Jeremy Gozlan
Martin Cepeda

March 16, 2021
Final report

HIGH RESOLUTION FACE RECOGNITION

M2 Transverse project with IDEMIA

Project repo: `https://github.com/cepedus/HD-Faces`

Contents

1 Introduction **3**
1.1 Objective . . . 3

2 Dataset **3**

3 Proposed approach **4**
3.1 Squeeze-and-Excitation Networks . . . 4
3.2 Additive Angular Margin Loss . . . 5
3.3 High volume–Low resolution training . . . 6

4 Problems and intermediary results **7**
4.1 IR–SE and ArcFace repositories . . . 7
4.2 Problems encountered . . . 7

5 Experimental setup **8**
5.1 Architecture . . . 8
5.2 Model runs . . . 9

6 Results **9**
6.1 Baseline models . . . 10
6.1.1 Lowres32 . . . 10
6.1.2 Lowres144 . . . 10
6.1.3 Highres32 . . . 10
6.1.4 Highres144 . . . 11
6.1.5 Highlow144 . . . 11
6.2 Baseline finetuning . . . 12
6.2.1 Lowres144 + highres . . . 12
6.2.2 Modfeatures: Lowres144 + Lowres32 . . . 12
6.2.3 Modfeatures: Lowres144 + Lowres32 avec highres . . . 12
6.3 Model comparison . . . 13

7 Model evaluation analysis **14**
7.1 Embedding space analysis . . . 14
7.2 Point set registration . . . 16
7.2.1 Lowres144 . . . 16
7.2.2 Highres144 . . . 17
7.2.3 Highlow144 . . . 17
7.3 Weight analysis . . . 20

8 Conclusion and Discussion **22**

Acknowledgements **23**

References **23**

1 Introduction

Face recognition is a known problem that still *faces* (not pun intended) a lot of challenges when the number of possible identities is to be extremely large. Current state of the art techniques focus on the design of sophisticated loss function that enhance discriminative power of learnt features. Among such losses, angular-based losses such as ArcFace [1] and CosFace [7] include margins in order to maximise class separability and thus provide lesser ambiguity when doing classification.

Within the applications of face recognition algorithms we can cite identification and identity safety, two areas in which IDEMIA is arguably the world leader in the subject, with a presence in 180 countries, #1 in police biometric systems, civil identity solutions and U.S. driver's license issuance [3]. In order to continuously improve the performance of face recognition solutions, one area of research is the possibility to improve low resolution face recognition through the use of a small face dataset of high resolution.

1.1 Objective

This project's goal is to understand the possible benefits and limitations of improving low resolution face recognition through the use of a small face dataset of high resolution.

2 Dataset

For this project, IDEMIA has provided a face identity dataset consisting of:

```
/data
├── highres
│   └── ...
├── highres_eval
│   └── ...
├── hres_eval_pairs
│   └── hres
│       ├── data
│       └── meta
└── lowres
    └── ...
```

Dataset directory tree

- `highres` contains 10.000 faces distributed across 2.500 identities (4 images per class). Each image has a resolution of 144x144 (RGB) and inter-eyes distance of 48 pixels.
- `highres_eval` contains 6.000 faces distributed across 2.000 identities not present in `highres`, 3 images per class). Each image has a resolution of 36x36 (RGB) and inter-eyes distance of 12 pixels.
- `highres_eval_pairs` contains 12.000 faces: 3.000 matching pairs and 3.000 non-matching pairs randomly sampled from `highres`.
- `lowres` contains 1.000.000 faces distributed across 50.000 identities (20 images per class). Each image has a resolution of 36x36 (RGB) and inter-eyes distance of 12 pixels.

The pointers to image files are stored in the files `highres_ids.txt`, `lowres_ids.txt` and `highres_eval_ids.txt`. It is worth to be noted that the dataset is of important size (5 GB), that all three folders of images do not contain any overlapping identities and that transferring the uncompressed files is extremely slow due to the quantity of files.

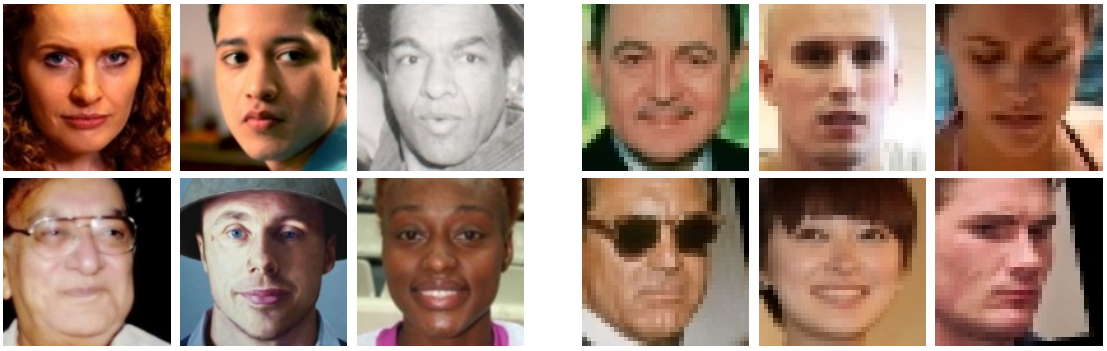


Figure 1: Sample high resolution faces (left) and low resolution faces (right)

Figure 2: Sample validation (high resolution) faces on two different identities

3 Proposed approach

In order to study the feasibility, benefits and limitations of improving high resolution face recognition, we can divide the face recognition pipeline in three main parts, namely 1) Backbone 2) Loss and 3) Training. These three components are available to experimenting under the repository cavaface.pytorch [5], which provides high-performance distribute parallel training framework for face recognition with pytorch, including various image feature extraction architectures and image loss networks, as well as various data augmentation(e.g., RandomErasing, Mixup, RandAugment, Cutout, CutMix, etc.) and bags of tricks for improving performance (e.g., apex, Label smoothing, LR warm-up, etc). In the following subsections we will described the chosen pipeline, which achieves the closest performances to SOTA in face recognition.

3.1 Squeeze-and-Excitation Networks

Introduced by [2], they are an extension to traditional residual architectures (such as ResNet) that introduce a novel architectural unit, the "Squeeze-and-Excitation" (SE) block, that adaptively re-calibrates channel-wise feature responses by explicitly modelling inter-dependencies between channels:

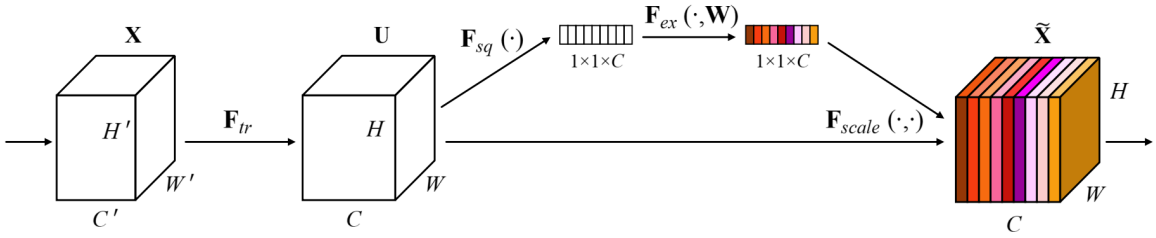


Figure 3: SE block: For any given transformation $\mathbf{F}_{tr}$ mapping the input $\mathbf{X}$ to the feature maps $\mathbf{U}$ where $\mathbf{U} \in \mathbb{R}^{H \times W \times C}$ (e.g. a convolution), we can construct a corresponding SE block to perform feature re-calibration. The features $\mathbf{U}$ are first passed through a squeeze operation, which produces a channel descriptor by aggregating feature maps across their spatial dimensions $(H \times W)$

The function of this descriptor is to produce an embedding of the global distribution of channel-wise feature responses, allowing information from the global receptive field of the network to be used by all its layers. The aggregation is followed by an excitation operation, which takes the form of a simple self-gating mechanism that takes the embedding as input and produces a collection of per-channel modulation weights. These weights are applied to the feature maps $\mathbf{U}$ to generate the output of the SE block which can be fed directly into subsequent layers of the network.

