# **"Hal9000"** Introduction to Machine Learning

## Introduction

Image retrieval is the task of finding similar or relevant images to a given query image in a collection. Unlike image classification, which aims at predicting at which class an image will likely belong to, image retrieval aims to discriminate images based on their visual features. Therefore the current purpose relies entirely on a task which indeed defines a threshold up to which two pictures can be expected to belong to the same identity by a confidence level rather than classify identities based upon the pool of inputs the model has been trained on. This supports tasks like face recognition, where the goal is to identify a person by comparing their face to a dataset of other faces. A face recognition system should therefore be able to evaluate two inputs as belonging to either the same identity or not regardless of having them being explicitly used in the training set (Melekhov et al., 2016).

Currently the aforementioned task is addressed by Deep Learning (DL) models which analyze the facial features of the input image by means of a Convolutional layers architecture and are generally trained upon *maximizing* as well as *minimizing* the representational embedding distance of two different and equal identities respectively. Consequently any DL-based face recognition model is trained over minimizing a *contrastive* loss function by means of an optimization algorithm in order to have a representative image embedded representation which can be further compared with other output for face discrimination. Images are then represented in a *n* dimensional space according to the dimension of the output vector and can be then grouped in clusters based upon the similarity displayed. The recognition system needs to compare the embedding representations of a *query* image against those of a given test set by means of a similarity score, namely a distance metrics that state how much two representations are close to each other.

The current work's purpose is to implement a face recognition system by utilizing an image retrieval approach. Given an input image, a set of *pre-trained* DL face recognition models use facial feature extraction techniques to represent the image as an embedded vector of features. The system then compares this feature vector to the feature vectors of other faces stored in the gallery in order to measure the similarity between the input and the known faces. The output of the system is a list of potential identities ranked by their similarity scores, ordered by either *Euclidean* distance or the *Cosine* similarity. The choice of metrics is therefore crucial to accurately measure the similarity between facial feature vectors, providing a quantitative measure of the closeness between two pictures representation, allowing the system to rank images as belonging to the same identity or not.

## Methods: models implemented

In order to effectively address the designated objective, the face recognition pipeline employed in this study incorporates a two-stage approach. The primary objective of this methodology is to initially perform facial extraction from the complete image, thereby extracting exclusively pertinent facial features. Subsequently, the extracted facial data undergoes processing within the face embedder module. To achieve this objective, the following methodologies were employed.

**MTCNN** (Zhang et al., 2016) provides a three stage face detection method which discards via non-max suppression all bounding boxes candidates whose face detection is currently not accurate. It then aligns the resulting face with respect to the vertical axis by means of detecting the bounding box center of each eye and then, after computing the angle between them, it reorients the cropped image accordingly. This model has been chosen for the high confidence,

low false positive cropping result and already implemented alignment at the expense of low computational efficiency. MTCNN has been implemented by the official Python library due to the availability of the pretrained model.

**Yolov8**[1]: is the latest version of the family of Yolo models (Redmond et al., 2016), which had been also explicitly pretrained on face detection. Despite the model does not align faces, it resulted in an easy to deploy approach to be tested. It has been considered for the present work mostly for the promising real-time object detection performance on high frame-per-second videos, allowing both accurate and fast prediction, especially if considering a static image. Originally implemented by converting into a more lightweight tensorflow lite model, we decided to rely on the original Ultralitics torch version due to the availability of an extensive documentation on how to interpret and manage the resulting outputs and to further exploit the multi-processing performance.

**FaceNet** (Schroff et al., 2015): based on a ResNet50 with inception block backbone implementation (GoogleLeNet, Szegedy et al., 2015), this model has been originally trained by minimizing a semi-hard Triplet loss by batching input via online-triplet mining (automatically defining a set of three images *{a, p, n}* given a batch of pictures).

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

For the final submission both the FaceNet and the 512 embedding vector (Facenet512) have been considered to test whether a higher dimensional representation would yield more accurate retrieval results or not.

