Master thesis : One-Shot Learning for Face Recognition

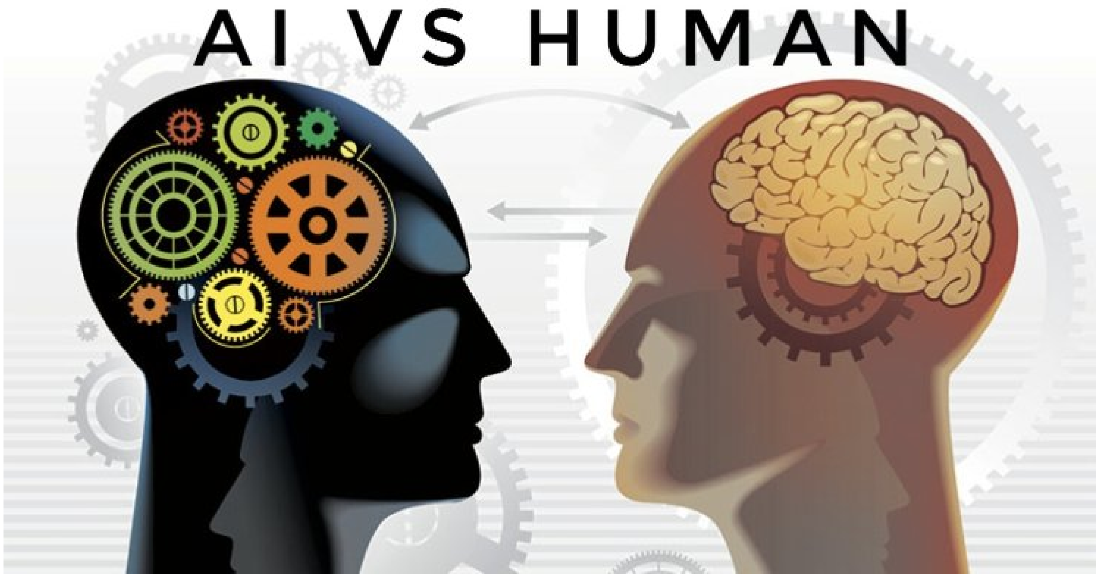
Auteur : Brieven, Géraldine
Promoteur(s) : Louppe, Gilles
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil en informatique, à finalité spécialisée en "intelligent systems"
Année académique : 2018-2019
URI/URL : http://hdl.handle.net/2268.2/6795

University of Liege - Faculty of Applied Sciences

Graduation Studies conducted for obtaining the Master's degree in Computer Engineering

Final Report
One-Shot Learning for Face Recognition

Author: Géraldine Brieven
Promoter: Pr Gilles Louppe



Academic Year 2018-2019

Abstract

This master thesis has been drawn up for the purpose of obtaining the Master grade in Civil Engineering in Computer Science. It relates to a research topic on deep learning focusing in particular to one-shot learning, applied to the face recognition problem. In this work, the face recognition problem is assimilated to the task consisting in identifying a person on a given picture. This classification problem fits quite well with the One-Shot learning setting since it's supported by a training set containing only **few instances of each face**, meaning that the model is expected to quickly integrate new people's face from few data, which represent the challenge carried by one-shot learning. In this work, 4 main phases are defined to perform the face recognition task.

First, databases containing face images have to be acquired. Here, the Labeled Faces in the Wild (LFW), the Celebrities in Frontal-Profile in the Wild (CFP), the FaceScrub cropped and part of the CASIA-WebFace are exploited, in addition to some other extra smaller databases representing common (i.e. not famous) people in order to be closer to real conditions.

From those databases, a training set is being built, made up of less than 10000 face pictures, which represent a quite limited data quantity regarding the standard size of face datasets that sometimes reach hundreds of millions of face images.

Once the data has been collected, the face images are being processed, which mainly consists in detecting the face on each picture, aligning it and finally cropping the picture. Notice that the face detection relies on an external model already trained for.

Next, the focus is on how to efficiently extract features from the face patches and make the in-class [1] closer and out-class [2] further in some embedding space. To do so, a Siamese Network is being trained, based on pairs of faces, to perform well on the verification task[3].

After a good Siamese Network has been trained, it can support the last step - the classification - consisting in assigning an identity to an input face image (probe) by leading some similarity computation between this input picture and identified face images contained in a gallery. Following that, the final predicted identity is the one the input face was the closest during the comparison process.

Besides those 4 main steps, to support the learning process based on few data, 2 extra steps are being defined. First, the Siamese network can be pretrained as encoder belonging to an autoencoder targeting to reproduce input faces. Next, a data augmentation process is defined by employing a Style GAN to derive synthetic face instances.

Finally, regarding the performance of the face recognition task, a top-10 accuracy of 84% can be obtained once a probe has been identified in front of a gallery of 200 people[4]. Besides this, the Siamese Network can reach a f1-score of 87% on the verification task. The autoencoder and the definition of synthetic data can improve the performance only in the case where very little data are initially available (less than 500 face pictures typically).

[1]picture instances representing the same person

[2]picture instances representing different people

[3]consisting in differentiating (resp. assimilating) 2 different (resp. similar) face identities

[4]where each identity is represented by 8 instances

Acknowledgements

First, I would like to thank Gilles Louppe for all the technical advice and supervision all over this work. I'm also very grateful for the warm support of all my friends (Emeline Marechal in particular) and my family!

Then, I thank all my peers who were open to discuss some aspects of my work and finally I want to acknowledge all people having contributed to make available the sufficient computational resources in order to support most of my experiments that couldn't run using my own machine.

Géraldine Brieven

Contents

Chapter 1

Introduction

1.1 Context and Problem Definition

The process of getting a discriminative embedding space as relevant as possible to support some specific task is very common in the machine learning area. Related to that process, one main constraint can be raised referring to the fact that a machine learning based solution often requires a quite large amount of data to be able to generalize appropriately. This last issue is overcome when dealing with the face recognition task in cases where:

- The quantity of training data is limited[1].
- At test time, correct identity predictions are targeted to be made, given only few examples (5-10 typically) of each new class (where one class refers to one identity here).

1.2 Global Solution Summary

To support this task, first, a convolutional network is being tuned, learning to differentiate any face identities at best so that the resulting model is able to generalize to unseen face identities. More precisely, the similarity concept is applied, bringing the model able to infer if 2 faces are more or less similar from a physionomic point of view.

After the tuning step, the model can be used in the task predicting the identity of a new given face picture, based on a gallery namely containing other pictures related to the expected identity. To perform this task, the probe $\mathcal{P}$ is compared to all the available labeled face images and the predicted identity is the one related to the pictures the probe $\mathcal{P}$ is the most similar to.

In addition to this, an autoencoder is being defined to perform unsupervised feature learning and a data augmentation process based on synthetic data is being designed.

The whole implementation of this solution is available through [2].

1.3 General Structure

Now a very global overview has been exposed, let's consider how this work is structured to detail the whole solution process in response to the face recognition task, using one-shot learning.

[1]which can be assimilated to less than 1000 people, each having about 10 corresponding face pictures typically

First, the problem to tackle over this work is formally defined and then specific objectives around that problem are enumerated through Chapters 2 and 3.

Following that, Chapter 4 exposes the state of the art of both one-shot learning and the face recognition application, focused on the techniques that are exploited when designing the effective solution. Besides those 2 points, some words are also dedicated to the procedure implementing a particular style GAN, employed to generate synthetic data in order to augment the initial dataset.

Next, the way the solution was designed is being exposed and the main implementation choices are justified. More precisely, over that fifth Chapter, first, the quantity and the nature of the data supporting all the learning and the evaluating processes is being developed so that the reader can measure the degree of difficulty the final algorithm can deal with. Then, the data processing phase is clarified, followed by the data augmentation process relying on synthetic data creation. After that, the Siamese Network training (supporting the projection of the input face image to a final discriminative embedding space) is being explained, relating about the network architecture, the various possible losses to direct at best the learning process and the hyperparameters setting. In addition to this, an optional pretraining phase is being defined as well, by using an autoencoder, to get familiar to the human face concept before tackling the more difficult task consisting in differentiating face identities. Moreover, some extra strategies to improve the current quality of the feature representation are being proposed. Once the Siamese Network has been defined, the algorithm exploiting it to perform the face recognition task is being described. Finally, for a comparison purpose, the classical classification approach is being discussed.

Once the whole solution framework has been deeply developed, the experimentation phase, investigating a very large number of possible scenarios, is being discussed. Each scenario is characterized by a specific parameters and module usage combination. In particular, 2 phases are being evaluated. The first one refers to the Siamese Network, evaluated on the verification task. The second one exploits the Siamese Network to identify faces by comparing an input face picture to labeled pictures stored in a gallery.

After this, before drawing to the conclusion, a retrospective showing both the strong and the weaker points of that work is provided, also exposing some other interesting tracks that couldn't be explored here but would nevertheless be worth examining.

Chapter 2

Definition of the problem

Before providing a review of the literature related to both few-shot learning and face recognition, let's formalize the face recognition problem that is tackled here. Similarly to classical classification problems, a learning set $\mathcal{LS}$ and a testing set $\mathcal{TS}$ (with no overlapping) are defined. They are both composed of labeled samples $(\mathcal{P}, i)$ where $\mathcal{P}$ represents a face image (that can be called probe here) and i the corresponding person/set of people on the image. In this work, only one person states on a given image and the objective consists in identifying that person.

From this point, the complexity of the task can be increased by playing on the size of the training set (decreasing the number of instances related to each person i and/or the number of people), namely. Besides this, the performance can be also highly influenced by the external features related to the pictures composing the testing set. Indeed, it's important to realize that, in the context of a face characterization, 2 kinds of features can be derived:

- The **high-level (external) features** referring to some external characteristics (like the background, the lightness, the hairstyle, extra accessories like glasses...), the expression of the face and the orientation.
- The **physiognomic features** allowing to distinguish one person from another through the shape of the face, the noise, the mouth, the eyes ...

In the face recognition task, the focus has to be only on the physiognomic features. From that point, a challenge is extracting those last features and making the impact of the external ones as "transparent" as possible over the face recognition task. Knowing this, it can be guessed that the more variable and rich the external features in the testing set, the more difficult it can be for a model to recognize a person since it has to deal with the 2 kinds of features, and finally isolate and consider only the physiognomic ones.

Lastly, notice that the complexity of the task is also influenced by the similarity between the faces belonging to the same dataset. The more distinct the faces, the easier the recognition.

Chapter 3

Definition of the objective

The general aim of this work is defining a system able to identify a single frontal[1] face having variable external features (hairstyle, face expressions...) and taken in variable conditions (i.e. different background, light, size, orientation ...), based on some other identified pictures (where the quantity varies from 4 to 10 per identity typically).

To do so, an iterative solution is proposed. At each "work iteration", the current solution is tried to be improved in terms of performance (i.e. accuracy on the face recognition task) and data quantity support (where this quantity is reduced). To do so, new modules are defined, each directly referring to the concepts which have been exposed in the state of the art. The procedure which is planned to be followed is detailed below:

1. Implementation of a system performing well on the face recognition task[2], when evaluated on a dataset sharing people in common with the training set. More precisely, to support the face recognition task, a **matching algorithm** is planned to be designed, relying on similarities between feature representations, rather than explicitly implementing a classifier. To support it, a **Siamese Network** is defined, in order to get a feature representation based on the reasoning that similar (resp. different) persons' faces should have a very similar (resp. different) feature representation, whatever the external characteristics. Finally, the quality of the feature representation can be evaluated through face recognition task relying on that.

2. Implementation of system based on a dataset $\mathcal{D}_i$ and used to perform well on another dataset $\mathcal{D}_j$ (such that $\mathcal{D}_i$ and $\mathcal{D}_j$ don't share any common people on the images they contain). Notice that it's related to the meta-learning concept discussed in Section 4.1.2, consisting in learning to **recognize faces in general**, independently from any specific instances.

3. Improvement of the current system that has to be able to perform well from few data[3]. To do so:

   (a) a **data augmentation** process is planned to be designed, based on the synthetic data generation.

   (b) a **pretraining phase** is planned to be defined where the Siamese Network is trained on another more simple task in order to learn faster when having to learn to perform well on the verification task.

[1] where the face can be just a little turned, as shown in Figure 5.2

[2] "good performance" referring to about 80% of top-5 accuracy, based on a gallery of 100 people

[3] where 8000 face pictures are exploited at training time typically

Chapter 4

Background

4.1 One-shot Learning

In many applications requiring Artificial Intelligence, the Neural Network architecture, inspired from human's brain, is getting more and more famous because of it's ability to perform complex tasks that only human could solve so far.

Basically, the Neural Network concept relies on the fact that learning results from getting experience. In the context of a Deep Neural Network, this experience is represented by the large quantity of examples which are used to train it repeatedly. The more general the training set, the more accurate the resulting trained Neural Network in making prediction from unknown input objects. However, related to that concept, some current weaknesses can be highlighted.

First, several existing machine learning algorithms learn "without understanding" what's true or false, real or imaginary, fair or unfair, only targeting to minimize a loss function. This point makes more difficult the improvement of the technique because of this black-box aspect of the model, where it's difficult to derive the reasoning leading the prediction task... The interpretability of Neural Networks is still an open subject!

Next, the Neural Network training is very costly. From a mathematical point of view, it's like learning a very complex function depending on a huge number of parameters that has to be optimized from scratch, according to a very large quantity of training examples. Some applications are even not achievable because they require rapid inference from only small quantities of data.

Finally, deep networks are usually extremely data hungry while data may be scarce and the data acquisition may appear as being very tedious, especially in the case where a great quantity of data has to be manually labeled (or tag).

Seeing those weaknesses, the few-shot learning concept raised.[1] Fundamentally, it consists in learning from few examples, using few iterations, like humans are able to learn fast from few examples and repetitions. Indeed, for instance, in the task of recognizing a banana depicted on an image, a human doesn't need to successively see like 5000 images of banana, while it's required for a common algorithm using a neural network in image recognition. The few-shot learning concept is then very challenging! It requires to approach a task differently.

To define this new approach, let's consider the different concepts which can be used:

[1]In this report, both one and few shot learning concepts will be employed in a similar way, pointing the fact of using a limited quantity of data. Strictly, one-shot learning refers to learn from one instance while few-shot learning requires few instances.

- The **Data Augmentation** consisting in generating more data in order to get a larger (and more general) learning set corresponding to the simplest approach. To do so, the simplest approach is slightly modifying the original examples. It's a good technique in the sense that, in practice, the world never offers perfect instances, there's often noise or slight differences compared to the theoretical view. A more sophisticate process would be to generate synthetic images supported by another neural network trained for.
- The **Deep Embedding Learning and Matching Network** consisting in first representing the input data in a low-dimensional embedding space and then exploiting the representation.
- The **Meta-learning** allowing to reach rapid learning from sparse data. To perform this, some experience which has been acquired from other related tasks is exploited. Many methods relying on that concept are developed but not all of them are supporting the final solution.