The *squeeze* operation uses global average pooling to generate channel-wise statistics, and the *excitation* operation employs a simple gating mechanism with a sigmoid activation. With this, it is possible to build IR–SE networks (ResNets with SE blocks) that achieve better performances in image classification and allow for adequately model channel-wise feature dependencies, which is of particular interest for face recognition.

3.2 Additive Angular Margin Loss

Abbreviated ArcFace, this loss was introduced by [1]. As the dot product between the DCNN feature and the last fully connected layer is equal to the cosine distance after feature and weight normalisation, we can utilise the arc-cosine function to calculate the angle between the current feature and the target weight. This result is then added to an additive angular margin to the target angle to get the target logit back again by the cosine function. Then, a re-scaling of all logits by a fixed feature norm allows to a direct plugging-in of a Softmax loss:

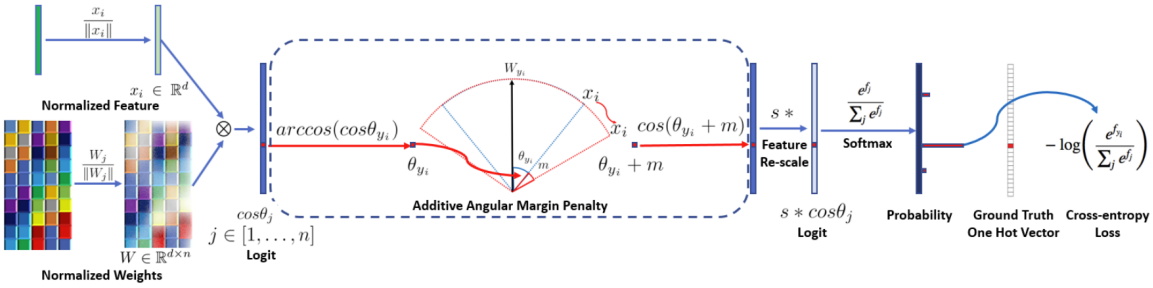
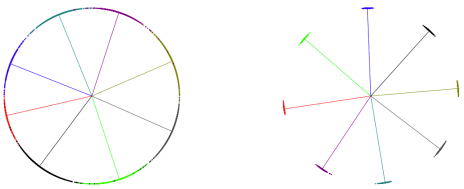


Figure 4: Based on the feature x_i and weight W normalisation, we get the $\cos\theta_j$ (logit) for each class as $W_j^T x_i$. We calculate the $\arccos\theta_{y_i}$ and get the angle between the feature x_i and the ground truth weight W_{y_i}. In fact, W_j provides a kind of centre for each class. Then, we add an angular margin penalty m on the target (ground truth) angle θ_{y_i} . After that, we calculate $\cos(\theta_{y_i} + m)$ and multiply all logits by the feature scale s. The logits then go through the softmax function and contribute to the cross entropy loss.

This loss allows for highly discriminative features for face recognition and improved training as the proposed additive angular margin has a better geometric attribute as the angular margin has the exact correspondence to the geodesic distance margin penalty in the normalised hypersphere:



(a) Softmax (b) ArcFace

Image taken from [1]

Toy examples under the softmax and ArcFace loss on 8 identities with 2D features. Dots indicate samples and lines refer to the centre direction of each identity. Based on the feature normalisation, all face features are pushed to the arc space with a fixed radius. The geodesic distance gap between closest classes becomes evident as the additive angular margin penalty is incorporated.

3.3 High volume–Low resolution training

The last aspect of a face recognition pipeline concerns precisely how to transfer the information of a large and dimensionally reduced dataset (at the expense of having less information per face) to a small high resolution dataset. A common ground on all possible approaches is the upsampling of low resolution images, which will ultimately allow to apply the trained architecture on both datasets. In order to do this, several leads can be considered:

1. Vanilla transfer learning by training in low resolution and fine-tuning with high resolution images.

2. 2 networks: a pre-trained architecture on low resolution is frozen and provides features from high resolution images in parallel with a simpler network during high resolution training. Features from both networks are concatenated and fed to a single classifier.

3. 2 networks with resolution awareness: same as above but the pre-trained network accepts only low resolution images (high resolution images are down-sampled for the first network and kept for the second, simpler network)

We will discuss this approaches more in detail in section 5.

4 Problems and intermediary results

4.1 IR–SE and ArcFace repositories

When starting this project, IDEMIA has been suggesting us to use the cavaface.pytorch implementation as our baseline code. It is high-performance distributed parallel training framework for face recognition that includes a large number of backbone networks and losses including IR–SE and ArcFace. At the beginning of this project, the library was suffering from a lack of updates, old dependencies and presented many errors. Also, being entirely focused on parallel GPU training, the code could not be run on our machines and so had to be entirely adjusted to be run on CPUs. While successful, the training time was so large and the computational requirements so big that we decided to found a way to run it on GPUs.

In the meantime, we also tried a second implementation of ArcFace in a TensorFlow based repository: InsightFace_TF [8]. This solution was focusing less on parallelization and CPU adjustments were manageable. However, in late November, cavaface.pytorch was updated which solved many problems and because of TensorFlow lack of computer vision tools, we switched to this newer version. As before, a non parallel CPU/GPU version has been made and tried to be run on cloud GPUs.

4.2 Problems encountered

This project while clear-forward in its direction was suffering from a small technical challenge, our computational limitations. Unlike small networks with low dimension inputs that can be run on CPUs, it requires an access to a powerful GPU cluster. Indeed, with one million images and a ResNet architecture, the computer we currently have cannot train such models. Most of the work we have been able to done so far was in the direction of solving this issue. We tested the following solutions:

- Google Colab[1]: Due to several abuses on GPUs, Google decided to restrict and limit the access to their GPUs. Which means after a few hours of training (less than one epoch), the instance is stopped. Secondly, transferring such a large number of images to Google Collab is extremely tedious and not stable.
- Qarnot[2]: With Qarnot, the main problem was the only ability to access CPUs. Additionally, data transfer to Qarnot is more than unstable resulting in constant upload errors.

[1] `https://research.google.com/colaboratory/faq.html`
[2] `https://computing.qarnot.com/en/FAQ`

We decided to contact the support multiple times first without much success and finally asked if GPUs could be allocated for this project. The answer we got was the fact that only a few GPUs were available and already overbooked and that the credits we got will only last a few hours. We decided to stop trying to get a Qarnot training running as CPUs were not a viable option.

