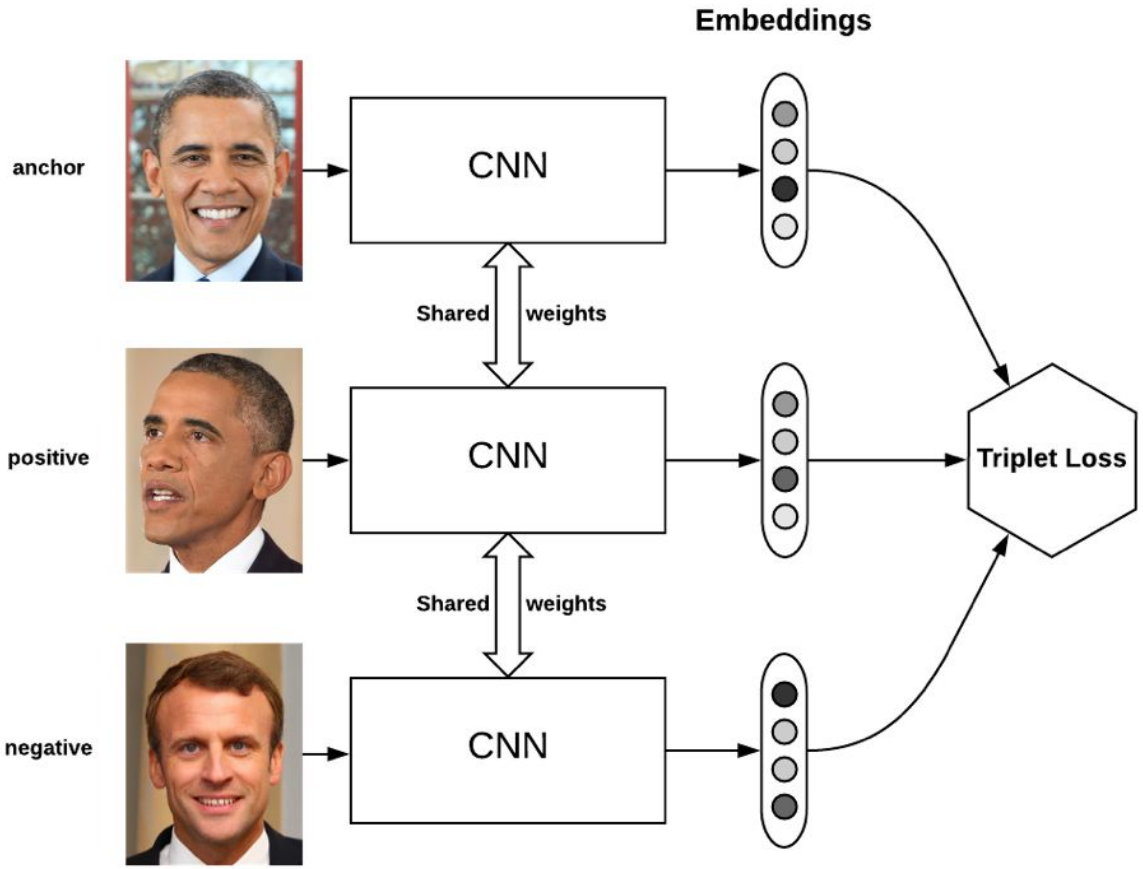
Triplet Loss

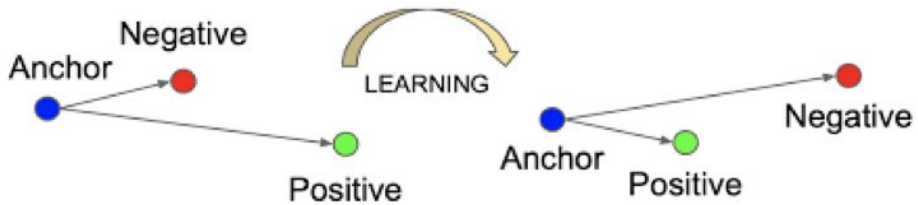
$$Loss = \sum_{i=1}^{N} \left[\| f_i^a - f_i^p \|_2^2 - \| f_i^a - f_i^n \|_2^2 + \alpha \right]_+$$

- Involves sampling 3 images, an anchor, a positive (same class as anchor) and a negative (different class from anchor).
- Use a p-norm distance function to increase the difference between anchor and negative whilst minimizing distance between anchor and positive.
- Sampling hard positives and hard negatives is key and difficult