Of course, those concepts can be combined together, making the final face recognition algorithm even closer to the human's behavior during the learning process.

Besides this, other techniques are also relevant to mention, despite they aren't exploited in the final solution that is provided at the end of this work:

- The **Memory Exploitation** where information is stored and used. The information can model a state or figure some intermediate representations derived from old data. This technique isn't detailed here since it's exploited in the solution design.
- The **Hierarchical Classification and multi label classification** where intermediate predictions are implemented in order to get a final one. This approach is quite intuitive and is easy to interpret. It basically consists in decomposing the problem into simpler ones.
- The **Active Learning** belonging to the class of the reinforcement learning algorithms. The idea is defining an agent dealing with the examples provided as input, in order to train the model. Even if it's not included in the final solution, it's developed a bit more in Subsection 4.1.5.

4.1.1 Data Augmentation

To perform some data augmentation, many approaches can be selected.

A first basic way to generate more image instances is by **rotating the initial image, changing the colors or cropping the image**, basically. Notice that this type of data augmentation may appear as more or less relevant, depending on the data processing phase that is defined. In the case where it namely consists in aligning the image to some reference points, generating duplicates by rotating the image would be useless.

A second more sophisticate approach consists in using a **generator** in order to produce synthetic different images. The advantage of this second approach is that the training of such a neural network doesn't require labeled data, which is much more manageable for some application like face recognition or text classification where the labelling process may be quite tedious while just finding unlabeled instances is easier.

The data augmentation method is developed in section 4.2 in the particular case of the face recognition problem representing the target of all this work.

4.1.2 Meta-Learning Exploitation

As humans, when we are learning a new task like word or object recognition, we usually have already some experience in such kind of task, making this learning process easier and faster. For instance, when we have to learn some new characters, we have already in mind the notion of straight line, circle... allowing us to characterize faster what we have to integrate as new knowledge. In the Neural Network settings, this concept is implemented through a "smart initialization of the parameters" of the network supporting the knowledge. To do so, several approaches can be considered.

First, in the literature, meta-learning models is most of the time defined as referring to a two-levels definition of the learning problem:

- the rapid learning where knowledge is acquired within each task
- the gradual learning extracting knowledge learned across all tasks

Those 2 types of learning are captured by a *meta-learner* responsible for the parameter initialization (and updating). For classification problems, it's trained in order to optimize the solution given by the *learner classifier*, on each task. More specifically, the training phase is led by **successively considering episodes**, each episode being represented by a pair of training and testing sets $(\mathcal{D}_{train}, \mathcal{D}_{test})$. At each iteration, the aim is minimizing the error on the current testing set by using the current training set and the past experience acquired from the previous episodes.[2] As final step, the aim is performing at best in a new task where new learning and testing sets are given. This procedure is illustrated in Figure 4.1.

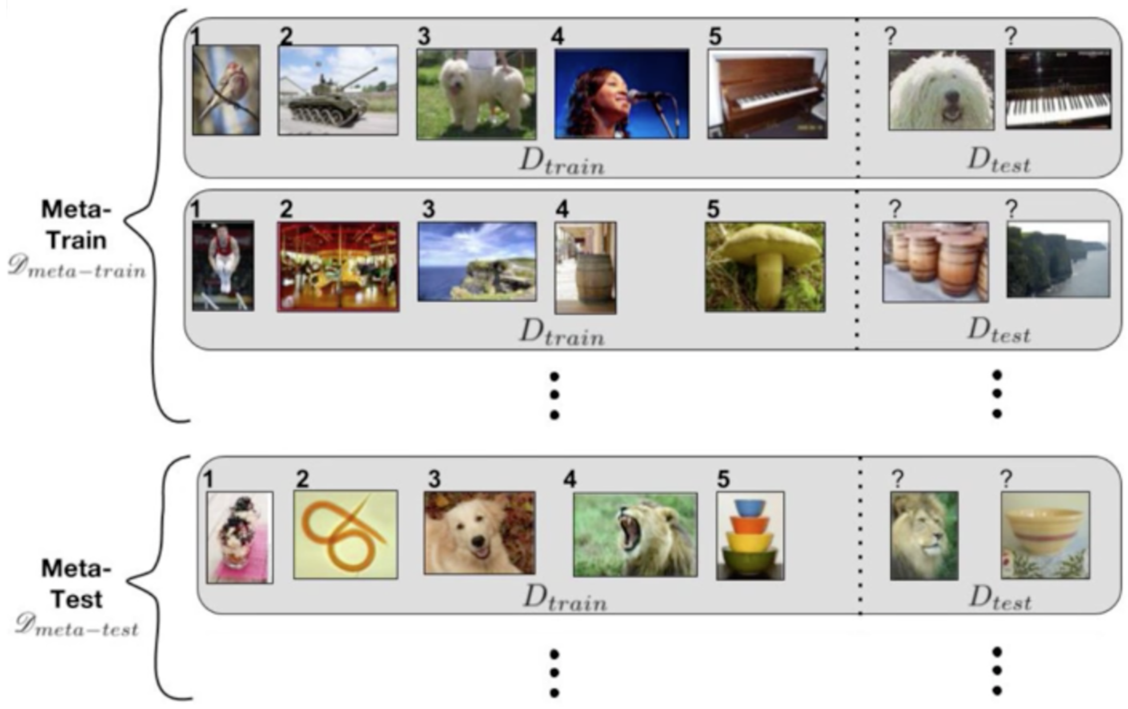


Figure 4.1: Meta-learning Setup

[2]The order of the episodes impacts on the meta-learner which is derived. Then it would be relevant to build different meta-learners (by just changing the order of the episodes) and averaging their prediction at test time.

Besides this, another approach consists in defining a model which is as general as possible, by making it **very adaptable** [3]. Here, contrary to the first approach, there's no learned update: the parameters related to the model are updated using the gradient-based method. The idea is training this model based on a certain set of learning tasks such that it will be able to manage later new tasks quickly and from few experience, by combining its prior knowledge and the small amount of new information.

From a mathematical point of view, it consists in maximizing the sensitivity of the loss function with respect to the parameters. By doing so, the local variations of the parameters may lead to significant improvement of the loss task [2]. One of the advantages of this concept compared to the others is that it can be applied to a large range of problems like classification or reinforcement learning.

Let's now formalize that last approach by modeling the meta-learning problem and the algorithm solving it. As usual, the goal is defining a model f_θ mapping some instances $\mathbf{x}$ to outputs $\mathbf{y}$. In this perspective where it's expected to be as general as possible, a generic notion of a learning task is introduced:

$$\mathcal{T} = \{\mathcal{L}(\mathbf{x}_1, \mathbf{y}_1, ..., \mathbf{x}_H, \mathbf{y}_H), q(\mathbf{x}_1), q(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{y}_t), H\}$$

where:

- $\mathcal{L}$ refers to the loss function
- $q(\mathbf{x}_1)$ represents the distribution over initial observation
- $q(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{y}_t)$ is the transition distribution
- H is the length of an episode[3]

In a meta-learning scenario, we're interested in a distribution over tasks $p(\mathcal{T})$ that we want to model in order to easily generalize over different tasks.

Then, in k-shot learning setting, the model is trained in order to learn a new task $\mathcal{T}_i$ derived from the task distribution, where the learning set contains k samples drawn from the distribution q_i and the loss function $\mathcal{L}_i$ to optimize is generated by $\mathcal{T}_i$.

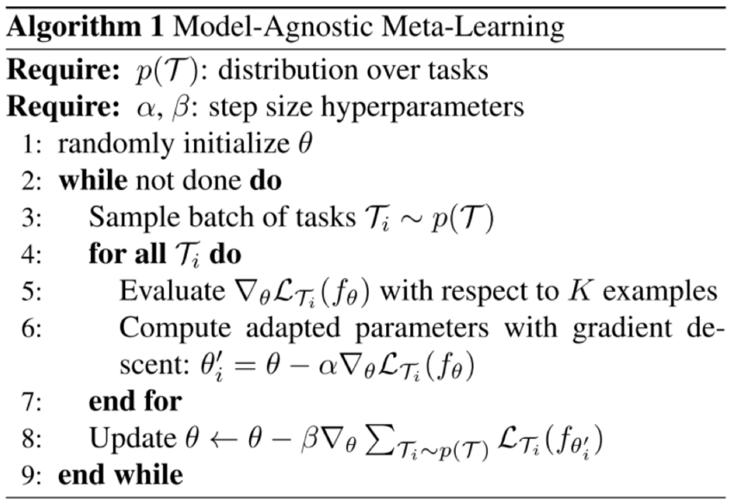
After training, when the model f_θ has to be adapted to a new task $\mathcal{T}_i$, the parameters θ become θ'_i. The update of the parameter is given by:

$$\theta' \leftarrow \theta - \alpha \Delta \mathcal{L}_{\mathcal{T}_i}(f_\theta) \tag{4.1}$$

where α is the learning rate, a hyperparameter specifying "how much we want to learn" from the new task.

Equation 4.1 simply illustrates that the new parameter values are given by the parameters θ (evaluated during the training phase) refined according to the corresponding loss function, evaluated on the new task i, in order to minimize it. Then, it's assumed that the loss function is smooth enough in θ so that the gradient-based method can be applied. The corresponding algorithm is given in Figure 4.2.

[3]For a classification problem, $H = 1$. The episode notion is relevant in the context of reinforcement learning.



Algorithm 1 Model-Agnostic Meta-Learning

```
Require: $p(\mathcal{T})$: distribution over tasks
Require: $\alpha$, $\beta$: step size hyperparameters
 1: randomly initialize $\theta$
 2: while not done do
 3:     Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
 4:     for all $\mathcal{T}_i$ do
 5:         Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to $K$ examples
 6:         Compute adapted parameters with gradient de-
            scent: $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
 7:     end for
 8:     Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$
 9: end while
```

Figure 4.2: Algorithm:Model-Agnostic Meta-Learning

In the same way, we have **transfer learning** relying on a pre-training phase on other similar task (where more data are available). This task supporting the pre-training phase is called the *source task*. Then, the values of the parameters are evaluated and can be used as the initial ones related to another similar task called the *transfer task*. Thanks to a smart initialization, the parameters are not expected to change much, making the learning phase faster and potentially the "input understanding" better. However, the performance of this approach are limited to *transfer tasks* which are sufficiently similar to the *source task*. Moreover, the pretraining conditions (i.e. on the *source task*) don't meet the inference at test time where only few instances are available, making less general the learned deep neural network architecture from prior knowledge. To overcome this lack of generalization, the matching network can be introduced. It's described in Section 4.1.3.

Finally, another way to deal with few labeled examples consists in **exploiting unlabeled data**[4] as additional experience [4]. Again, we can think about humans. When we are learning, we get some "initial knowledge" (that can be assimilated as the one derived from the labeled data) and then we are able to make links between this initial knowledge and new similar coming instances (which can be viewed as the unlabeled data).

In this setting, 2 scenarios can be imagined. A first (and more simple) one, where all the unlabeled data are supposed to be related to the same class as the given labeled data. And a second (and more realistic) one where some "unknown" examples occurred (i.e. belonging to a distractor class).

Let's now formalize the problem. The training phase involves iterations over episodes. Each episode is made up of:

- A Support Set $\mathcal{S}$ (composed of labeled data)
- An Unlabeled Set $\mathcal{R}$
- A Query Set $\mathcal{Q}$ (whose instances are aimed to be correctly labeled)

which is a replication of the testing conditions, as illustrated in Figure 4.3.

[4]assuming that unlabeled data is available.

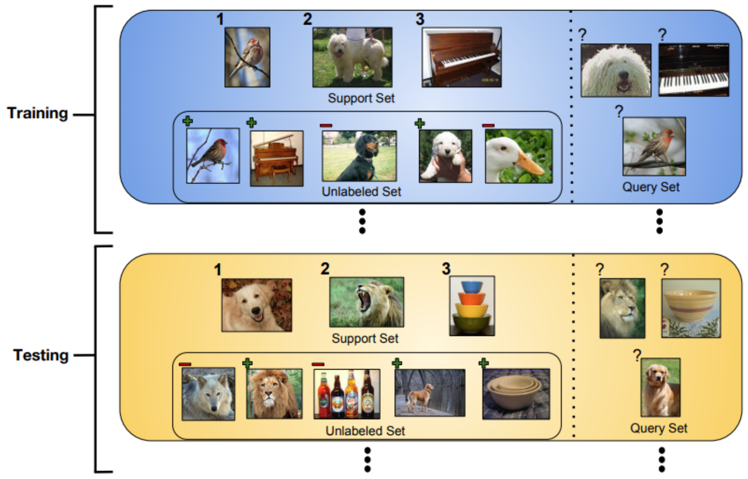


Figure 4.3: Semi-Supervised Few-Shot Learning Setup

During the training phase, a prototypical network is trained in order to derive the best feature representation from any input. More precisely, it consists in learning an embedding function parametrized as a neural network. This concept is described later, in Section 4.1.4. To do so, first the labeled data are exploited, followed by the unlabeled set $\mathcal{R}$, used in order to provide a refined prototype, since it provides in so way additional experience to complete the current one.

4.1.3 Matching Network

Related to what is described above, let's develop now the Matching Network class[5]. A Matching Network relies on good features representations. In classification problems, once the input is expressed in this new space, a metric considering the deep neural features is applied in order to match the labels at best with their feature representation. This procedure is illustrated through Figure 4.4 where g_θ and f_θ represent the feature extractors[5] whose outputs are compared based on a similarity function in order to get the best matching and derive the label of the test example. This approach is very interesting because it's partially non-parametric, allowing the model to include more examples faster. During the training phase, it's targeted to match the test and train conditions. In other words, here in the context of one-shot learning, it means that the training phase is supported by few instances per classes (composing a support set S), moving the task from one minibatch to another, similarly to the testing phase where few examples are available for a new task.

A typical task that Matching Networks could deal with is image recognition where a small labeled support set is defined and one unlabeled example is aimed to be labeled. To perform this task, a classifier $c_S(\hat{x})$ has to be derived from the support set S. Then, a mapping is defined as followed:

$$S \rightarrow c_S(\hat{x}) \quad \text{expressing} \quad P(\hat{y}|\hat{x}, S)$$

where $S = \{(x_i, y_i)\}_{i=1}^{k}$ and P is parameterised by a neural network and represents the conditional probability of labelling the test example $\hat{x}$ with label $\hat{y}$, taking into account the support set.