- Microsoft Azure: Because of our student status, it seems that we can have access to 100 dollars of Azure credits per Polytechnique account. We investigated this possible path for performing our experiments, but due to the limited duration of the trial (1 month) and the extensive setup needed to have the running code and an interface to launch, monitor and recover the results, we did not go further into setting-up Azure.
- IDEMIA: IDEMIA team has been extremely comprehensible with this problem but because of confidentiality reasons and the fact that we were not employees, they could not justify an access to their computational power. The option we had was to deliver them a script that they could run for us on their GPUs. We used this approach to train our models in both `lowres` and `highres` datasets, recovering the network checkpoints and training traces to analyze results/perform fine tuning.

5 Experimental setup

5.1 Architecture

When it comes to neural network architectures for high-scale face recognition, multiple options exist and can be tested under the chosen repository [5]. The two parts of a face recognition pipeline are the embedding network and the classification features network, namely *backbone* and *head* as described in section 3.

- Backbone: Usually a CNN type architecture that is used to create embeddings for an image. Options include Resnet, Resnet IRSE, MobileNet, DenseNet or even GhosNet and many mores.
- Head: Taking as input the image embedding formed by the backbone, an head is used to separate different classes spatially. It includes ArcFace, CosFace or the Spherical losses.

For this project and due to limited computational resources, we decided to use as backbone the following network. A description of the network parts can be seen in more details in Figure 5:

- Input layer: It is composed Conv2d, a batchnorm and a ReLU activation.
- Body: For the body, we went with 8 IR-SE blocks (4 distinct blocks used twice). An example of a block can be seen in 5
- Output layer: Taking as input the last IR-SE block output, it forms the embeddings for the ArcFace.

```
(input_layer): Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (2): PReLU(num_parameters=64)
)
 (body): Sequential(
   (0): bottleneck_IR_SE(
     (shortcut_layer): MaxPool2d(kernel_size=1, stride=2, padding=0, dilation=1, ceil_mode=False)
     (res_layer): Sequential(
       (0): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
       (1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
       (2): PReLU(num_parameters=64)
       (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
       (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
       (5): SEModule(
         (avg_pool): AdaptiveAvgPool2d(output_size=1)
         (fc1): Conv2d(64, 4, kernel_size=(1, 1), stride=(1, 1), bias=False)
         (relu): ReLU(inplace=True)
         (fc2): Conv2d(4, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
         (sigmoid): Sigmoid()
       )
     )
   )
(output_layer): Sequential(
  (0): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (1): Dropout(p=0.4, inplace=False)
  (2): Flatten()
  (3): Linear(in_features=41472, out_features=128, bias=True)
  (4): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=False, track_running_stats=True)
)
```

Figure 5: Input, Body and Output layers of the Backbone network

Depending on whether the model is trained on the `lowres` or `highres` data, or the up-scale/downscale versions of them, the network input size was changed accordingly to either a (32,32,3) or a (144,144,3).

5.2 Model runs

Below is the list of models tried and their achieved performances on the highres.eval dataset which consisted of 6000 pair of identity matching and non matching pairs. Being of size (144,144,3), the evaluation pair images were downscaled to (32,32,3) in case the network was trained on low resolution images (not up-scaled). A detailed description is presented alongside with the obtained results in the following section.

6 Results

For each model, we show the corresponding curves for the training loss and validation accuracy during training, as well as the ROC curve for the last epoch.

6.1 Baseline models

Using only a single dataset for training, these models represent good baselines for comparing models.

6.1.1 Lowres32

This model was trained on the non up-scaled low resolution images (1 million images) and was evaluated on downscaled eval pairs.

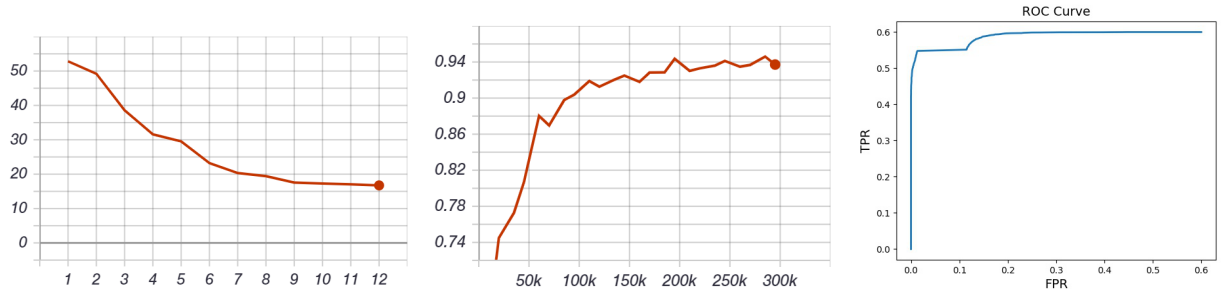


Figure 6: From left to right: Training loss, Validation accuracy and Validation ROC curve

6.1.2 Lowres144

Representing the most important baseline, this model was trained on the up-scaled low resolution images (1 million images) and was evaluated on the high resolution eval pairs (untouched).

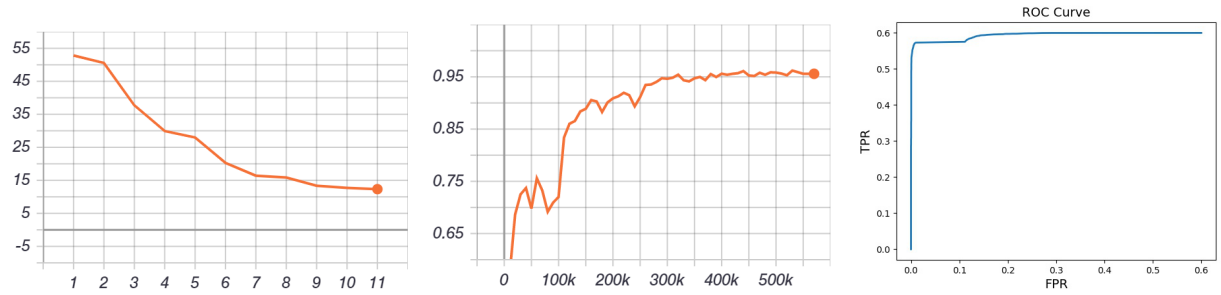


Figure 7: From left to right: Training loss, Validation accuracy and Validation ROC curve

6.1.3 Highres32

This model was trained on the down-scaled high resolution images (10 k images) and was evaluated on the down-scaled high resolution eval pairs. With only a few images in a low resolution, the training was extremely fast but yields poor performances.