**ArcFace** (Deng et al., 2019) is a model which takes advantage over the same FaceNet512 architecture (ResNet50 backbone) by proposing a loss function which aims at maximizing the distance between pairs of images belonging to different identities by an angular margin penalty, hence representing the resulting embedding representation as lying onto a N dimensional hyper-sphere (N=512). Identities are then matched according to the closeness expressed by their mutual angular distance which has been proved to accurately segregate and represent identities according to the class they belong to. The model has been trained by the following softmax loss function:

$$L_3=-\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{n}e^{s\cos\theta_j}}.$$

**SFace** (Zhong et al., 2022) also implements a novelty softmax based loss function which aims at projecting the N dimensional face embedding representation onto an hypersphere by jointly considering the minimization of the intra-class distance and the maximization of the inter-class distance (*as below*)

$$r_{intra}\left(\theta_{y_i},\theta_j\right)=\frac{s\sum_{j=1,j\neq y_i}^{C}e^{s\cos\theta_j}}{e^{sf\left(\theta_{y_i}\right)}+\sum_{j=1,j\neq y_i}^{C}e^{s\cos\theta_j}}\frac{\partial f(\theta_{y_i})}{\partial\cos\theta_{y_i}},\qquad r_{inter}\left(\theta_{y_i},\theta_j\right)=\frac{se^{s\cos\theta_j}}{e^{sf\left(\theta_{y_i}\right)}+\sum_{k=1,k\neq y_i}^{C}e^{s\cos\theta_k}}.$$

with the following loss function.

$$L=-[r_{intra}\left(\theta_{y_i},\theta_j\right)]_b\cos\left(\theta_{y_i}\right)+\sum_{j=1,\ j\neq y_i}^{C}[r_{inter}\left(\theta_{y_i},\theta_j\right)]_b\cos\left(\theta_j\right)$$

The novelty here relies on the assumption that the intra-class similarity and the inter-class distance can be addressed separately in order to better tune the minimization of the objective function. The model has been originally developed

[1] Ultralytics. Available at: https://github.com/ultralytics/ultralytics.

upon the same backbone as in ArcFace yielding a 512 dimensional embedding vector and trained over the aforementioned loss.

All models for face representation presented so far have been taken from the LightFace repository (Serengil et al., 2020), downloaded along with the pre-trained weights and deployed as keras objects with the only exception of SFace which has been implemented as a *tflite* model because of the unsolved hindrance in the conversion from the original *onnx* file. With this being said, no customly trained model has been used for the submission of the current project due to the assumption that the current state-of-the art publications presented here already extensively satisfied the need of having a model being trained over face recognition dataset benchmarks such as LFW (Huang et al., 2007), YTF (Wolf et al., 2011), IJB-A (Klare et al. 2017) et similia.

## Methods: inference procedure

The complete pipeline works by initially defining a confidence level for the cropping process. This is usually set in order to allow for a more permissive recognition of faces thus yielding results which may be more likely to be false positives. If no face is found the current image under examination is saved as it is meaning that it can either represent a face whose features are hindering the detection (like if it is partially occluded) or no face is displayed at all. This will allow to have at least a single processed image for those belonging to the query and the gallery and to prevent any false negative (*i.e.* having at least a representation, although not centered and cropped, of a given identity). Before computing the image embedding representation the whole batch of query and test images cropped picture is coerced to be read as a RGB image (retaining the first 3 channels), resized to match the input dimension of the model selected and normalized (*i.e*. rescaled according to the model's requirements). Multiple attempts were made in order to detect the best image normalization procedure for both ArcFace and SFace models due to inconsistencies found over different adaptations[2] *(Table 1)*. The whole matrix of images is then grouped in a single 4D array and loaded into memory in order to be passed as input to the model by batches. Two Feed-forward runs are then computed in order to retrieve the embeddings of the query and gallery set in two distinct matrices by simply running the predict method over the newly generated array of images. Mutual distance metrics is computed by comparing each *i-th* query embedding vector against all the embeddings from the gallery set by extracting the cosine similarity

```
tmp_distance = 1 - query_set[i,:].dot(test_set.T)/np.outer(np.linalg.norm(query_set[i,:]),test_set_norm)
```

and the euclidean distance.

```
tmp_distance = np.linalg.norm(query_set[i,:]-test_set, axis=1)
```

The resulting scores are then placed besides the name of the original image from which the cropping has been executed: this can eventually deal also with some gallery images which present more than a single identity and needs to be retrieved for whichever identity is queried.