[5]and $g = f$, potentially

Then, once a new support set S' is provided in order to lead the labelling of a new test example, we simply use the parametric neural network defined by P to predict from $\hat{x}$ the correct label $\hat{y}$ according to:

$$argmax_y P(\hat{y'}|\hat{x'}, S)$$

Concerning the model itself, the output is represented as a linear combination of the labels in the support set:

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i \tag{4.2}$$

where a is an attention mechanism represented by a kernel on $X \times X$. This attention mechanism fully specifies the classifier and the cosine distance function, typically, in order to measure the matching between its input.

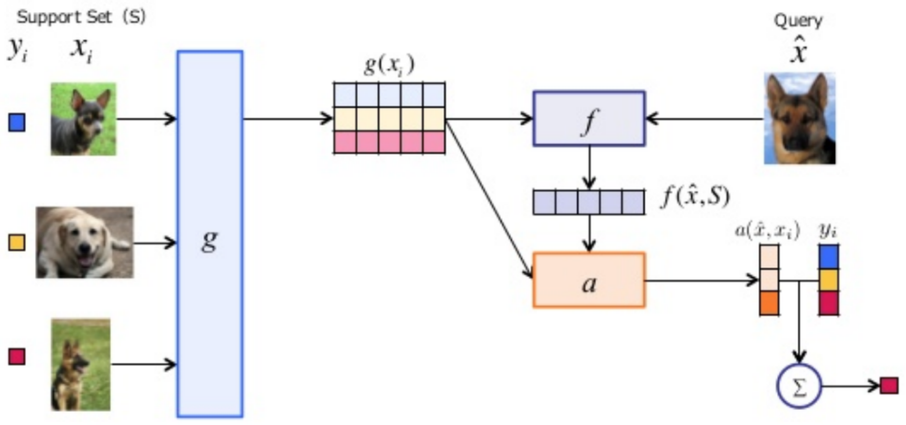


Figure 4.4: Matching Network

The model that is obtained can also be viewed as a function bounding some particular examples of the support set to a given input $\hat{x}$, in order to finally retrieve the associated label of the given input.

This algorithm can be extended to the Memory Matching Network where a memory module is added, allowing to compress and generalize the instances of S under slots in memory, leading to a better generalization and then higher performance [6]. Once the memory slots are derived, they are provided as inputs to a RNN, playing the role of *contextual learning*. It aims to predict the parameters of the CNN dedicated to the test instances[6]. By doing so, the *contextual learner* ensures long-term memory over all the classes occurring in the learning set and short-memory dedicated to the knowledge related to the classes at test time. Moreover, here, it's important to underline that parameters are computed on the fly.

4.1.4 Deep Embedding Learning

Let's focus now on the Deep Embedding Learning setting supporting a classification task. This setting consisting in creating a low-dimensional space to represent the input data, such that the resulting feature representation is more discriminative, making easier the classification, as typically developed in [7].

[6]This concept of learning a learner whose goal is inferring parameters is typical of meta-learning, again.

To do so, several procedures can be led by implementing a loss function according to. A first reasoning consists in encouraging the representation of examples belonging to the same class to be as close as possible to each others. In the same way, a lower-bound can be imposed on the distance between the representation of instances related to different classes in order to differentiate them at best and then making easier the class prediction from the feature representation. Those conditions are expressed through the *contrastive loss*, basically optimizing the differentiation between the representations of 2 different subjects. A more simple version of this loss function is the *margin-based loss*, only targeting to minimize the distance between feature representation of instances of the same class. Finally, another very interesting loss function following the same reasoning is the *triplet loss*. It considers 3 images:

- an anchor instance A (considered as the reference)
- a positive instance P (belonging to the same class of instance A)
- a negative instance N (belonging to a class different from A)

From this setting, the idea is training a feature embedding making closer instance A to the instance P compared to instance N. It can be expressed through:

$$\mathcal{L}(A,P,N) = max(0, ||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha) \tag{4.3}$$

where α represents the margin and $f(.)$ is the embedding function. Here the Euclidian distance metric was selected.

This reasoning is illustrated through Figure 4.5.

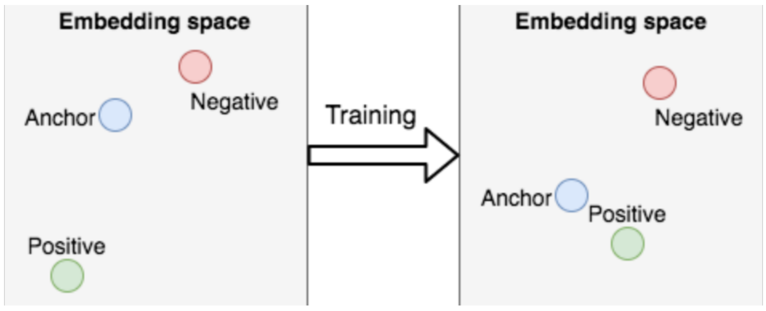


Figure 4.5: Triplet Loss Reasoning

At this stage, it's also important to raise the influence of sampling! Indeed, for a fixed loss function, it was observed that different accuracies were computed, depending on the way the training set was built. More specifically, 3 sampling methods can be employed:

- **Random Approach** where the samples are simply selected randomly.
- **Semi-Hard Approach** consisting in imposing: $||f(A) - f(N)||^2 < ||f(A) - f(P)||^2$ in order to make "harder" the training phase (and then increasing the computation efficiency). In that case, it's assumed that some embedding function f has already been evaluated.
- **Distance-Weight Approach** selecting more informative and stable examples by constraining the distance between pairs (A, N) to follow a specific distribution.

An overview of deep embedding learning is provided through Figure 4.6. First, the batches supporting learning are built, following a specific sampling strategy. Then the images are represented through some features. Finally, a loss function measures the quality of the embeddings according to some criteria depending on the loss function itself.

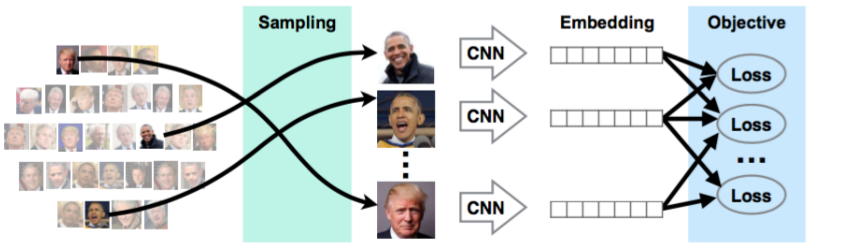


Figure 4.6: Deep Embedding Learning

Finally, once evaluated, in general, the deep Convolutional Neural Network has demonstrated high performance in feature representation, especially for image recognition. However, high performance achievement requires a quite large quantity of annotated data in order to define a relevant deep feature representation generalizing well... That's why here, in the context of one-shot learning, the `CNN` architecture is used, coupled with other modules stimulating the learning phase or a specific reasoning.

Siamese Network

Related to the global approach described above, the Siamese Network is developed. This network targets to learn a deep relevant feature representation by relying on a similarity function [8] [9], based on pairs of images. Indeed, instead of directly learning to classify its input, the neural network learns to differentiate two inputs related to different classes. It's illustrated through Figure 4.7. First, the input images are given to a Convolutional Neural Network (sharing the same parameters values) transforming them into their corresponding feature representation. After this, the feature representations of both inputs are compared through a distance function measuring how similar they are. During training, the objective function is minimized when the distance between the representations of similar (resp. different) objects is small (resp. large). After training, the resulting Convolutional Neural Network can be used for classification, supported by a Matching algorithm for instance.

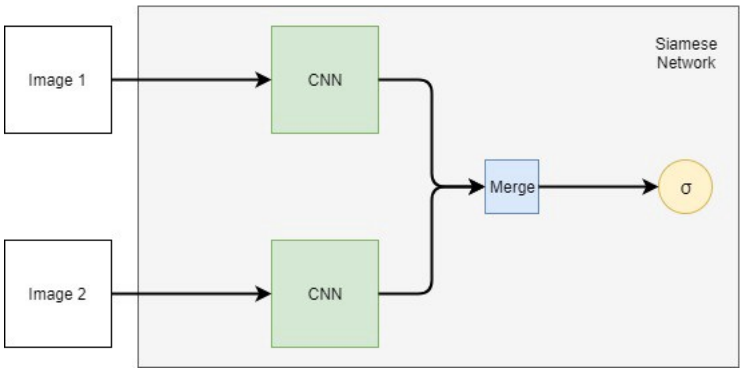


Figure 4.7: Siamese Network

One recent paper [10] extended the use of the Siamese Network by combining it with the attention mechanism, which demonstrated quite high performance. Again, contrary to the basic approach, there's no conditioning on class identities (i.e. no explicit label which is defined). The algorithm just relies on similarities between objects' parts in a weakly supervised manner, using both Siamese similarity and attention mechanism. This technique allows to reach better generalization.

More specifically, the algorithm takes one exemplar x representing an instance of a class A[7] and one target example T including or not an instance of class A. Moreover, the target example is supposed to be larger such that it contains different target locations l. From this setting, a model is defined through a similarity function $s_l(x, T)$ computed in a specific location l in the target example T. This similarity function is as large as the example x is similar to the content at locations l of T.

This approach is particularly interesting in the context of face recognition where no labelling is required and where a face may have first to be detected on the picture, before being identified.

4.1.5 Active Learning

Finally, let's consider a last concept related to few-shot learning. As briefly mentioned above, machine learning applications design may require lots of human resources in the task consisting in collecting and labelling the instances supporting the learning phase. To alleviate this weakness, active learning can be implemented. It consists in defining an intelligent agent managing the collected examples provided as input, in order to train the model [11][12]. These examples are treated online. For each of them, the agent has to decide to predict the corresponding label or to request the true label of the example, if it considers that the uncertainty is too large. The associated reward is maximal for correct labeling, minimal for incorrect labeling and intermediate for labels that have been requested. By doing so, it's targeted to learn from the least possible instances, and maximizing the accuracy at the same time. This illustrates the Few-Shot Learning concept.

4.2 Face Recognition Problem [1]

In this One-Shot learning setting, the focus is on **deep** face recognition. Basically, it refers to learn a hierarchical architecture of features discriminating at best the faces that have to be differentiated.

Nowadays, face recognition has become one of the most researched topics in computer vision and biometrics. Here, the Face Recognition task is tackled by exploiting deep neural network abilities.

Face recognition may refer to 2 categories of problem. First, the goal can be just verifying that a specific face represents a given person. In this case, a one-to-one similarity is computed between the gallery (containing instances related to a specific identity) and the probe (i.e. the picture that has to be checked). In the other hand, the aim can be identifying a person on a picture, based on a gallery containing labeled pictures of several people. In this last case, the identity is derived after computing some one-to-many similarities. A more complete view on the different possible scenarios included in the FR problem class is exposed through Figure 4.8.

[7]which is not explicitly defined in the algorithm

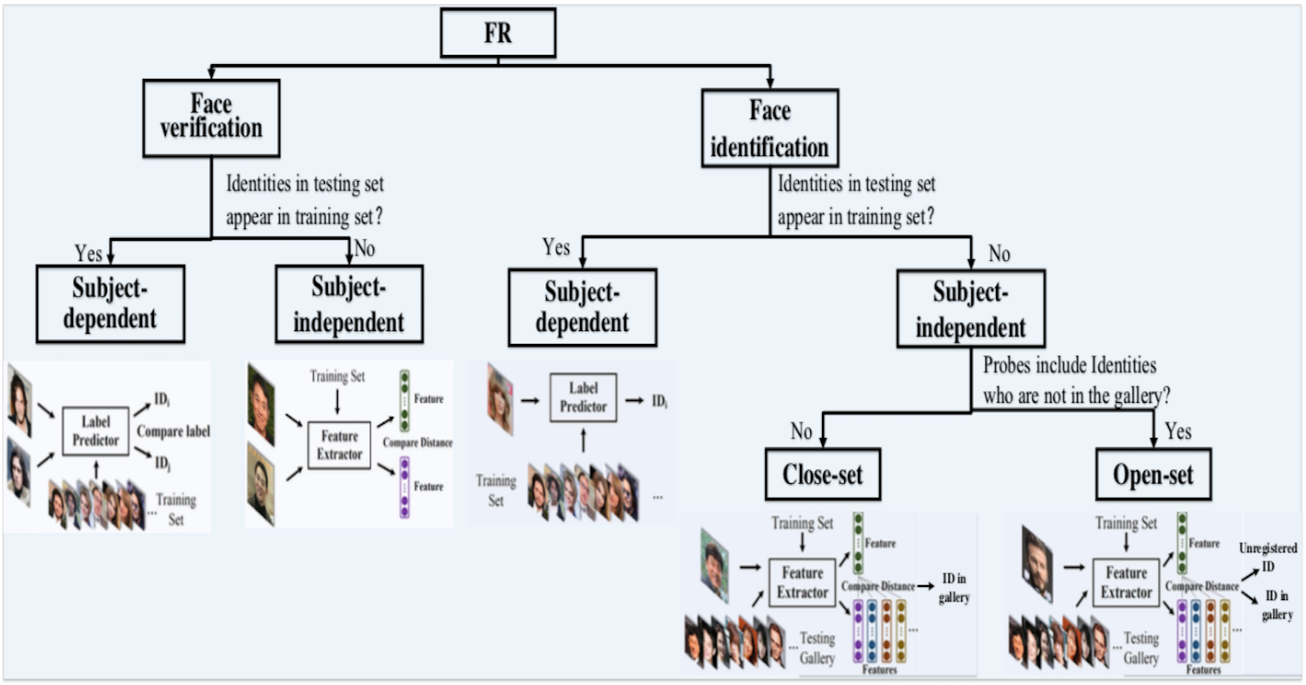


Figure 4.8: Different Training Protocols

4.2.1 Terminology

First and foremost, let's define some specific terms employed all over this report.

- **The Face Recognition task** here refers to the identification of a person on a given picture. It doesn't include the face detection process, meaning that all the given pictures have to illustrate one and only one face. Over all this work, both the face recognition and the face identification processes refer to the same task.
- **The Face Verification Task** consists in predicting from 2 face pictures if they represent the same person or not. This task is performed by the Siamese Network and can be used as sub-task to design a face recognition system.
- **A gallery** $\mathcal{G}$ represents a set of labeled face pictures $\mathcal{G}_{ij}$, where i and j respectively refer to person i and the index of the pictures related to person i.
- **A probe** $\mathcal{P}$ represents an unlabeled face picture that has to be identified by being compared to the pictures contained in the gallery.
- **The matching phase**, also called classification phase, refers to the process where the probe is compared to each instance contained in the gallery to be finally associated to the most similar ones whose identities are known.
- **The external (i.e. view) features** refer to the features that can be derived from a face picture but that don't characterize the identity of the face, in opposition to the physionomic feature.
- **A synthetic image** $\mathcal{I}_{synt}$ is a face image created by a generator. It can be either a variant of a given face picture or a synthetic person.