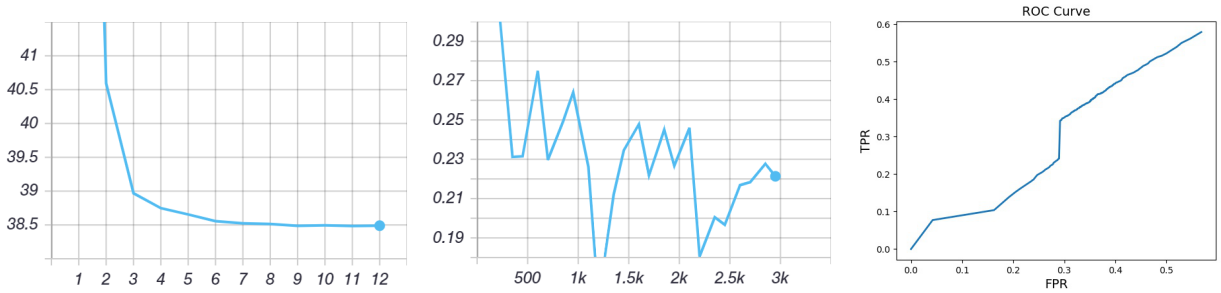


Figure 8: From left to right: Training loss, Validation accuracy and Validation ROC curve

6.1.4 Highres144

This model was trained on the high resolution images (10 k images) and was evaluated on the high resolution eval pairs. With only a few images in a high resolution, the training was also extremely fast but yields poor performances as well.

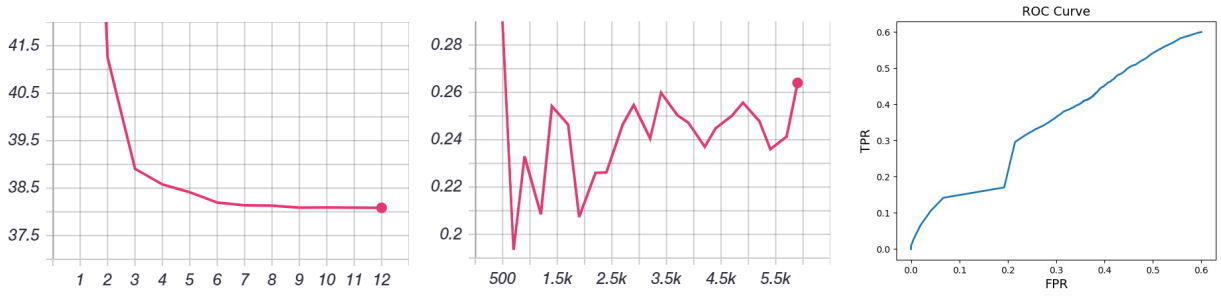


Figure 9: From left to right: Training loss, Validation accuracy and Validation ROC curve

6.1.5 Highlow144

This model was trained on both the high and up-scaled low resolution images (1M + 10k images) and evaluated on the high resolution evaluation pairs.

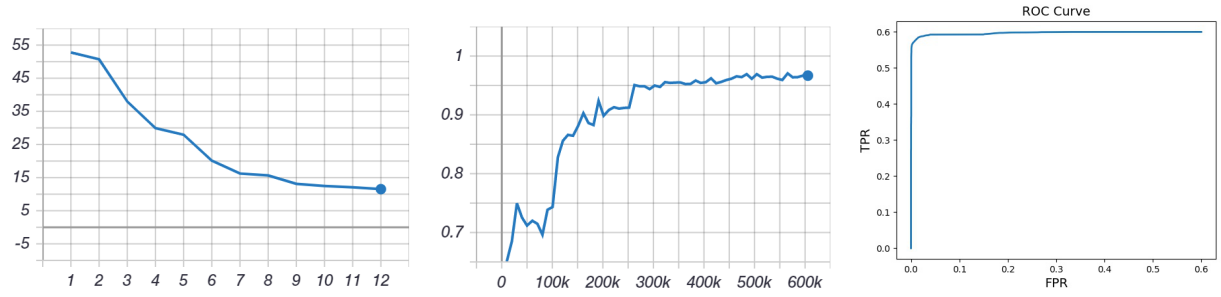


Figure 10: From left to right: Training loss, Validation accuracy and Validation ROC curve

6.2 Baseline finetuning

6.2.1 Lowres144 + highres

To see if a better model could be trained with the baseline Lowres144 and high resolution images, we fine-tuned it.

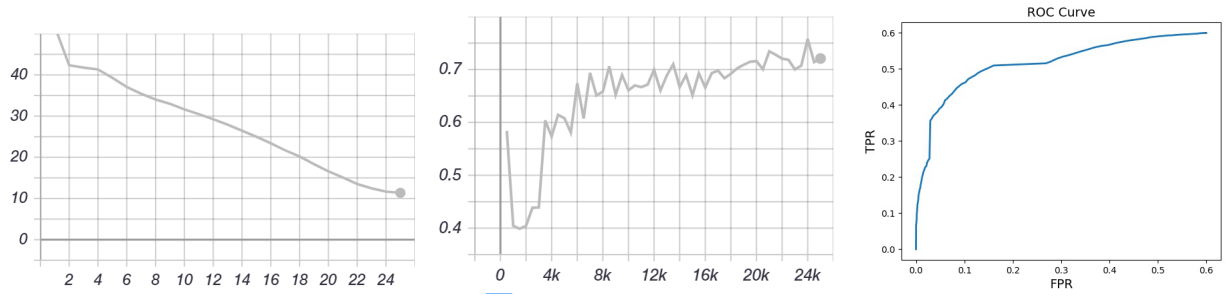


Figure 11: From left to right: Training loss, Validation accuracy and Validation ROC curve

6.2.2 Modfeatures: Lowres144 + Lowres32

Using two trained models, the Lowres144 and Lowres32, we used the two backbones and froze every layer. Embeddings of the two backbones were finally summed and passed to a new head and fine-tuned with high resolution images.

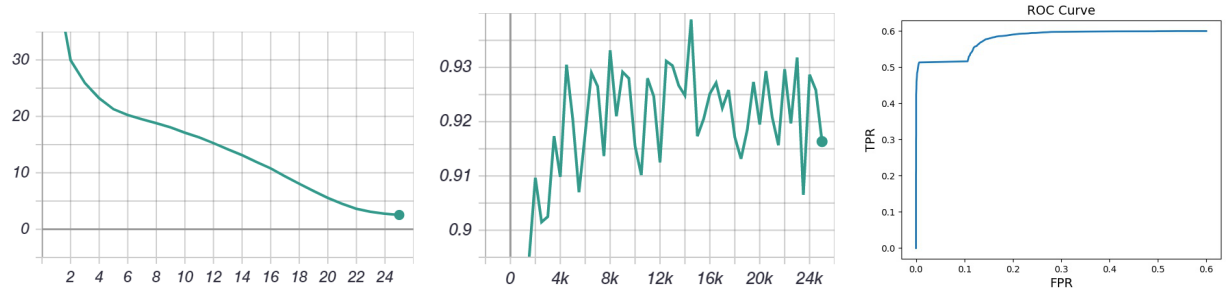


Figure 12: From left to right: Training loss, Validation accuracy and Validation ROC curve

6.2.3 Modfeatures: Lowres144 + Lowres32 avec highres

Using two trained models, the Lowres144 and Lowres32, we used the two backbones and froze every layer until the Output layer. Embeddings of the two backbones were finally summed and passed to a new head and fine-tuned with high resolution images.