## Results

Visual inspection of the Face detection performance on the resulting cropped images has been performed in order to check the accuracy of the faces boundaries. All query images were cropped by setting the detection confidence level at 0.1. The algorithm was found to accurately detect and then save the portion of the picture which was associated with a human face by the highest confidence first and other boxes belonging to candidate faces afterward. This yields no misses or pictures that were saved after having not been detected by the model. Facenet achieved the best

[2] From a discussion over the current LightFace implementation it was argued over which image normalization procedure would have been more suitable for several models.

performance over the whole test dataset (*Table 1*). The following results were achieved by testing each combination between cropping algorithms: the most accurate performance was seen by cropping query and gallery set by Yolov8 and MTCNN respectively. Interestingly ArcFace and SFace resulted in a significant lower performance with respect to the one shown by Facenet (*top1*=0.97, *top5*=0.98, *top10*=0.98).

| Model Name | Image normalization | Top 1 | Top 5 | Top 10 |
|---|---|---|---|---|
| “Facenet” | Standardize across channels | 0.9719 | 0.9813 | 0.9813 |
| “Facenet512” | Standardize across channels | 0.9158 | 0.9532 | 0.9532 |
| “ArcFace” | Rescaling centering to 0, ± 1 (by subtracting 127.5 and dividing by 128) | 0.8411 | 0.9065 | 0.9158 |
| “ArcFace” | Rescaling to [0,1] (dividing by 255) | 0.8317 | 0.9065 | 0.9065 |
| “SFace” | None | 0.5887 | 0.7196 | 0.7476 |
| “SFace” | Rescaling centering to 0, ± 1 (by subtracting 127.5 and dividing by 128) | 0.0373 | 0.1028 | 0.1588 |
| “SFace” | Rescaling to [0,1] (dividing by 255) | 0.0467 | 0.1121 | 0.1495 |

**Table 1:** top 1, top 5 and top 10 retrieval accuracy of the models currently implemented is displayed along with the image normalization implemented. All accuracy were tested by sorting the resulting prediction given a query image by means of cosine similarity.


**Figure 1**: Top 5 identities ordered according to the similarity with respect to the query image. The model correctly retrieved the same identity regardless of the initial face orientation, the filters applied and the partial left occlusion.

## Discussion

The choice of cropping identities’ faces was driven by the assumption that the higher the percentage of non-relevant information excluded the more accurate the embedding representation is, thus yielding more reliable results. Unfortunately we need to state that the face detection has been suboptimally performed due to the lack of an implementation which should have adaptively decreased the confidence level in the case in which no bounding boxes would have been detected. This would have significantly reduced the number of false positives and improved the

whole preprocessing overall therefore preventing any manual intervention. Moreover the current implementation was found to be prone to high computational time requirements and excessive memory usage due to the fact that it was not developed in order to process a higher than ~ 3000 image gallery set and no data loader was implemented due to the need for adapting the input cropped image to the required preprocessing steps (resizing and normalization). The whole query and gallery dataset were loaded into memory to be processed in batch yielding excessive RAM usage: moreover no model exploited any GPU computation in the aforementioned adaptation. Extensive research on how to adapt the underperforming models are needed in order to effectively compare the face retrieval performance with the best performing model.

## Conclusions

The system shed some light on the deep face representation and retrieval performance by depicting identities according to embedding vector estimation from models that were trained based on different theoretical assumptions. This can in principle fulfill the face recognition task allowing to verify the performance of a plethora of state of the art models. The retrieval performance from embedding representation gathered from Facenet was found to achieve the best overall accuracy overcoming the most recent publications despite having been proved to achieve better results. Future works may draw some insights in the evaluation of these latest methodologies in order to address which image normalization may improve the actual retrieval performance.

## References

Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4690-4699).

Huang, G. B., Mattar, M., Berg, T., & Learned-Miller, E. (2008, October). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In *Workshop on faces in Real-Life Images: detection, alignment, and recognition*.

Klare, B. F., Klein, B., Taborsky, E., Blanton, A., Cheney, J., Allen, K., ... & Jain, A. K. (2015). Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1931-1939).

Melekhov, I., Kannala, J., & Rahtu, E. (2016, December). Siamese network features for image matching. In *2016 23rd international conference on pattern recognition (ICPR)* (pp. 378-383). IEEE.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815-823).

Serengil, S. I., & Ozpinar, A. (2020, October). Lightface: A hybrid deep face recognition framework. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)* (pp. 1-5). IEEE.

Wolf, L., Hassner, T., & Maoz, I. (2011, June). Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011* (pp. 529-534). IEEE.

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE signal processing letters*, *23*(10), 1499-1503.

Zhong, Y., Deng, W., Hu, J., Zhao, D., Li, X., & Wen, D. (2021). SFace: Sigmoid-constrained hypersphere loss for robust face recognition. *IEEE Transactions on Image Processing*, *30*, 2587-2598.