Triplet Loss

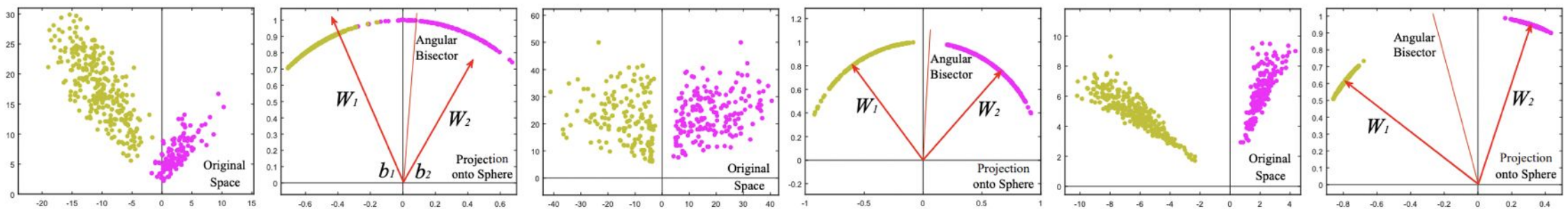




Sphere Face

$$L_{\text{ang}} = -\sum_i \ln \frac{\exp\left\{\|\mathbf{x_i}\| \cos(m \cdot \theta_{y_i,i})\right\}}{\exp\left\{\|\mathbf{x_i}\| \cos(m \cdot \theta_{y_i,i})\right\} + \sum_{j \neq y_i} \exp\left\{\|\mathbf{x_i}\| \cos(\theta_{j,i})\right\}}$$

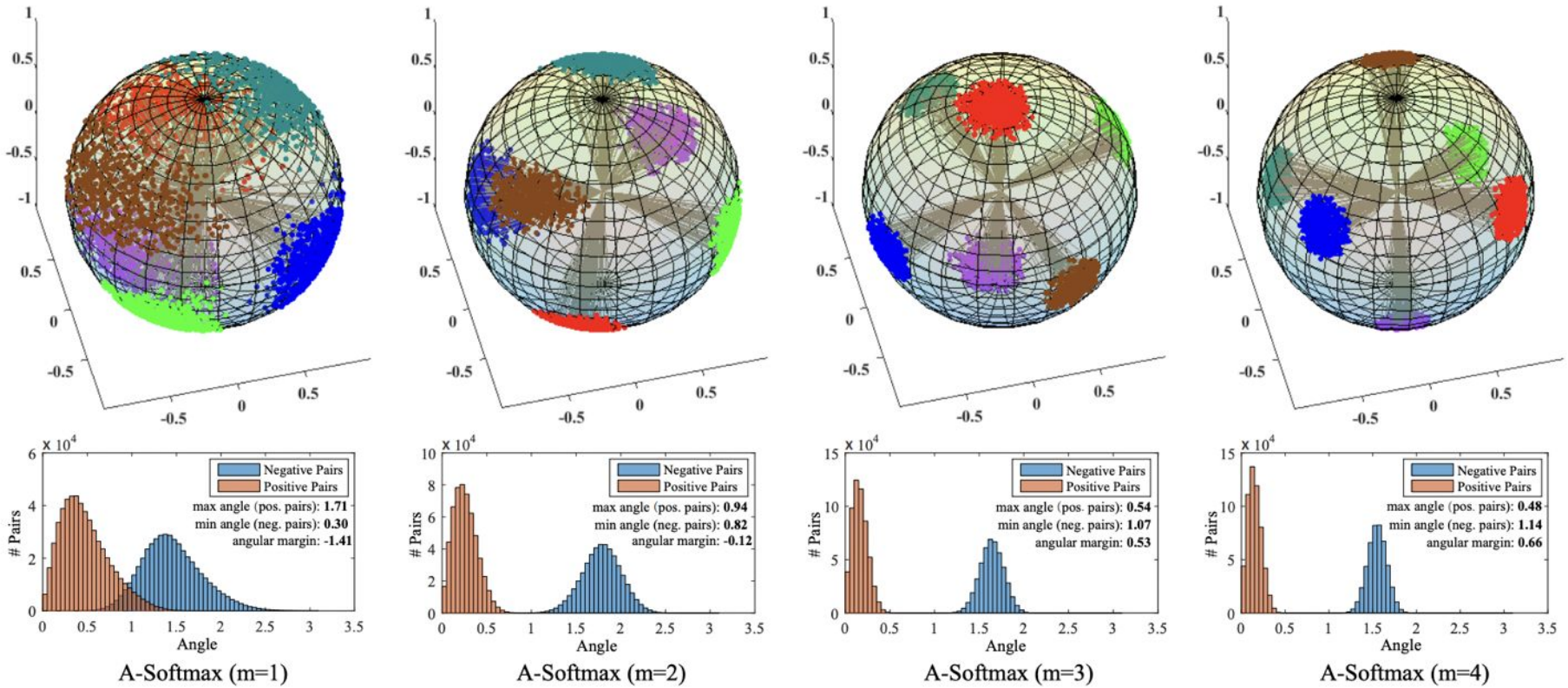
- Makes use of an angular margin, imposed by θ
- The learned features construct a discriminative angular distance equivalent to the geodesic distance on a hypersphere manifold
- θ :- denotes the type of decision boundary learned, which leads to different margins for different classes



(a) Original Softmax Loss (b) Original Softmax Loss (c) Modified Softmax Loss (d) Modified Softmax Loss (e) A-Softmax Loss (f) A-Softmax Loss