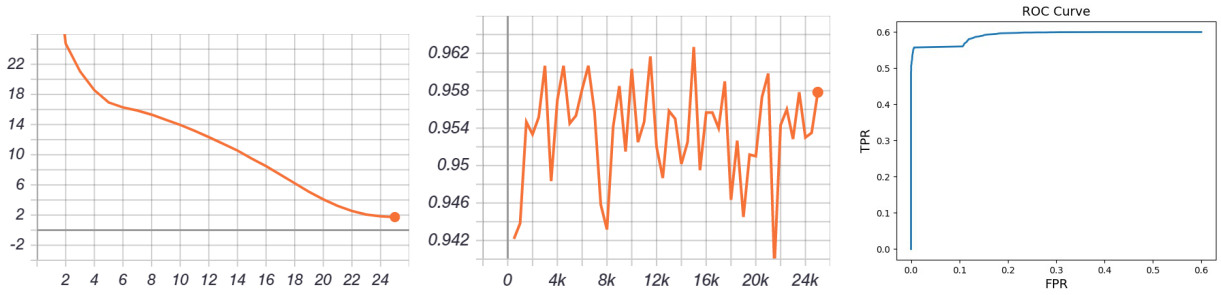


Figure 13: From left to right: Training loss, Validation accuracy and Validation ROC curve

6.3 Model comparison

Following the same color convention as in the previous sub-subsections, we obtained the following panorama for our models:

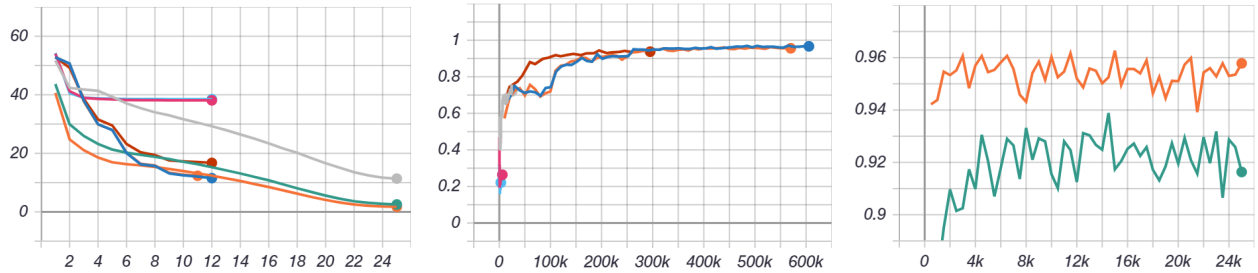


Figure 14: From left to right: Training loss for all models, Validation accuracy for Baseline models, Validation accuracy for Modfeatures models

We observe the following interesting behaviours:

- Models trained only on `highres` (32 and 144) perform very poorly due to the small amount of data compared to trainable parameters.
- The performance of the best model `highlow144` must be analyzed cautiously: high resolution images are present as well in the training set so the model has been given true high resolution and up scaled low resolution images. Nevertheless, the gain in accuracy is less than 1% which can be purely random as well.
- Even with a down-sampling of the validation set, the model `lowres32` is still competitive, which means that a priori it generalizes well.
- Baseline models `lowres32` and `lowres144` are the 2nd and 3rd best performing models, without any sort of fine-tuning.
- When using two models for finetuning with high resolution images, it seems that entirely freezing backbones (training the head only) resulted in poorer performances while fine-tuning the Output layers resulted in an sightly improved accuracy (by a small margin but with lesser stability).

7 Model evaluation analysis

While test scores such as Top-k accuracy, training losses or other metrics are interesting to compare models performances, we wanted to investigate the intrinsic behavior of models, especially the backbone embedding space and network weights. This is particularly interesting as the embedding space could have a particular structure, and its analysis could give more insight on the generalization capabilities of a face recognition system.

7.1 Embedding space analysis

The output layer of a backbone produces an embedding of size 128 representing an image. Having different resolution images used for training, either used separately, together or with fine-tuning, many questions arose when it came to the embedding produced by the backbones:

- Is there an adaptation problem where the network optimizes the two different resolutions images separately?
- Are embeddings properly placed in the desired space? For instance, in case we reduce the embedding dimension to 3, are the embeddings placed in a sphere and are they occupying the whole sphere? (Should be when using ArcFace).
- Are same identities embeddings close? Are different identities embeddings far away?
- In case a network has been trained solely on low resolution images up-scaled, is an up-scaled low resolution image embedding in the same space than an high resolution image embedding?
- If we fine-tune a network with another resolution dataset, are the embedding in the same space? If they are not in the same space, by how far? Can a linear transformation solve this problem?

First, we analyze the embedding distribution between models trained only on `lowres`/`highres`:

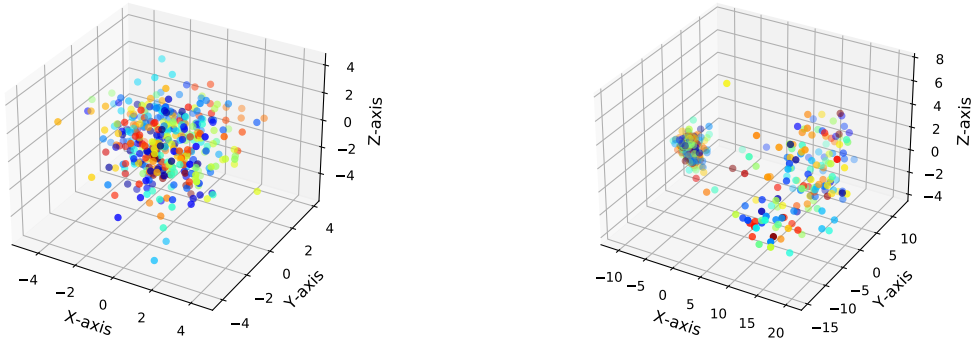


Figure 15: PCA (3 components) of non-normalised embeddings of `highres`. At left, lowres144 model and at right, highres144. Only 100 randomly selected identities are shown from the 2500 total identities in `highres`.

When looking at the Figure 15, we can see that the highres144 seems to have two distinct spaces, one distinguishable faces and one for others (many data points are compressed in the same location). This coincides with the analysis in the previous section: training only on highres gives a poorly performing model, which cannot distinguish identities due to the unbalance between training samples and trainable parameters. This contrast in the embeddings for each class can be seen also if we compare the intra-class vs. inter-class distance distribution. We note that all the good performing models (lowres32/144, finetune of lowres) exhibit this same behaviour. As a reminder, a perfect clustering is achieved if these two distributions are disjoint and the intra-class distance is notably smaller than the inter-class distance, this is, the classes are well separated:

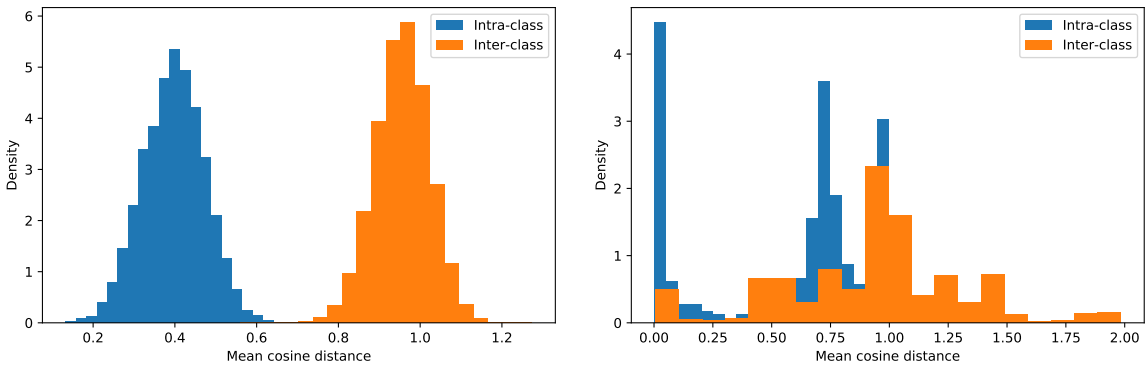


Figure 16: Histogram of pairwise distances of non-normalised embeddings of `highres`. At left, lowres144 model and at right, highres144.