- **The disturbance** refers here to the distance between the feature representations of 2 instances of the same person.
- **Siamese Network** $\mathcal{S}$ is a specific kind of network that takes as input pairs of face images and compute for each of them comparable output vectors derived after propagating each picture through the same network.

Let's now get an overview on the different components of face recognition system.

4.2.2 Face Processing

As specified before, in this work, the reasoning consists in deriving a high level abstraction representing the facial identity, in a stable way, such that the features that are derived are **invariant to face pose, light and expression changes**. Focusing on the poses, 2 kinds of image processing can be mentioned:

- **One-to-many Augmentation** where, from a single image, many pictures are generated. Notice that this processing is very relevant in the context of one-shot learning where few instances are available.

  To do so, first, some simple transformation can be applied, like mirroring or rotating the image.

  Second, some 3D-face reconstruction can be performed in order to compensate for low resolution, poor contrast and non-frontal pose.

  Third, to avoid building 3D representation and then projecting the image back to a 2D domain, some CNN can be directly used in order to synthesize images with different poses. Here, 2 types of neurons can be defined: the deterministic hidden ones, learning the identity features, and the random hidden ones capturing the view features.

  Lastly, some GAN can also be exploited in order to get synthesized images. This last approach is developed in a deeper way in the following section since it's investigated and exploited in this work.
- **Many-to-one Augmentation** where, on the contrary, from several images representing a same person, one single view is generated, so that it's more representative of the person, in some way, because it's not related to any specific "condition" (like the luminosity, the pose ...). In terms of implementation, a stacked autoencoder can be used, mapping a set of non-frontal faces to a single frontal view. Moreover, as before, a CNN or a GAN can be exploited to reconstruct a single representation of the person.

4.2.3 Deep Feature Extraction

Following the processing face, a very important and challenging step is the deep feature extraction, providing the feature representation of a given image that supports after the face recognition task itself. To lead this phase, a neural network (with some specific structure) is trained under a defined loss function that direct the variation of the parameters of the network.

First, about the network architecture, Convolution Neural Network (i.e. CNN) was revealed as being the most popular in the area of image recognition, because if its ability to extract relevant features from images. Some famous CNN architectures are the AlexNet, the VGGNet the GoogleNet and the ResNet. Some architecture variants were also designed, relying on the definition of a max-feature-map activation function (to reduce the computational cost) or combining the output of 2 CNNs to obtain bilinear representations.

To go further, some joint alignment-representation networks were also experimented. They refer to an end-to-end system made up of different modules, each of them optimizing the performance of the the main face recognition problem. The different modules can be responsible for face alignment, face representation or face recognition itself.

Following the same multi-modules reasoning, multiple networks solutions were also designed and could bring some improvement to the basic backbone network approach. In this context, 2 kinds of networks can be defined:

- **Multi-Input Networks** leading one-to-many augmentation.
- **Multi-Task Learning Networks** working on some side tasks for identity classification like pose, illumination or expression estimation, but also face detection, gender estimation ...

Next to this, considering the loss functions, many ways can be explored. A very common one relies on the *softmax loss* whose objective is maximizing the probability attached to correct class. However, for the face recognition problem, this simple approach has limited performance in feature derivation. As alternative, other losses are are exploited, all of them explicitly optimizing the difference in feature representations corresponding to different identities. Here are those different losses:

- The ***Euclidian-distance-based loss*** both minimizing the intra-variance and maximizing the inter-variance related to the classes. More precisely, the loss functions that are defining in this setting are the *contrastive loss*, the *triplet loss* (explained before in the context of the Siamese Network) and the *center loss*. Notice that this function overcome the instability problem that may occur with the 2 first functions.
- The ***Angular/Cosine-margin-based loss*** in the same way enforcing similarity (resp. dissimilarity) between feature representation of same (resp. different) labeled images. More specifically, in this setting, the *L-softmax* and the *A-softmax* can be implemented, both inspired from the *softmax loss* but including also the (dis)similarity optimization concept.
- Some improved versions of the *softmax loss* mainly consisting in normalizing features (or weights) or enforcing the norm of samples to be some specific value R. This last method refers to the ***Ring Loss***.

An illustration of some of the losses mentioned above is shown through Figure 4.9.

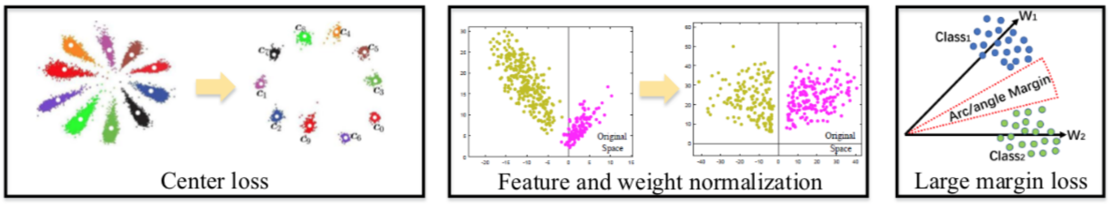


Figure 4.9: Center Loss, Feature and Weight Normalization and Large Margin Loss

4.2.4 Face Matching

Once an optimized feature representation is derived from a given input image, it can be exploited in order to perform:

- **Face Verification** relying on a metric that has to make 2 different classes as separable as possible. This task can be typically performed based on a Siamese Network by additionally defining a threshold. This threshold is compared to the distance between the embeddings of the 2 input images so that it can finally output a boolean value telling if the 2 input faces represent the same identity or not.
- **Face Identification** relying on a voting system, where the probe is compared to each picture of the gallery and some similarity score is outputted from each comparison. In this setting, it's important to mention that in practice, most of the time, it's not realistic to get a training set containing images of all the people the model will be used for as recognizer, meaning that some transfer learning is "naturally" introduced to deal with the face recognition task in a practical context. Some faces (belonging to the source domain) are used to train a model so that it can represent any particular faces at best. Then faces of new people (belonging to the target domain) are given as input to the model that derive a corresponding feature representation.

4.2.5 Data Resources

The samples composing the database support the learning process of the face recognition model, meaning that the choice of those samples is very important. Those samples have to be as much representative as possible to reduce at best sample bias (due to over-represented environments, specific expressions, specific poses...) In practice, acquiring face databases isn't an easy task. Most of the available ones contain pictures of celebrities (because of privacy issues), which isn't so representative of all the faces a FR system could encounter in the future...

Moreover, as specified before, the "conditions" of the picture may impact on the performance because of some variational aspects like the pose, the age, the Makeup the person or the resolution of the picture, namely.

4.2.6 Face Recognition State-of-the-art Performance

Currently, by approaching the face recognition task as first a verification problem (i.e. 2 labels can be predicted: "same" or "different"), an accuracy of 98.63% can be reached when the training is directed by the center loss function, based on the 10,575 subjects and 494,414 images, relying on the VGG16 Architecture whose training takes around 7 days to reach such performances [13]. More generally, all the best performance on the verification task could be obtained by relying on deep learning solution and large datasets. Getting accuracies related to one-shot learning conditions (i.e. limited training set) appeared as difficult...

In the same way, when searching for performances related to the face identification task, all the results were related to very powerful algorithms relying on models that were trained on more millions of face images. In this setting, for a gallery size containing tens of thousands of identities, an accuracy around 97% can be obtained.

Finally, especially in the face recognition problem, it has to be realized that the performance highly depends on many factors like the expression of the faces, how specific the faces are, how 2 faces are similar to each other, how diverse the testing set is, how the background of the picture looks like ... Knowing that, the reader has to understand that the results comparison isn't as relevant as it could be in other straighter applications.

4.3 Style-Based Generator Architecture for Generative Adversarial Networks

In this last section, the data augmentation process employed in this work is developed. It is supported by a Style-Based Generator. Basically, this model represents high level attributes in an embedding space after having been trained as a generator. In particular, the feature representation z in this embedding space can be adapted (under some distribution constraints) so that once projected back into the initial "real space", the high level attributes (like the pose ...) have changed while the physiognomy of the initial face remains the same.

First, regarding the structure supporting the generator, a baseline is defined and a constant is provided as input. Then, over this baseline, some adaIN modules are added in order to capture the style of an external input A and transfer it to the current representation that was progressively built from the input constant. By doing so, it's possible to understand the mapping between the network layers and the high-level feature modelling by observing which image characteristics are impacted.

Concerning this input A, it's also important to notice that it results from the latent code z having been turned through a mapping network into another representation w.

Finally, in addition to that input A, some stochastic noise is introduced (pixel per pixel) to impose small variations and allow the network to learn to use it to control stochastic variations.

The overall structure is illustrated in Figure 4.10.

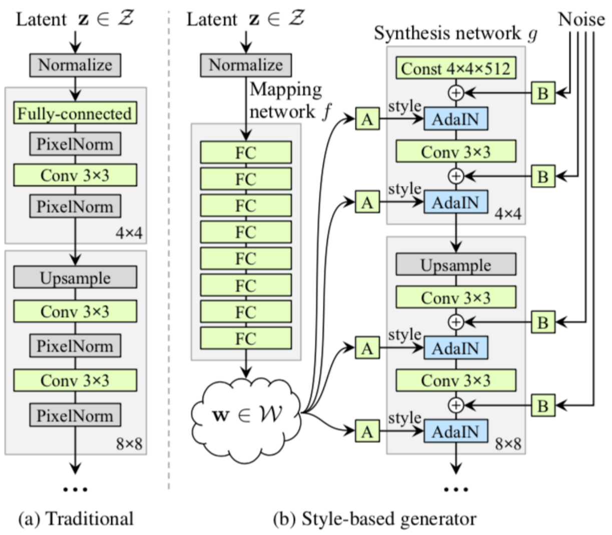


Figure 4.10: Style-based generator

This Style-Based Generator may be used for different applications like mixing several faces, generating synthetic person faces or generating synthetic pictures from a given one, where only high-level features have changed. Here, the interest is on the last point extending the instances related to a given person. By generating new instances, first, a Siamese Network, responsible for distinguishing different people, can become more robust to high-level features variation. Next, it can also appear as useful during the face recognition task: rather than comparing the probe to each picture in the gallery, several instances of the probe are compared, making the final result more robust, again. The bad side of that process is the cost, both in terms of time and space.

Chapter 5

Solution Implementation

Basically, in order to recognize a person represented on a picture, a good approach consists in designing this problem through 2 main phases.

During the first one, the input image is projected into a discriminative feature space. That refers to the **embedding definition phase**. To do so, a deep neural network can be trained to map the input picture to a more discriminative feature representation having the property of highlighting similar identities and separating different ones. In other words, once a feature representation is derived from a picture, the aim is that it remains quite invariant once another picture with the same person is given and, on the opposite, that it appears as as different as possible once a picture with another person is provided. Knowing that, The embedding network can be trained, being supported by a loss function reasoning on the distance between the feature representation of pairs of faces. This is developed in Section 5.3.

Next, during the second phase deeply exposed over Section 5.4, a probe has to be identified. To do so, the feature of that given probe face is compared against a gallery and the most similar gallery faces give the identity of the probe face. This is assimilated to the **classification phase**.

Besides those 2 phases representing the heart of the solution, some other modules have to be defined. A first one is responsible for the the data processing phase all the data have to go through. This step is explained in Subsection 5.2.2. A second one refers to the triplet definition supporting the training phase. In addition to this, a data augmentation module is implemented as well, where synthetic instances are generated from images representing real people. Then, those synthetic instances can be optionally used as training data (or during the face recognition task as extension for the probe). Moreover, the embedding network can be optionally pretrained by being included in an autoencoder so that it can learn how to represent a face.

Finally, a model evaluation structure is defined to get the performance of the Siamese Network that is exploited during the classification phase. All this process is illustrated through Figure 5.1[1].

[1]Where the doted components refers to optional modules, defined to improve the current basic solution

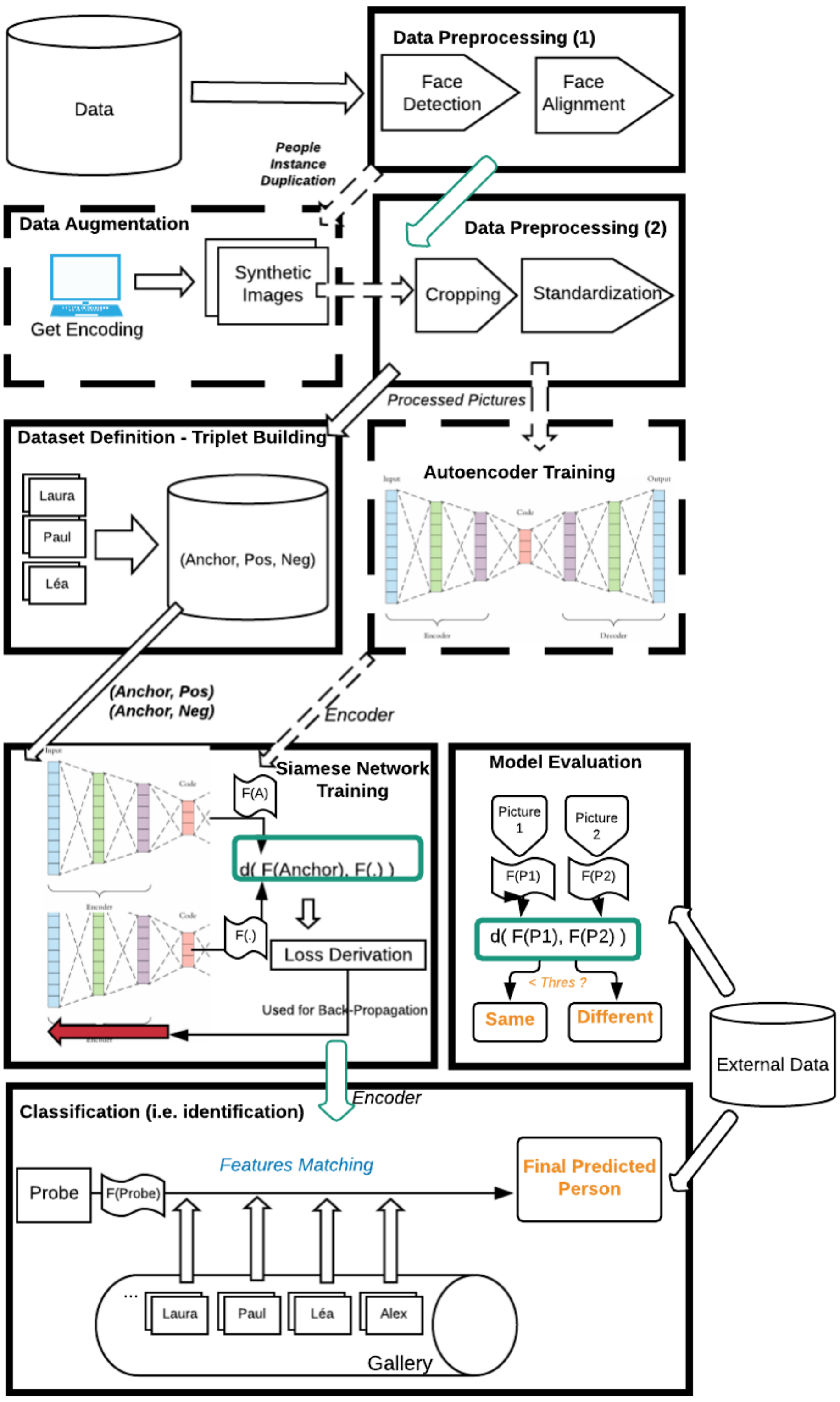


Figure 5.1: Global Structure of the solution

5.1 Implementation Details and Libraries

From a pure implementation point of view, as it can be guessed, the whole pipeline was written in Python.