Sphere Face

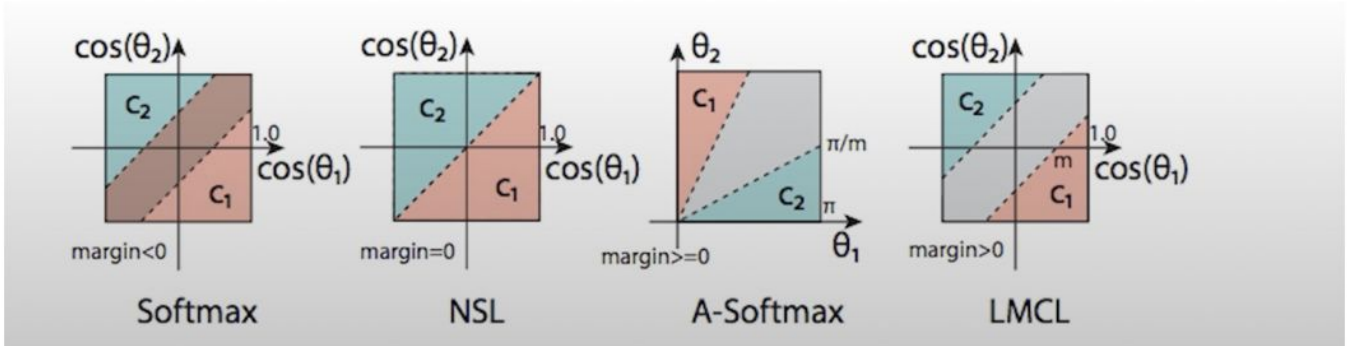
M ∝ Angular Margin



CosFace

$$L_{\text{lmc}} = -\frac{1}{N}\sum_{i=1}^{N}\ln\frac{\exp\left\{s\cdot\left(\cos\left(\theta_{y_i,i}\right)-m\right)\right\}}{\exp\left\{s\cdot\left(\cos\left(\theta_{y_i,i}\right)-m\right)\right\}+\sum_{j\neq y_i}\exp\left\{s\cdot\left(\cos(\theta_{j,i})\right)\right\}}$$

- **Similar to Sphere Face**
- **Forms the decision margin in cosine space rather than angular space.**

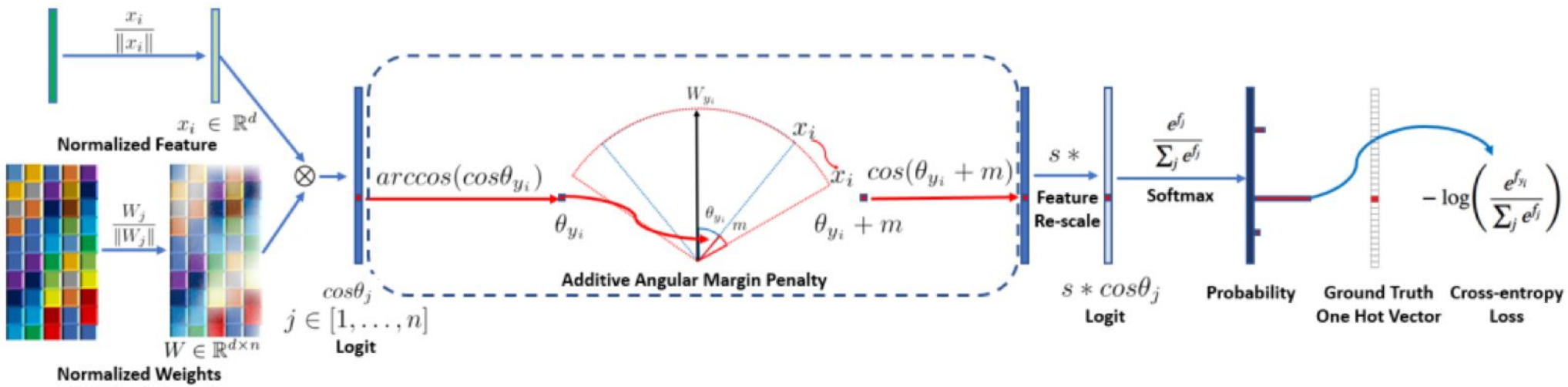


ArcFace

$$L = -\frac{1}{N}\sum_{i=1}^{N} \ln \frac{\exp\left\{s \cdot \cos(\theta_{y_i,i} + m)\right\}}{\exp\left\{s \cdot \cos\left(\theta_{y_i,i} + m\right)\right\} + \sum_{j \neq y_i} \exp\left\{s \cdot (\cos(\theta_{j,i})\right\}}$$

- Builds on the concepts of the sphere face and cos face.
- Replaced the multiplicative angular margin in CosFace, with an additive margin ‘m’

(a) Norm-Softmax (b) ArcFace



ArcFace

- **The additive factor of ‘m’ has found to lead to better convergence as compared to its multiplicative counterpart in Sphere Face.**

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, ~~SOTA~~) **Former SOTA**
  - https://arxiv.org/pdf/2102.06171v1.pdf
  - Quite doable actually