We shift our attention now to the generalization capabilities of lowres144/32 and highlow144:

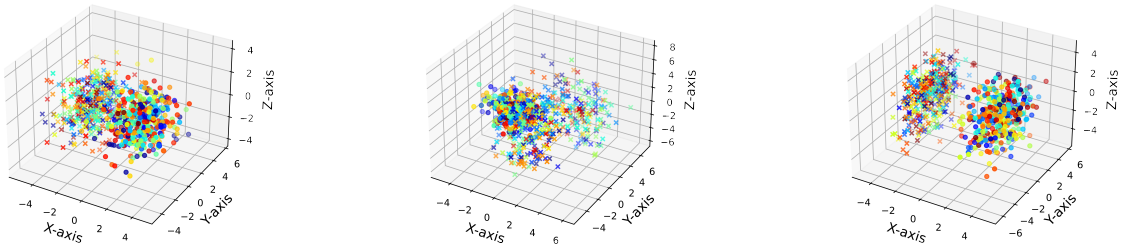


Figure 17: PCA (3 components) of non-normalised embeddings of `highres`. In crosses: image as-is. In circles: image down-scaled to 32x32 then up-scaled back to 144x144. At left, lowres144 model, at center, lowres32 and at right, highlow144. Only 100 randomly selected identities are shown from the 2500 total identities in `highres`. PCA was done on the concatenation of both model's embeddings.

We observe that the embeddings of downscale-then-upscaled images are almost linearly separable, at the same time that the same-class distances are very similar for both groups. This means that the models can distinguish if the input is a blurred image and classify it accordingly, but also that in a mixed dataset with varying resolutions, the accuracy will be greatly impacted as networks are not invariant to resolution (Adaptation problems).

7.2 Point set registration

As seen above during the embedding analysis, it seems that low resolution and high resolution version of a same image (also same identity) yield embeddings that lives in locally distinct but similar shape spaces. To see if the two embeddings of a same image could be placed at the same location, we decided to use Point Set Registration techniques. To do this, we first performed dimensionality reduction of the two desired set of embeddings formed by a same network with Principal Component Analysis and decided to only keep three dimension to treat each embedding as a 3D point. This gave us a 3D point cloud for each set of embeddings that we could fit to Coherent Point Drift [6] using the Python-CPD package [4]. Among the many techniques available within it, we decided to test the Rigid, Affine and Deformable transformation.

We show now the point clouds as in figure 17 and the corresponding alignment after fitting the Coherent Point Drift algorithm.

7.2.1 Lowres144

After reducing the dimensionality of the two set of embeddings (concatenated), we performed visualization of the points for each identity. (It is to be noted that for each identity, four possible images exist and so eight points are present in the graph).

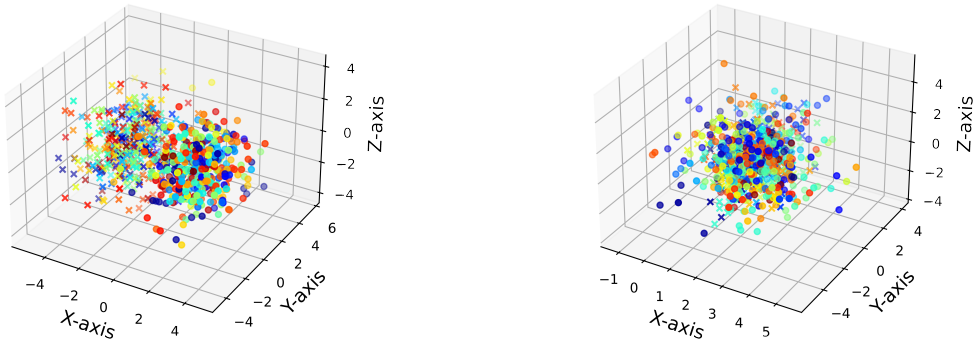


Figure 18: PCA (3 components) of non-normalised embeddings of `highres`. In crosses: image as-is. In circles: image down-scaled to 32x32 then up-scaled back to 144x144. At left, raw embeddings and at right, after CPD. Only 100 randomly selected identities are shown from the 2500 total identities in `highres`. PCA was done on the concatenation of both model's embeddings.

7.2.2 Highres144

Following the same procedure than for the Lowres144, the results we obtained were the following:

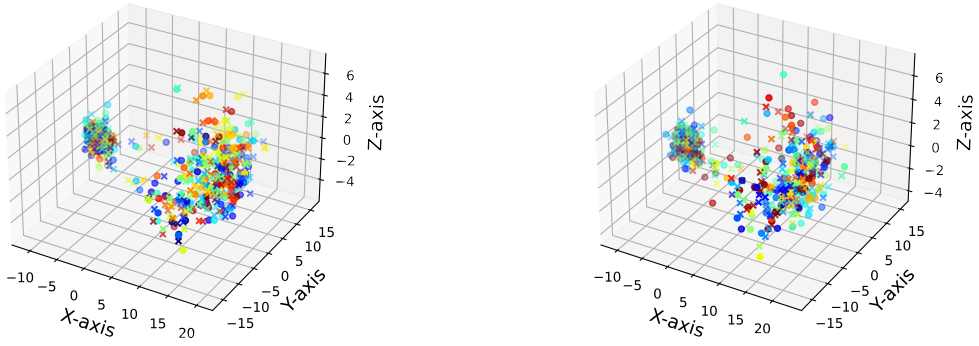


Figure 19: PCA (3 components) of non-normalised embeddings of `highres`. In crosses: image as-is. In circles: image down-scaled to 32x32 then up-scaled back to 144x144. At left, raw embeddings and at right, after CPD. Only 100 randomly selected identities are shown from the 2500 total identities in `highres`. PCA was done on the concatenation of both model's embeddings.

7.2.3 Highlow144

With a network trained using low resolution images upscaled and the high resolution images, we supposed that this network might have a chance to put the embedding space in the same location.