In addition to this, regarding all the deep learning solution part, at the very beginning of that work, a first very important choice has been the library that would support it. The main deep learning python libraries that were candidates are the following:

- **Keras** that can be considered as a high-level neural networks API and that uses TensorFlow in backend.
- **TensorFlow** that also provides a certain level of abstraction (in terms of computational graph, namely). It's also popular because it offers distributed training support, scalable production deployment options and support for various devices like Android.
- **Pytorch** which can be defined as a lower level framework for deep learning, allowing to write custom layers for instance or look under the hook of optimization tasks. Moreover, it observes strong GPU acceleration and efficient memory usage. However, contrary to Keras, managing implicitly the GPU use, PyTorch requires us to specify when to transfer data between the CPU and GPU.

Finally, after having consulting different posts and platforms where the choice of a python library for neural networks were discussed, Pytorch was selected since it appears as the best in a research context because of its flexibility and low level of abstraction, allowing a better understanding of all what's happening during the learning process.

However, later over this work, some external models and codes have been used, implemented in tensorflow, so that the final solution relies on both pytorch and tensorflow libraries. That final situation mixing different deep learning libraries appeared as problematic at the end, once a sophisticate optimization procedure was imagined, requiring to join both pytorch and tensorflow networks for training. This point is discussed a bit later over Subsection 5.3.5.

5.2 Data

Let's consider now the solution content, by first describing all the data processes. This first step, despite not being the most attractive, remains very important since the data that is used directly impact on the robustness and the accuracy of the final solution.

5.2.1 Databases

As a first step, data has to be collected. Different databases could be found, each of them containing labeled pictures, taken in different conditions (with colored background, with different resolution, more or less cropped ...) and with different kinds of people (with variational gender, age, nationality, with potentially glasses ...). Moreover, some databases also contain some non-frontal pictures. An exhaustive view on the databases that are used is exposed through Table 5.1 and additional comments are given about them.

First, the databases[14] "Aberdeen", "Iranian", "Pain expressions" and "Utrecht ECVP" provide 1240 pictures in total. Despite the quite limited quantity of data, the data source combination setting is expected to lead to some models generalizing better. Moreover, in the context of the

Database	**#People**	**#Images per Person**	**Quantity**	**Resolution** $x \times y$
"Aberdeen"	90	$[1; 18]$ (10 on average)	678	$[336 \times 480; 624 \times 544]$
"Iranian"	34	11	231	1200×900
"Pain expressions"	12	7	84	720×576
"Utrecht ECVP"	69	2	129	900×1200
Georgia Tech face	50	15	750	640×480
Yale face	15	11	159	320×243
Faces94	153	20	2556	180×200
Labeled Faces in the Wild	1192	$[2; 20]$ (6 on average)	13 155	250×250
Celebrities in Frontal-Profile in the Wild	500	10 frontal and 4 profile	4948	$[90 \times 90; 600 \times 600]$
CASIA-WebFace	241	$[5; 30]$ (20 on average)	2336	250×250
FaceScrub cropped	530	$[6; 12]$ (10 on average)	5217	300×300
In Total:	2652	/	30243	200×150

Table 5.1: Databases

Siamese Network, data images are combined to form pairs or triplets to support the training and testing sets, meaning that the final quantity of instances can become very large, as detailed in Section 5.3.3. Next, the *Georgia Tech face*[15], the *Yale face*[16] and the *Faces94*[17] databases are selected as well, all containing colored cropped images with very different people. Those are illustrated through Figure 5.4.

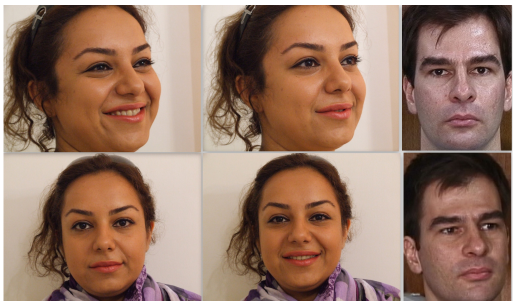


Figure 5.2: "Iranian" and Georgia Tec Face Databases

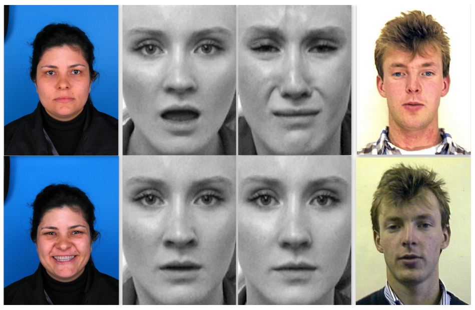


Figure 5.3: "Utrecht ECVP", "Pain Expressions" and "Aberdeen" Databases

Besides those databases, other bigger ones have been found thereafter. A first one is the *Labeled Faces in the Wild* database[18], containing 11,507 images of faces collected from the web. However, only 1192 people have 2 or more related pictures meaning that only 4493 pictures could finally be exploited in classification phase since at least 2 pictures per person are required to match a probe with a person (except if those pictures are prevented from being picked as probe but still, representing an identity by 1 or 2 instances in the gallery remains quite poor). Nevertheless, all the "single instance" face identities (i.e. the unique ones representing a person) can be used in the pre-training phase, when the target is learning a good feature representation of a human face, independently from an label. In the same way, those can represent the Negative N in a triplet. Those considerations encourage the *LLW* database to be exploited during the first phase, where the Siamese Network is trained. Concerning the content of the pictures itself, it can be assimilated to a medium level of difficulty since the background isn't uniform and may even include (part of) other faces, as illustrated through Figure 5.5.

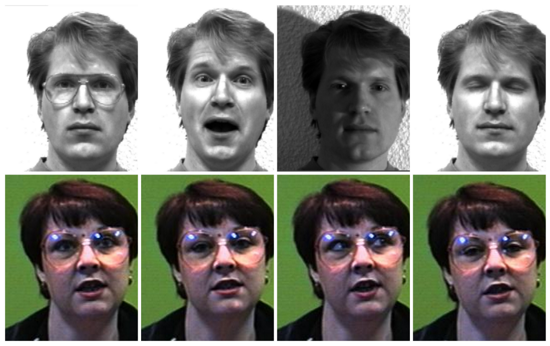


Figure 5.4: Yale and Facd94 Databases

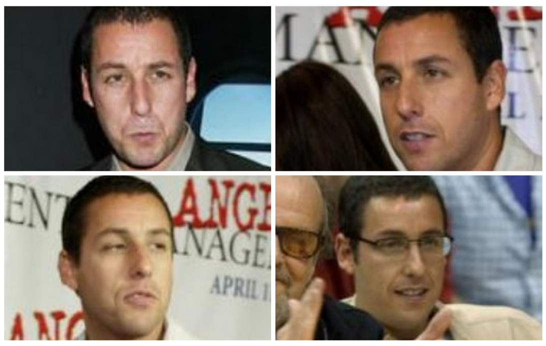


Figure 5.5: Labeled Faces in the Wild

Two other quite large databases are the *Celebrities in Frontal-Profile in the Wild* database [19], containing both frontal and profile views of each person and *Face Scrub cropped*. Each image is cropped and the quality of the pictures varies. Typical pictures they contain are shown through Figures 5.6 and 5.7.

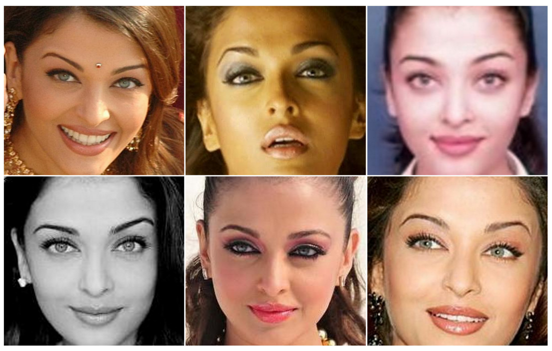


Figure 5.6: Celebrities in Frontal-Profile in the Wild

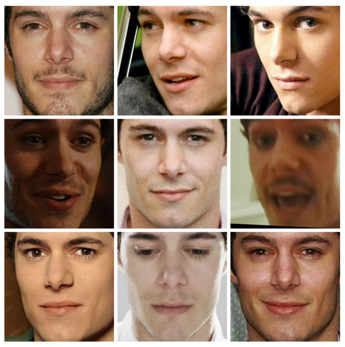


Figure 5.7: Face Scrub cropped

Finally, the *CASIA-WebFace*[20] database is also used to support the experiments. Contrary to all the previous databases, this last one has the advantage of containing a huge amount of images. However, a non-negligible drawback is the poor quality of some pictures. In addition to this, several picture instances related to the same person are sometimes very difficult to associate, due to different positions, light or hairstyle, age..., as illustrated through Figure 5.8, or even wrong. That's why this database was partially checked manually and only 2336 pictures were kept to be use for testing, so that the testing data are reliable.

5.2.2 Data Processing

Now a clear view on the data has been provided, let's detail how those data have to be pre-processed before being exploited in the training and testing phases.

Figure 5.8: Casia Web Face Database

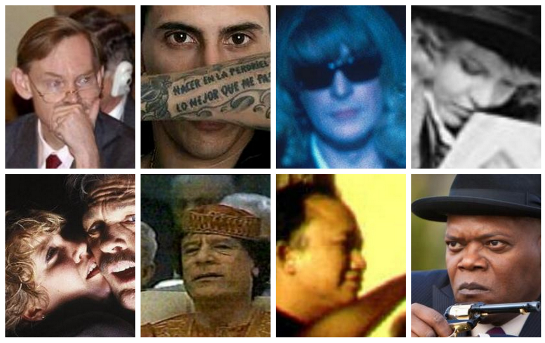


Figure 5.9: Sample of pictures where faces are not detected

First, as illustrated from the Figures showing some instances contained in the different databases, the faces are depicted in different scales, orientations and positions from one picture to another, which makes more difficult the capture of the faces physiologic features. To overcome this weakness, some alignment processing is applied on each picture in order to ensure each image to be well-centered on the face. More specifically, first a LandmarksDetector model is used in order to locate precisely the face and its high level features. The use of this model is pretty interesting since it also allows to catch the pictures of poor quality corresponding to the pictures whose landmarks can't be derived. Such pictures are illustrated through Figure 5.9 to get an idea about the robustness of the LandmarksDetector model. Those pictures were removed from the database.

After this step, the face picture can be re-oriented, potentially shrinked (where a pad is then defined) and cropped so the focus is mainly on the face. The final chosen dimensions of the input images are 150×200. In this way, whatever the initial dimensions of the input images, the processed image feeding the neural network remains manageable (i.e. the input isn't too large) while the amount of information (i.e. the image content) is still complete enough (the facial characteristics still stand out). The evolution carried by the alignment processing is illustrated through Figure 5.10.

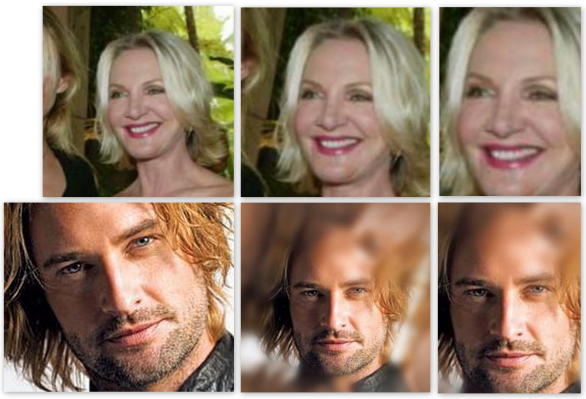


Figure 5.10: Alignment Process

Finally following this, once the RGB images have been turned into tensors, each value (in [0, 255]) is standardized following the formula:

$$input = \frac{(input - mean)}{std}$$

where here the mean is set to 0.5 and the standard deviation to 1.

5.2.3 Data Augmentation supported by synthetic images

As a reminder, the main goal of this work is learning from few data, which appears as quite challenging since the main issue of deep learning methods is that they need to be trained from a large amount of data so that enough variations can be captured to generalise to unseen samples. In front of this, as mentioned before, a quite intuitive way to proceed is generating new data from the current ones, where those new data are variant of the ones contained in the initial databases. Here, in particular, 2 approaches can be considered:

- **Instances Generation** where from a given real picture of a given person some variants are generated by playing on the smiling expression, the age and the gender, in a limited extent. The result of this process is illustrated through Figure 5.11.