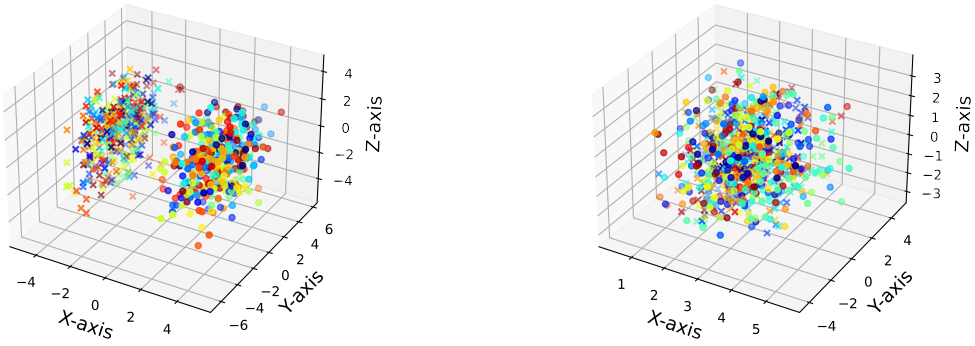


Figure 20: PCA (3 components) of non-normalised embeddings of `highres`. In crosses: image as-is. In circles: image down-scaled to 32x32 then up-scaled back to 144x144. At left, raw embeddings and at right, after CPD. Only 100 randomly selected identities are shown from the 2500 total identities in `highres`. PCA was done on the concatenation of both model's embeddings.

Only in this last model we observe a coherent alignment, where the model had even a more clear difference between blurred and orginal samples (compared to figure 17)

A last analysis can be performed on the effect of Point Set Registration. We show now the joint histogram of class distances for lowres144 and highlow144:

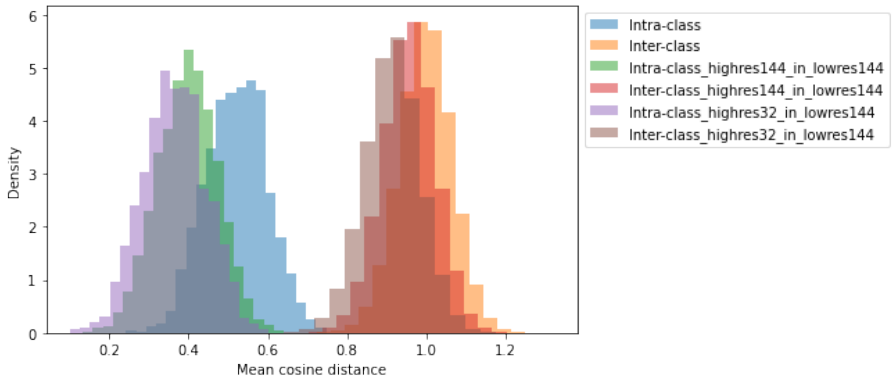


Figure 21: Distribution of class distances for `highres` images as-is and down-then-upscaled. We have 3 pairs of histograms: one for all embeddings without distinction and 2 within each set of `highres` images for Lowres144.

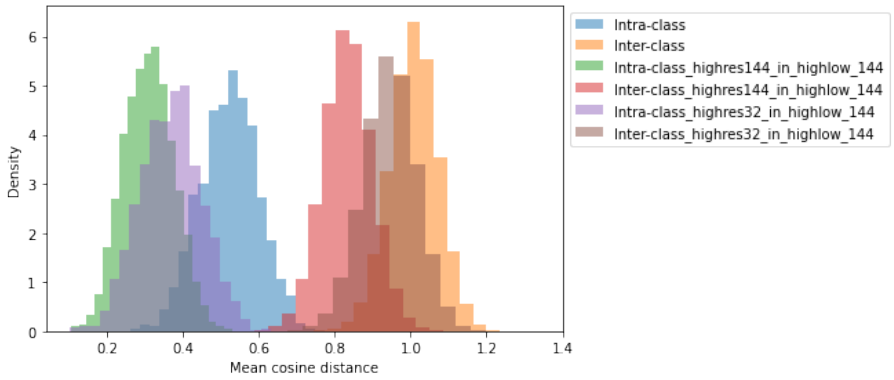


Figure 22: Distribution of class distances for `highres` images as-is and down-then-upscaled. We have 3 pairs of histograms: one for all embeddings without distinction and 2 within each set of `highres` images for Highlow144.

We observe the behaviour that was introduced with figures 17, 18 and 20. Although the classes are well separated for each model (greater inter-class distance and smaller intra-class distance, both with low dispersion), when we mix both types of input images, this is, with varying initial resolution, we observe that it is harder to distinguish each identity as the intra-class and inter-class distances overlap. This phenomenon is accentuated in the Highlow144 model (which is visually represented in fig. 20).

Finally, we study the effect of PSR in making good clusters when using mixed high resolution and down-sampled high resolution images:

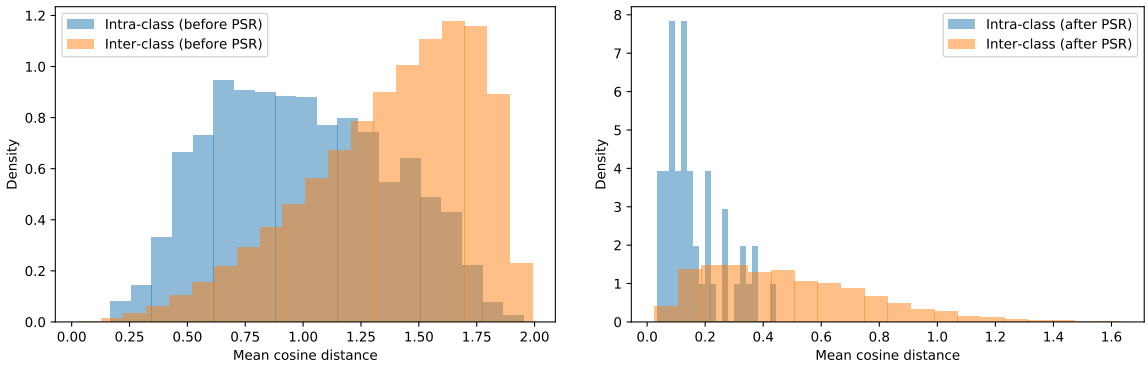


Figure 23: Histograms of cosine distance for Lowres144 before (left) and after (right) Coherent Point Drift. These distances are computed over the PCA embedding due to the point-alignment algorithm. 50 identities were selected at random among the 2500 available to fit Coherent Point Drift.

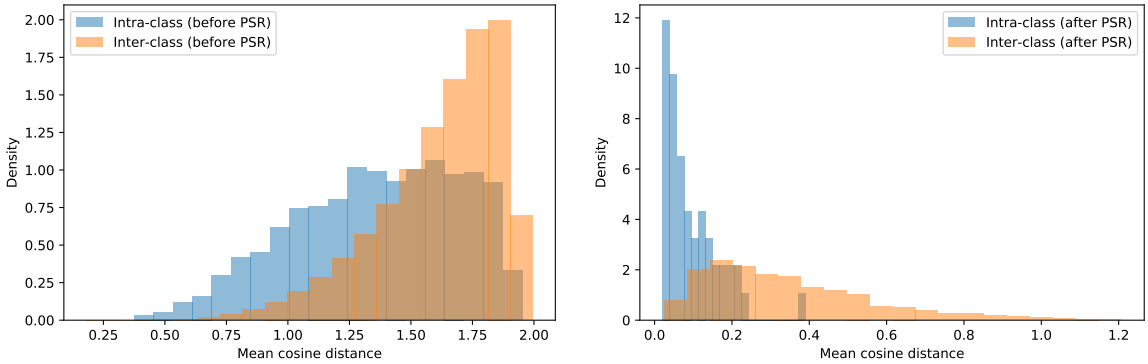


Figure 24: Histograms of cosine distance for Highlow144 before (left) and after (right) Coherent Point Drift. These distances are computed over the PCA embedding due to the point-alignment algorithm. 50 identities were selected at random among the 2500 available to fit Coherent Point Drift.