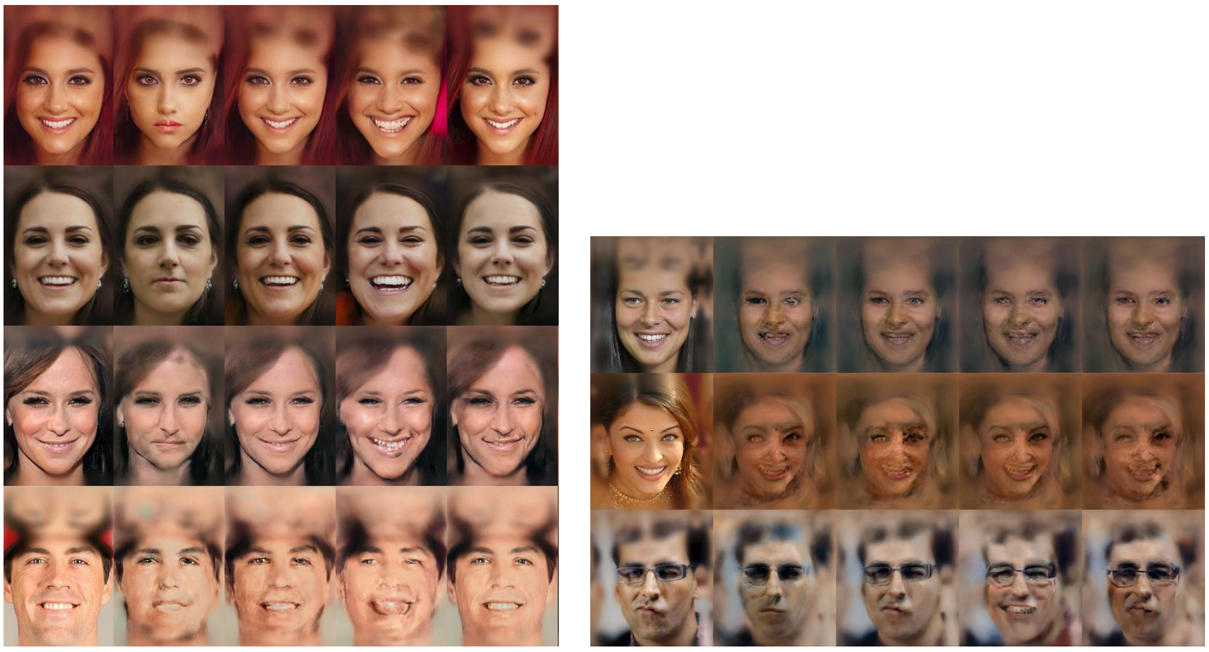


Figure 5.11: Synthetic Images: The first column contains the real initial pictures and the rest illustrates the synthetic images

- **Person Generation** where synthetic faces are created by the generator and several instances of those synthetic people can be generated following the instance generation procedure discussed just before. More specifically, a synthetic person is defined from a "random" latent representation vector, derived under some distribution constraints, and projected back to initial space pictures are represented in.

Regarding the implementation of this data augmentation module, it was supported by the open-source software given by [21]. More specifically, here are the steps that were followed in order to generate synthetic images from a real given one p_{real}:

1. **Get Latent Representation z from p_{real}.** To do so, a pretrained perceptual model is optimized considering p_{real} over a certain number of iterations. Concerning that number of iterations, it has to be mentioned that it must be set to at least 500 in order to get a relevant latent representation, proper to the current person. To fix it, there's a tradeoff that has to be considered between;

   - A large computation time but generated pictures with higher-quality and more robust to external feature variations
   - A limited computation time but also generated pictures with a limited quality

   Here, the number of iterations was finally set to 1200.

2. **Application of variations on z**. Once **z** is derived, some variation can be applied to some specific values z_i with respect to a specific distribution in order to preserve the physiognomy of the current person, by only playing on "external features" like the face expression. The latent direction leading the values changes were already learnt from previous experiment by the author of the open-source software [21]. Right now, 3 external feature variations can be applied (more or less intensively, depending on some coefficient setting):

   - Smiling Expression
   - Age
   - Gender

   After some testing experiments, a good scenario was generating 4 synthetic images, 3 referring to different smiling expression and 1 to different ages.

3. **Integration of the new synthetic pictures in the database**

From Figure 5.11, it can be observed that for some initial picture, the generated instances are quite "bad", at least from a human perspective, even after the number of iterations and the coefficients measuring the intensity of the variations have been tuned. That raises some questions about how relevant synthetic images, poor from a human point of view, can be for a machine. This question is answer in chapter exposing the results obtained across different scenarios where synthetic data are exploited. In addition to that, it obviously also open the door to some improvement over this synthetic data generation phase.

5.3 Optimal Feature representation for the Face Recognition

5.3.1 Training Choices: Hyperparameters

In this part, the focus is on the setting of the weight decay, the learning rate, the number of epochs and the batch size. It can be very tricky to choose the best combination of all those hyperparameters. Many scenarios have been experimented, each hyperparameter varying on a specific range of values. After this experimental phase, the final selected values are:

- BATCH_SIZE= 32
- WEIGHT_DECAY= 10^{-3}
- (initial) LEARNING_RATE= 10^{-3}
- NB_EPOCHS[2] $\in [30; 70]$

In particular, concerning the learning rate, a learning rate scheduler is defined. Three scenarios have been studied:

- A first basic one with a **constant learning rate**
- A second one with a **linear-decreasing learning rate**
- A third one with an **exponential-decreasing learning rate**

After the experimentation session, the linear-decreasing learning rate appears as providing both the best result and learning behavior (with few instability).

Besides the learning rate, the number of epochs was set in order to avoid the overfitting phase.

Around those 2 hyperparameters tuning, many scenarios, with a variable data quantity, have been experimented and for each of them, the epoch where the highest f1-measure was reached has been registered. From those data, as you can guess, it could be observed that the optimal number of epochs highly depends on the size of the training set. Here, in a work based on one-shot learning, this aspect has to be kept in mind since the quantity of data is discussed (and then may be different from one experiment to the other). For a very small number of input pairs, the number of epochs has to be reduced to avoid any overfitting while it's increased for large training sets. Typically, 30 epochs are sufficient once training a network on 3000 triplets while 70 epochs may be required for a training set of about 15000 triplets. In the same way, the optimal learning rate management has to take the quantity of data into account. Typically, when training is based on more than 10000 triplets, a constant learning rate set to 0.001 leads to a better learning process since the data are more diverse, meaning that there's "more to learn" from. In addition to the data quantity impact, it's important to highlight the fact that both the number of epochs and the learning rate should be tuned together. More precisely, for a small number of epochs, a constant learning rate is preferred while for a larger number of epochs, the learning rate should be decreased to avoid overfitting.

Finally, notice that the complexity of the architecture of the network impacts on those observations. Here, the quantitative information that are mentioned refers to the results related to the basic architecture. All the architectures that were experimented are developed in the next section.

[2]This is adapted depending on the size of the training set, as discussed below

5.3.2 Training Choices: Embedding Network Structure

When designing a network structure, a balance has to be found between network depth, model performance, training time and memory use. Here, 3 architectures were experimented:

- **"Basic" Net** (inspired from the simple architecture given in [22] that represented the start point of this work)
- **AlexNet**
- **VGG16**

A more detailed view on the architecture is exposed through Tables 5.2, 5.3 and 5.4. From those Tables, it can be noticed that from the Basic Net to the VGG16 Net, the architecture is more and more complex. The target behind the experimentation of those 3 architectures was to see how much the performance can be improved by using a more sophisticate structure of network. The corresponding results are shown in the next chapter.

From Table 5.2, it can be seen that the Basic Net produces a 512-dimensional output feature vector which appeared as being able to capture the stable individual features. Similarly, an output vector size of 1024 was tested as well, since 512 features may appear as incomplete, but no improvement is brought in this setting.

Regarding the 2 other architectures exposed in Tables 5.3 and 5.4, 3 scenarios have been explored, where the final feature representation contained respectively 1024, 2048 and 4096 features. The best performances are reached with the smallest embedding vector for both architectures.

Layer		**Feature Map**	**Size**	**Kernel Size**	**Stride**	**Activation**
Input	Image	1	$200 \times 150 \times 3$	/	/	/
1	Convolution	64	$194 \times 144 \times 64$	7×7	1	relu
	Dropout (0.2)	64	$194 \times 144 \times 64$	/	/	/
	Pool	64	$48 \times 36 \times 64$	4×4	1	/
2	Convolution	128	$44 \times 32 \times 128$	5×5	1	relu
	Dropout (0.2)	128	$44 \times 32 \times 128$	/	/	/
	Pool	128	$11 \times 8 \times 128$	4×4	1	/
3	Convolution	256	$7 \times 4 \times 256$	5×5	1	relu
	Dropout (0.2)	256	$7 \times 4 \times 256$	/	/	/
	Pool	256	$1 \times 1 \times 256$	4×4	1	/
Output	Linearization	/	512	/	/	relu

Table 5.2: Basic Net

Layer		**Feature Map**	**Size**	**Kernel Size**	**Stride**	**Acti-vation**
Input	Image	1	$200 \times 150 \times 3$	/	/	/
1	Convolution	64	$48 \times 35 \times 64$	11×11	4	relu
	Pool	64	$23 \times 17 \times 64$	5×5	2	/
2	Convolution	192	$23 \times 17 \times 192$	5×5	1	relu
3	Convolution	384	$19 \times 13 \times 384$	7×7	1	relu
	Pool	256	$8 \times 5 \times 384$	5×5	2	/
4	Convolution	256	$6 \times 3 \times 256$	5×5	1	relu
5	Convolution	256	$6 \times 3 \times 256$	3×3	1	relu
	Pool	256	$2 \times 1 \times 256$	3×3	2	/
	Dropout (0.5)	256	$2 \times 1 \times 256$	/	/	/
Output	Linearization	/	1024	/	/	relu

Table 5.3: Alex Net

Layer		**Feature Map**	**Size**	**Kernel Size**	**Stride - Padding**	**Acti-vation**
Input	Image	1	$200 \times 150 \times 3$	/	/	/
1	$2 \times$ Convolution	64	$200 \times 150 \times 64$ 3×3		1 - 1	relu
	Pool	64	$100 \times 175 \times 64$	2×2	2 - 0	/
2	$2 \times$ Convolution	128	$100 \times 175 \times 128$	3×3	1 - 1	relu
	Pool	128	$50 \times 37 \times 128$	2×2	2 - 0	/
3	$3\times$ Convolution	256	$50 \times 37 \times 256$	3×3	1 - 1	relu
	Pool	256	$25 \times 18 \times 256$	2×2	2 - 0	/
4	$3\times$ Convolution	512	$25 \times 18 \times 512$	3×3	1 - 1	relu
	Pool	512	$12 \times 9 \times 512$	2×2	2 - 0	/
5	$3\times$ Convolution	512	$12 \times 9 \times 512$	3×3	1 - 1	relu
	Pool	512	$6 \times 4 \times 512$	2×2	2 - 0	/
	avgPool	512	$7 \times 7 \times 512$	/	/	/
6	Linearization	/	512	/	/	relu
	Dropout (0.5)	512	512	/	/	/
7	Linearization	/	1024	/	/	relu
	Dropout (0.5)	1024	1024	/	/	/
Output	Linearization	/	1024	/	/	relu

Table 5.4: VGG16 Net

5.3.3 Siamese Network: Distance-based approach

Triplet Definition

In order to identify a person, a Siamese Network is defined, based on the distinction between 2 faces on 2 given images. More precisely, as expressed in Section 4.1.4, the network is trained based on triplets (A, P, N), where A is the anchor, P is another picture representing the same person and N a picture representing another person. Notice that, depending on the loss, the triplets can be split into pairs (A, P) and (A, N).

In the particular context of one-shot learning, relying on triplets rather than single input images really represents a strength because it allows to significantly increase the number of samples composing the training set. Typically, imagine you have on your disposal a dataset of I people, each one having J corresponding pictures. If each picture is defined once as the anchor and some corresponding "positive" and "negative" pictures are picked from the image dataset, a set of $(I * J)$ triplets is defined, corresponding to the number of pictures. However, it's important to realize that for each person i, C_J^2 different pairs (A, P) can be generated. Knowing that, in the case where a database of 20 people is provided, each person having 8 corresponding instances, 560 different pairs[3] (A, P) can be defined.

Besides this, regarding the total number of different triplets that can be built, it's given by:

$$I.((I-1).J).\sum_{j=2}^{J}(j-1)$$

where:

- the first factor I is the total number of people
- the second one $(I-1).J$ represents the number of possible negatives N
- the last one $(j-1)$ refers to the number of possible positives P, considering successively each instance j related to the current person so that there's no duplicate.

Applying this to the last example, 85120 different triplets[4] can be derived.

At this step, in the case where several databases are used and knowing the large number of possible triplets that can be defined, it can be relevant to ensure that N is picked a sufficient number of times among the pictures coming from the same database, especially in the case where databases are small and very specific (i.e. where the external features are the same). In this way, the differentiation is more difficult and the learning phase is more relevant because the focus is constrained to be on the face itself since the background and other extra characteristics related to the "external conditions" the picture was taken in are the same.

Triplet Loss

Concerning the loss functions to employ, first, the triplet loss function was experimented to support the model training. It's given by:

$$\mathcal{L}(A, P, N) = \max(0, d(f(A), f(P)) - d(f(A) - f(N)) + \alpha)$$

In this setting, 3 hyperparameters choices have to be made.

First, seeing the expression of the triplet loss, a distance metric and a margin α have to be set. For the distance metric, as for the classification phase, the Manhattan distance, the mean-square distance and the cosine distance were experimented and the final picked one was the mean-square distance, giving the best model performance. Next, for the margin, again different

[3]where $560 = 20.C_8^2$

[4]where $85120 = I.((I-1).J).\sum_{i=2}^{J}(i-1) = 20.(19.8).\sum_{i=1}^{8}(i-1)$

scenarios were tested and no significant difference was observed around the common value 0.2. This last value was then selected.

In addition to this, a threshold allowing the model to support a final boolean prediction has to set so that the model can be evaluated on the verification task during both the training process and later. The tuning strategy for this threshold consists in computing the median M_p of the distances related to all pairs (A, P) and the median M_n of the distances related to all pairs (A, N) belonging to the training set. Then, from those medians, the threshold is determined after the last iteration as followed:

$$t \leftarrow \frac{M_p + M_n}{3}$$

Another more sophisticate approach to tune the threshold and understand the model's behavior consists in representing the the False Acceptance Rate (FAR) and the False Rejection Rate (FRR) expressed according to the threshold, as illustrated through Figure 5.12. In this setting, the selected threshold value t^* is such that $FRR(t^*) = FAR(t^*)$, where there's a perfect performance balance on the 2 classes ("same" and "different", referring to people identities). Notice that here, the FAR refers to the proportion of face pictures that were assimilated as representing the same person as the anchor, while it's not the case. Similarly, the FRR represent the proportion of face pictures predicted as representing the same identity as the one related to the probe, while they are not.