In this last histograms, although the analysis is less direct as they show distances computed over the PCA representation of the embeddings. It shows a) that the PSR algorithm has the effect of bringing much closer the same identities (peaks in intra-class distances after applying the algorithm) and b) reducing the overall distances between as-is and down-scaled-then-up-scaled images.

An important observation is that Coherent Point Drift is class agnostic, which means that having a peak in intra-class distances after its application means that the spatial structure itself of the embeddings is similar, thus reinforcing the hypothesis that both models distribute the identities in the same manner, but "shift them" when they have different resolution, as observed in figures 17, 18 and 20.

7.3 Weight analysis

Because it seemed that high resolution and low resolution images are placed at different locations within a space, we wanted to understand this behavior in the backbone networks. We decided to take the Lowres144 and HighLow144 (same architectures like any other models), and to analyze the convolution filters of the first layers produced by these networks.

First the Conv2d located in the Input layer that comprises 64 3D filters. After visualization, it seems that the HighLow144 network was able to learn precise and useful kernels such as edge detectors for instance. In the other hand, the Lowres144 produced closed to random kernels for the same Conv2d.

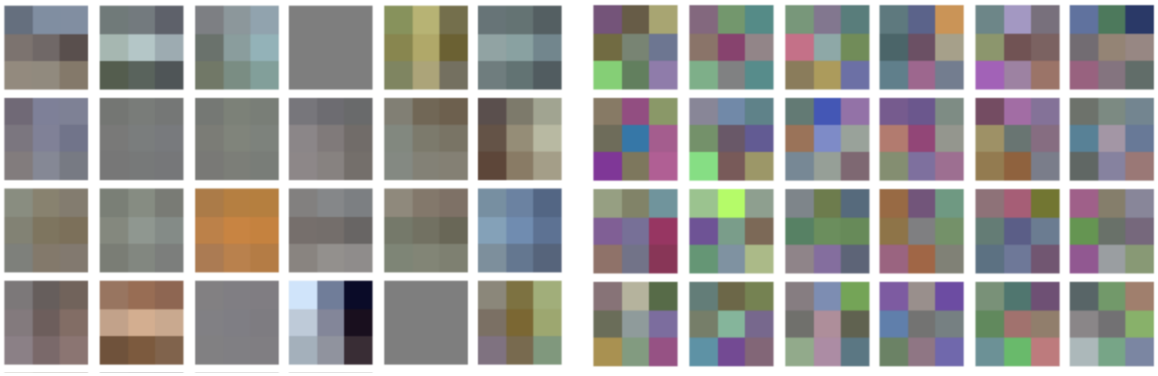


Figure 25: Few kernels for the first Conv2d of the Highlow144 (left) and Lowres144 (right)

Looking at the Input layer convolution, we can analayze the cumulative sum of the 64 filters. By looking at the difference, it may seem that the filters are not the same (can also be deduced from the graphical representations above).

```
Input layer convolution:  Highlow144 kernel sum tensor(853.2804)  Lowres144 kernel sum tensor(969.2402)
Difference Highlow144 and Lowres144 Input layer convolution kernel sum:  tensor(-115.9598)
```

Figure 26: Input layer convolution filters sum for Highlow144 and Lowres144

With multiple convolutional layers present in all IR SE blocks but with more than 3 channels, we could not plot them and so we decided to calculate the cumulative sum of the kernel per layer and to compute the difference per layer between the two networks.

```
torch.Size([64, 64, 3, 3])
Backbone body module 0 : Conv1 kernel sum tensor(17818.4453) , Conv3 kernel Sum: tensor(1854
6.9941)
Backbone2 body module 0 : Conv2 kernel sum tensor(18142.9219) , Conv3 kernel Sum: tensor(1730
7.6816)
Difference backbone - backbone2: Conv1: tensor(-324.4766) , Conv2: tensor(1239.3125)

torch.Size([512, 512, 3, 3])
Backbone body module 7 : Conv1 kernel sum tensor(1237735.6250) , Conv3 kernel Sum: tensor(105
1512.2500)
Backbone2 body module 7 : Conv2 kernel sum tensor(1218573.) , Conv3 kernel Sum: tensor(114558
8.8750)
Difference backbone - backbone2: Conv1: tensor(19162.6250) , Conv2: tensor(-94076.6250)
```

Figure 27: Kernel cumulative sum of the first two convolutional layer of block 0 (Top) and block 7 (Bottom) and their differences

While these differences might push us to think that these convolutional layer filters are indeed different, we cannot conclude as we are not able to understand these metrics properly.

8 Conclusion and Discussion

This research oriented project goal was to investigate the effect of a small dataset of high resolution images to help on a high scale low resolution face recognition task, conversely whether we can transfer the learning of many low-resolution images to high-resolution recognition. After training and testing a wide variety of models using either a single or the two possible datasets, we see that the addition of high resolution images does not seemed to improve the accuracy by a significant margin.

However, one noticeable effect was the separation of high and low resolution images into different spaces. Through the use of transformation techniques, this separation can be drastically reduced which might be interesting in term of classification improvement. If we had more time, it could have been interesting to introduce adversarial techniques to make high resolution and up scaled low resolution image on the same space.

As for now, networks leans to distinguish images based on the ”true” resolution which is a problem for high scale face recognition when multiple datasets of different sizes are to be mixed.

Acknowledgements

We want to particularly thank all IDEMIA team and especially our advisor Mr. Damien Monet which was a great help to us in so many ways. Even if the start of the project was difficult due to limited computational resources, we were able to properly run the models and gather valuable insights and recommendations thanks to IDEMIA's help. This project subjet was really interesting and allowed us to see classification in a different manner through the usage of other and more sophisticated architectures and losses.

Finally, we would like to thank our master directors, Mrs Marie Paul Cani and Mr Erwann Scornet that made this project possible.

References

[1] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. ArcFace: Additive Angular Margin Loss for Deep Face Recognition, 2019. URL https://ieeexplore.ieee.org/document/8953658.

[2] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu. Squeeze-and-excitation networks, 2019. URL https://ieeexplore.ieee.org/document/8701503.

[3] IDEMIA. Corporate brochure. https://www.idemia.com/sites/corporate/files/about-us/download/idemia-corporate-brochure-2020_0.pdf, Jun 2020.

[4] S. Khallaghi. Python-CPD, 2020. URL https://github.com/siavashk/pycpd.

[5] Y. Li and L. Chi. cavaface.pytorch: A Pytorch Training Framework for Deep Face Recognition, 2020. URL https://github.com/cavalleria/cavaface.pytorch.

[6] A. Myronenko and X. B. Song. Point-set registration: Coherent point drift. *CoRR*, abs/0905.2635, 2009. URL http://arxiv.org/abs/0905.2635.

[7] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu. CosFace: Large Margin Cosine Loss for Deep Face Recognition, 2018. URL https://ieeexplore.ieee.org/document/8578650.

[8] C. Wei. Insight Face in TensorFlow, 2018. URL https://github.com/auroua/InsightFace_TF.