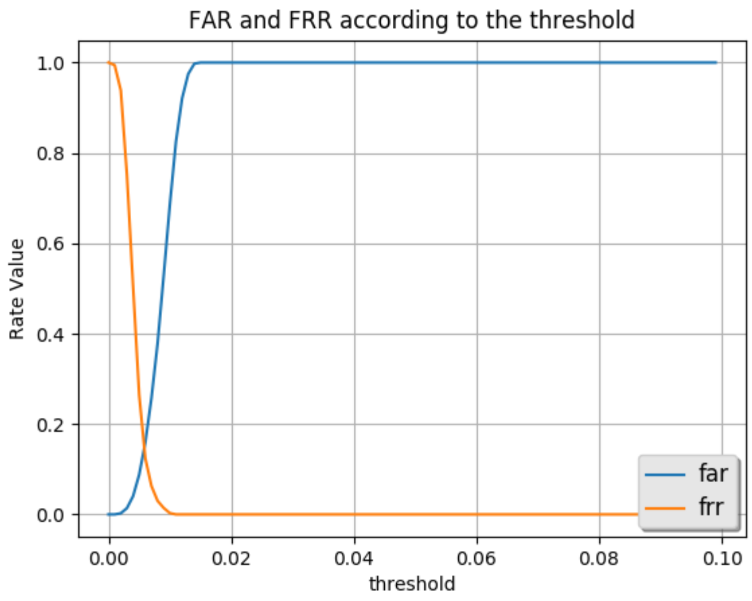


Figure 5.12: Equal Error Rate = 0.16

In addition to this, the intersection gives the Equal Error Rate (EER) $(= FRR(t^*) = FAR(t^*))$ that could be used as metric to evaluate the model performance for the distance loss based models. However, since all the losses that are experimented are not distance-based, the classical f1-score is preferred.

Contrastive Loss

Next to the triplet loss, the contrastive loss is defined to be compared to the previous loss function. Like the triplet loss, it's based on distances. However, it considers pairs of images rather than triplets. The contrastive loss is given by:

$$\mathcal{L}(A, X) = \frac{1}{2}.(1 - Y).d(\mathcal{F}(A), \mathcal{F}(X)) + \frac{1}{2}.Y.\max(0, \alpha - d(\mathcal{F}(A), \mathcal{F}(X)))$$

where A is the anchor, X is the second picture of the pair (that can be positive P or negative N), Y is the expected output (being 0 if X is a positive instance with respect to A and 1 otherwise) and α is the margin.

Finally, similarly to the triplet loss, at the end of the training process, a threshold t is defined in the same way so that the model can output a final boolean prediction telling if the 2 pictures are predicted as representing the same person.

Cross Entropy Loss

Besides those 2 first "pure" distance-based losses, a cross entropy loss function is experimented as well, relying on the certainty of the final prediction related to the verification task. In this case, 2 classes have to be explicitly defined so that a score can be assigned to for leading the final prediction and supporting the loss. Knowing this, each triplet is split into 2 samples $[(A, P), 0]$ and $[(A, N), 1]$, where 0 (resp. 1) is the label specifying that A and P (resp. N) represent the same (resp. different) people. To employ this loss, as before, the input images from the input pair are projected into an embedding space. 2 vectors are obtained from this first step and the element-wise difference $d(F(A), F(X))$ between them is computed. Then, the resulting difference vector goes through a linear layer connected to 2 final neurons, each being associated to the 2 classes, as illustrated through Figure 5.13. A score $s[.]$ is associated to each of those two classes and the target is maximizing the score of the true class.

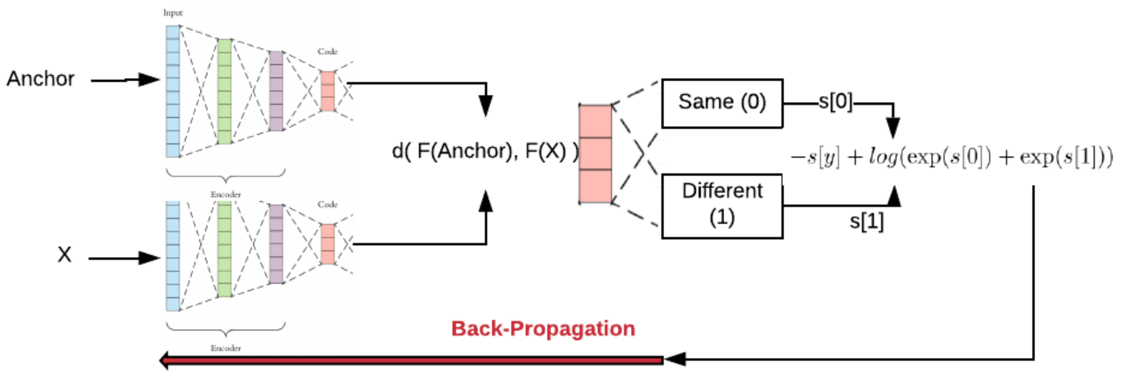


Figure 5.13: Cross Entropy Loss

5.3.4 Pretraining using an Auto-Encoder

Now the global conditions the Siamese Network can be trained in have been described, let's explain how the training phase could be reinforce.

In general, when designing a deep learning solution, it has to be realized that directly learning to perform well on some task from only samples composed of input images and a corresponding expected output may be very tricky! As an intermediate step in the learning process, an interesting approach consists in pretraining a network, where the aim is just making it "familiar" to the concept of interest (the face in this case). More precisely, the goal is to train a network able to derive a relevant feature representation of a face.

To do so, an auto-encoder can be defined. It's composed of 2 main components - an encoder and a decoder - that are connected together and works as followed. First, an image is provided as input to the encoder which derives a feature representation corresponding to this image. Then, from this feature representation, the decoder has to retrieve this initial image at best. So the overall structure is trained in order to minimize the difference between the output of the decoder and the initial input image. This is illustrated through Figure 5.14.

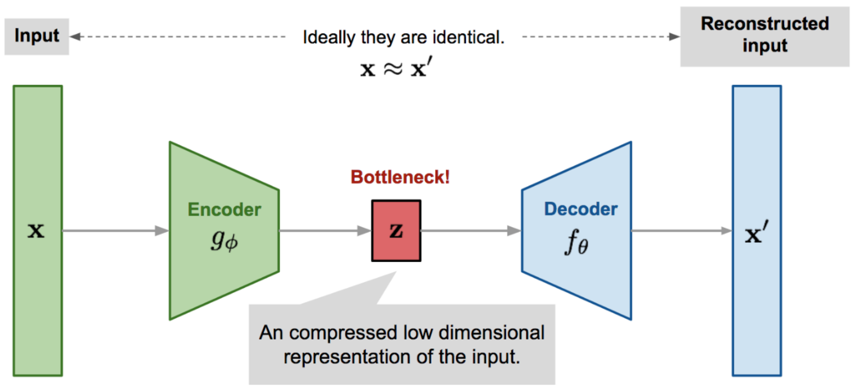

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
$\mathbf{x}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Figure 5.14: Autoencoder Structure[5]

This process can be assimilated as acquiring pre-knowledge by performing a simple task (here consisting in defining a good feature representation of the image so that it's easy to get it back) in order to exploit it later for a more complex task (here referring to differentiate faces of different people, which requires a good embedding system). By doing so, the network isn't trained from scratch: it has "already an idea about what a face looks like" and how it can be represented.

Here in particular, 3 autoencoders have been defined, one for each architecture specified in subsection 5.3.2, and for each of them the decoder has the sequence of layers composed of a linear layer followed by 3 transposed convolutional layers alternated by maxpool layers. All that sequence is ended by a sigmoid activation function.

After a training over 180 epochs based on about 17500 face images, the outputs resulting from one face instance provided as input to each autoencoder are illustrated through Figure 5.15.

For all the three structures, it can be noticed that the results are pretty similar. In addition to that, it has to be mentioned that in any case the output is quite invariant to the specific face that is given, meaning that the structure is able to represent a face in general, but not the features characterizing the identity of a person. This can be due to the simple structure of the decoder or to the limited quantity of data. Since unlabelled data can be exploited here, it could have been relevant and not too costly to collect some additional ones (still independent from the ones belonging to the data used for testing) and train the autoencoder on.

[5]Notice that the notations on that figure don't match with the ones used in this report. The target of that figure is only to illustrate the autoencoder concept.

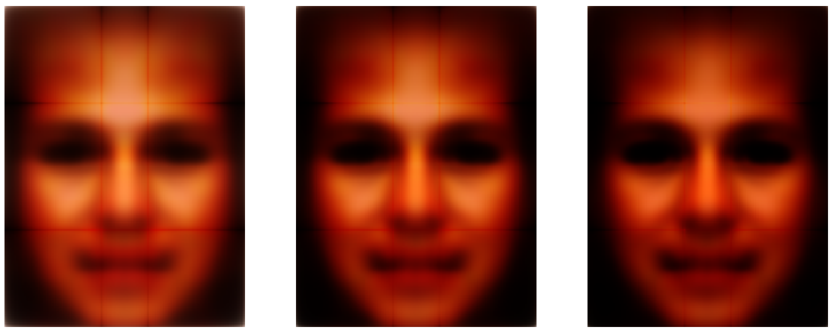


Figure 5.15: Result from the autoencoder based on Basic Net, Alex Net and VGG16

5.3.5 Iterative Solution Improvement

In this subsection, some strategies are considered in order to improve the performance of a current model that has been trained. In all the following strategies, the basic idea is trying to identify where the current solution is bad and then retraining the models on the face triplets detected as currently not well-understood. In particular, here are the 2 procedures that have been defined:

- **Retraining on triplets where** $d(\mathcal{F}(A), \mathcal{F}(N)) < d(\mathcal{F}(A), \mathcal{F}(P))$
  This first solution typically exploits the fact that from the same database, a large number of triplet datasets can be derived based on the same set of images. Knowing this, the idea here is constraining the triplets that are built by reasoning on the distances between features representations computed from the current Siamese Network. That can be assimilated to the first meta-learning approach reflected through Figure 4.1, where an iterative process is defined over different training set to lead the training phase.

- **Adversarial Siamese Network** where synthetic data are defined so that the Siamese Network can be retrained on and learns from cases that couldn't be well captured in the past. To do so, the following main steps are followed:

  1. Predefinition of consistent directions $\mathbf{w}$. The aim here is to enforce the variation of some specific external features (gender, smiling ...) without altering the physionomic ones. As a reminder, right now, 3 have been defined in the solution proposed in [21]. Those were established by using tagged data. Here, the procedure that is employed is much more simpler since no tagged data have been collected. The idea is only applying little variations to $\mathbf{z}$ by experimenting different directions $\mathbf{w}$ whose components are randomly picked, constrained to follow the normal distribution $\mathcal{N}(0, \epsilon)$, where epsilon is set to a small value (0.005 typically).

     That first step has been implemented and experimented, giving as expected some slight variation of the initial real picture and still preserving the physionomic features since the variation are limited.

  2. Optimization of a specific consistent direction $\mathbf{w}$. To do so, a training set made up of pairs of face pictures (A, B) is defined. Then, for one specific iteration, the encoding $\mathbf{z}_B$ (related to picture B) is computed and the current variation $\mathbf{w}$ is applied. After that, the resulting synthetic picture B_{synth} is derived and given as input, joined to A, to the current Siamese Network outputting a distance $d(\mathcal{F}(A), \mathcal{F}(B_{synth}))$.

     Over one epoch, $\mathbf{w}$ is updated in order to:

  - maximize the distance if A and B represent the same person (since a bad result corresponds to a large distance when the same person is represented)
  - minimize the distance if A and B represent different people

  This process is repeated over 50 epochs and supported by 1000 pairs of faces.

3. Once a final direction $\mathbf{w}$ is obtained after training, it's applied to generate synthetic data supporting a new training set that can be used to retrain the Siamese Network that appeared as very poor when dealing with them in the previous step.

All this process is illustrated through Figure 5.16.

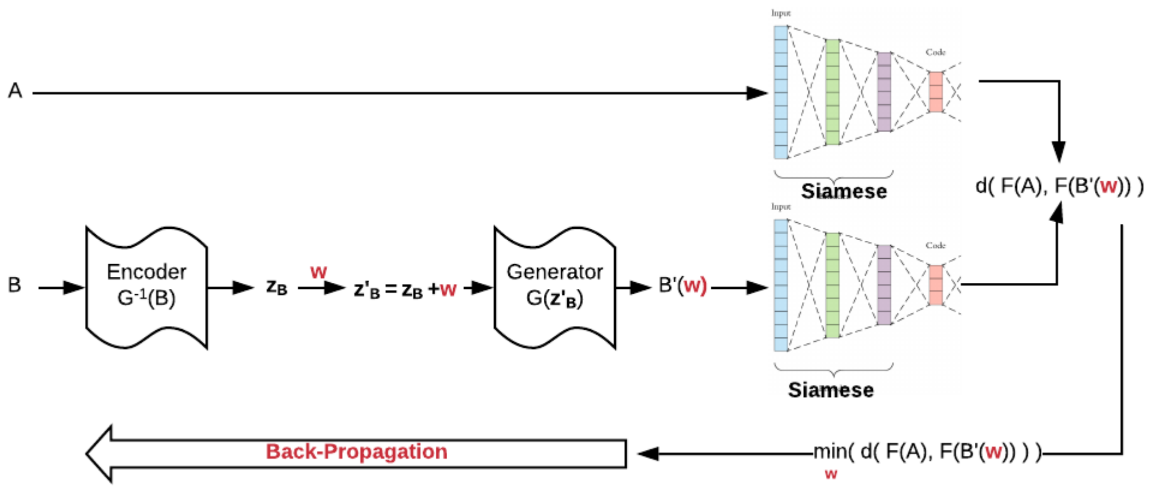


Figure 5.16: Adversarial Siamese Network structure, where $i_A \neq i_B$ (i_A referring to the identity of the face on picture A)

Unfortunately, that remains a piece work in progress since that idea was raised quite late over this work and a big issue was encountered, referring to the association of networks based on different libraries. A quite tedious but safer approach would have been to reimplement all the pytorch datasets, network and loss function in the tensorflow framework to make compatible all the system. However, to avoid that big redundancy and modification process all over the current solution, it was preferred to explore a way to turn a pytorch model into a tensorflow one. To support this, the best seemed to pass through the *onnx* intermediate form. However, the final tensorflow model that could be derived wasn't trainable... Seeing that and knowing that this approach would also require non-negligible extra time to be tuned, evaluated and discussed properly, that part of the solution was put on hold.

In both cases, notice that the process can be repeated a certain number of times and based on the same initial data, depending on the expected final performance. However, in the first strategy, this number of repetitions has to be bounded, otherwise the model ends up overfitting. This number of repetitions has to be tuned with respect to the initial quantity of data and the number of triplets related to each face picture.

5.4 Classification (i.e. Face Identification)

Right now, the focus was really on getting the most relevant feature representation as possible. After this step, the classification still has to be performed by defining an algorithm deriving the identity of a probe $\mathcal{P}$ by comparing it to the pictures in the gallery $\mathcal{G}$. After the comparison process, the final selected label is the one whose corresponding faces were predicted as the most similar.

To perform this comparison process, for one particular probe $\mathcal{P}$, each person i in the gallery is considered. Then, for each person, each corresponding picture $\mathcal{G}_{ij}$ is compared to the probe. At this step, 2 procedures can be followed: a **boolean approach** or a **distance approach**.

5.4.1 Boolean Approach[6]

In this first approach, the pair of pictures $(\mathcal{P}, \mathcal{G}_{ij})$ is given as input to the Siamese Network $\mathcal{S}$ outputting 0 if both pictures are predicted as representing the same person and 1 otherwise. As a reminder, whatever the loss that has been used during the training phase (soft-max or distance-based), the model is always able to derive a final boolean answer, resulting from the embedding of the 2 input pictures, predicting if they represent the same identity. In this setting, a score s_i is associate to each person i in the gallery and this score is incremented each time $\mathcal{S}(\mathcal{P}, \mathcal{G}_{ij}) = 0$.

For a given probe, once a score has been computed and attached to each person in the gallery, the final predicted identity is the one corresponding to the the person in the gallery whose score is the highest. This can be translated through:

$$i_{\mathcal{P}} = \underset{i}{\operatorname{argmax}}\{\sum_j \mathbb{1}(\mathcal{S}(\mathcal{P}, \mathcal{G}_{ij}) == 0)\}$$

where $i_{\mathcal{P}}$ is the index of the predicted person.

In addition to this, in order to save some computation time, a *tolerance parameter t* is defined to limit the number of comparisons. This parameter specifies the maximum number of mistakes the model is "allowed" to make per person. More precisely, for a given set of pairs $(\mathcal{P}, \mathcal{G}_{ij})$, once the model has predicted t times 1, no new pair $(\mathcal{P}, \mathcal{G}_{ik})$ is tested since the person i is directly rejected.

5.4.2 Distance Approach

. A second way to compare the probe $\mathcal{P}$ to a gallery picture $\mathcal{G}_{ij}$ is by computing the distance between their feature representation - $d(\mathcal{F}(\mathcal{P}), \mathcal{F}(\mathcal{G}_{ij}))$ - in order to rank the gallery people according to.

To do so, first, several distance metrics $d(\mathbf{x_1}, \mathbf{x_2})$ have been experimented:

- the Manhattan distance $|\mathbf{x_1} - \mathbf{x_2}|$
- the Euclidean distance $\sqrt{(\mathbf{x_1} - \mathbf{x_2})^2}$

[6]Also referring to a voting approach based on the verification task

- the Cosine distance $\dfrac{\mathbf{x_1} \cdot \mathbf{x_2}}{\max(\|\mathbf{x_1}\|_2 \cdot \|\mathbf{x_2}\|_2, \epsilon)}$

More precisely, once a distance has been evaluated between the probe and each picture of the gallery, for each person i, the median of the distances $d(\mathcal{F}(\mathcal{P}), \mathcal{F}(\mathcal{G}_{ij}))$ is computed over all the j instances (related to one particular person) to get a single distance value d_i for each person in the gallery. Finally, the predicted identity for the probe is the one corresponding to the gallery person i whose d_i is the lowest. This can be translated through:

$$i_{\mathcal{P}} = \underset{i}{\operatorname{argmin}}\{\underset{j}{median}(d(\mathcal{F}(\mathcal{P}), \mathcal{F}(\mathcal{G}_{ij}))\}$$

where $i_{\mathcal{P}}$ is the index of the final predicted person.

5.4.3 Gallery

Besides the 2 possible strategies that can support the classification algorithm, let's now describe deeper the gallery. First, the size is set between 10 and 400. This last choice can be understood when seeing the results exposed in the chapter developing the experimentation phase, showing that the model couldn't perform well for larger sizes... In addition to this, a balance restriction is imposed so that each person in the gallery has the same number of pictures. Notice that this last condition is important when the classification is supported by the voting system, especially if the Siamese Model the system relies on tends to output more False Positive compared to False Negative. In that case, when employing an unbalanced gallery, a person being represented by more pictures than the others is naturally more likely to be predicted. On the contrary, relying on a Siamese model outputting more False Negative can be an issue as well since the expected identity may be "missed" during the comparison process. Seeing that, it can be said that the "way" the model learnt can directly be reflected during the classification phase, when relying on a voting system. Knowing this, more formally, 2 important aspects have to be carefully considered to ensure the classification to work well:

- The positive recall related to the Siamese model evaluated on the verification task has to be "high enough". Indeed, if it's not the case, for many probes, the model could miss the matching with the person in the gallery while, on the contrary, if the positive recall is high and the negative recall lower, it's not such a big issue as before since the corresponding gallery pictures are more likely to be matching, besides potential other mismatch that should be overcome through the voting system (while it can't for a low positive recall where the score related to the correct identity may be too low).

- The number of pictures per person should be high enough (and sable, as mentioned before). By imposing a number of pictures which is high enough, if the Siamese model has acceptable performance, the final predictions are more likely to be correct since the voting (or ranking) system would be more robust (because based on more comparisons).

Besides this, it has to be mentioned that each probe has to match with a person in the gallery since no distractor is defined. However, a threshold is defined anyway to express the degree of certainty of the final prediction. More precisely, for both approaches (based on a ranking), the score difference between the 2 top people after having been sorted has to be large enough. Otherwise, several identities are predicted.

Chapter 6

Experiments

As a reminder, the global face recognition system implemented in this work is made up of 2 phases. First, the Siamese network is defined, whose performance can be evaluated on the face verification task. Next, the face recognition task itself can be achieved over the classification phase, based on the Siamese Network, whose performance relies on the prediction of the identity of a probe.

In Section 6.1, the focus is on the performance of the Siamese Network. Next, once a model performing well has been derived, the classification step is explored over Section 6.2.

6.1 Siamese Network Evaluation

6.1.1 Evaluation Metrics

As specified before, in the case of the Siamese Network approach, both training and testing sets are naturally balanced because they are derived from the triplets definition 2 pairs (A, P) and (A, N) can be derived from. Knowing this, the accuracy metric can be used in order to correctly reflect the performance of the model.

Besides this, notice that the recall or the precision metrics can be interesting to highlight as well, in order to understand better the model's behavior. Typically, a Siamese model may tend to predict most of the picture pairs as representing the same person and still reach acceptable final results by doing so, while, in general, the expected behavior would be more to have balance between the similarity recognition and the differentiation between faces. Of course, this aspect depends on the final application of the face recognition system. If the aim is more on people detection, maximizing the positive recall is preferred typically. But that remains a particular case.

Considering this, in order to take into account those 2 last metrics and in the perspective of potentially later defining unbalanced datasets (since much more pairs (A, N) could be defined and exploited), the **f1-measure** is used all along the Siamese model evaluation. Moreover, in this way, the same evaluation procedure can be applied in the case where a classification setting is experimented (where each class is represented by one person).

6.1.2 Basic Conditions

When evaluating and comparing several Siamese models, the same training, validation and testing sets are used, all based on the same images leading to the same triplets. This last restriction on the triplets could be omitted in the case of large datasets. However, here, since it may happen that few data are employed, the triplets themselves may impact on the performance since some pair of faces are naturally easier to differentiate (or assimilate) than others.

In particular, if there's no specification, the databases the training and the validation sets are built from are the *LFW* and all the "small" databases *Aberdeen*, *Iranian*, *Pain Expressions*, *Utrecht ECVP*, *Georgia Tech face*, *Yale face* and *Face94*. By doing so, the diversity of both the training and the validation sets is ensured.

Regarding the quantity of data, **8104**[1] **pictures** support the training set and no synthetic data are used, except if specified. From those 8104 pictures, 16000 triplets are generated (i.e. more or less 2 triplets are related to each picture). That number of triplets can obviously be increased but in practice, it has been observed that a too large number of triplets from an initial quite small set of pictures leads the model to overfit.

Besides this, concerning the parameters setting related to the Siamese model itself, the default architecture is the **Basic Net**, trained based on the **triplet loss** and **pretrained** with an autoencoder.

6.1.3 Data discussion

Triplet Nature Impact

The impact of the triplets briefly discussed before is investigated here, by repeating the same evaluation process 10 times on a model trained based on the same initial images, but having led to different triplets. The f1- score average that is obtained is $86,23\%$ and the associated **standard deviation is $\mathbf{2,89}$%**. From those quantitative observations, the reader can realize the the impact of the image combination process is non-negligible.

Data Diversity Robustness

All over this work, it has been observed that the performance of a model are also strongly dependent on how different the pictures are both over the training set and in the training compared to the validation set, in terms of people and external conditions. That meets the observation related to the impact of the randomness of the image combination process. This point is important to study in order to get a real idea about the context where the models that are built could be exploited and give good results. Indeed, for instance, for an application scanning an ID picture and having to identify the person on, a model performing well only on pictures belonging to the same database would be enough to support the application. However, in the case where a face has to be recognize to unlock a phone for example, a model robust to the background, the face orientation and other external condition features is required to get reliable results.

Here in particular, 4 models are compared, all of them being trained on the same amount of data:

1. The first one is trained on pictures belonging to the same database and is evaluated on a testing set containing the same people (but obviously different instances). This refers to a close set identification framework.

[1]This total number is smaller than the total quantity that can be computed from Table 5.1 since some pictures contain people turning sideways or pictures of low quality that are then removed. Moreover, part of the pictures are used to support the validation set.

2. The second one is trained again on pictures belonging to the same databases and is evaluated on a testing set containing different people picked in the same databases (meaning that the external conditions on the pictures are pretty similar in the training and the testing set).
3. The last one is trained on pictures belonging to a mix of databases and is evaluated on a testing set containing the different people picked in another database, the *CASIA-WebFace*, whose face pictures are very diverse.

The corresponding resulting performances are illustrated in Table 6.1. From that Table, as expected, it can be observed that the more different the training from the testing sets, the more difficult the verification task. However, the seeing the limited quantity of data and quite simple architecture that is used, the final f1-score evaluated in the conditions that are the closest to the reality remains quite good.

Training and Testing sets	**f1-score**
1) sharing databases and people in common	92.03 %
2) sharing only database in common	87.68 %
3) not sharing anything in common	77.44 %

Table 6.1: Performance depending on the nature of the testing set, with respect to the training set

Is it better to have a large or a deep database ?

Regarding the data quantity, an interesting question that can be raised is if it's more helpful to increase the width or the depth of the database, where the width refers to the number of different people and the depth to the number of pictures per person. The motivation behind this question is that it can help relevantly setting the parameters of the data augmentation process able to generate:

- synthetic people (i.e increasing the width of the database)
- synthetic instances related to real people (i.e increasing the depth of the database)

Intuitively, both are interesting since a wide database allows to better capture inter-class variation while a deep database learns better intra-class variations are captured, making the CNN model able to learn more robust features.

To answer this question, 2 datasets $\mathcal{A}$ and $\mathcal{B}$ are defined, both containing 5000 pictures. The difference is that the dataset $\mathcal{A}$ is composed of 1000 different people with 5 corresponding pictures while the second one consists in 500 different people with 10 corresponding pictures. Once those 2 datasets have been defined, 2 models $\mathcal{M}_A$ and $\mathcal{M}_B$ are respectively trained on and the final performance of each of those models is given through Figure 6.1. It can be observed that training a model based on a deeper dataset seems to be slightly preferred, leading in this setting to a f1-score 5% higher compared to the case where a wider dataset supports the training phase.

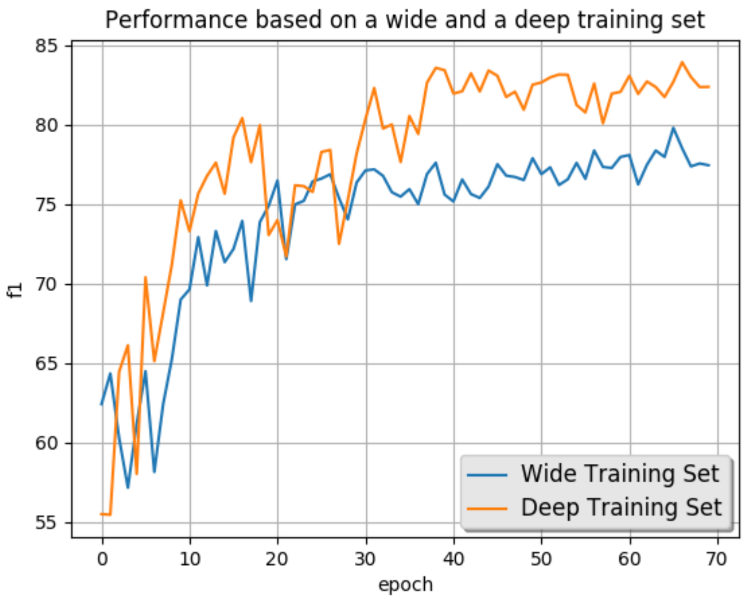


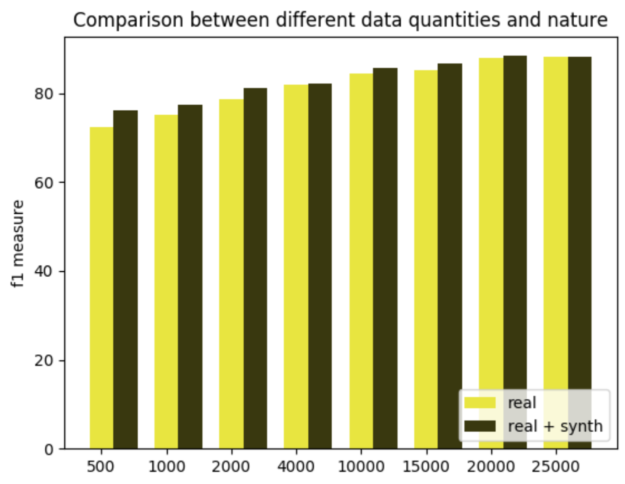
Figure 6.1: f1 measure evaluated from a model trained on a wide and a deep dataset

Data Quantity and Synthetic Data Exploitation

A second interesting aspect to investigate is how significant the addition of synthetic data into the dataset can be. After having noticed that a deeper training set could be more relevant than a wider one and seeing that the limitation regarding the data are more on the instances quantity rather than the people quantity, the focus is only on the synthetic instances generation (meaning that no synthetic people are defined).

To evaluate the impact of those additional synthetic instances, they are first introduced into the training dataset. Then, 2 scenarios are compared. A first one where the Siamese Network is trained based on the dataset $\mathcal{D}_{real}$ derived from data where people are represented by **only 2 instances**. And a second one where the training is supported by the dataset $\mathcal{D}_{real+s}$, where this last dataset exactly contains the same people's instances, **augmented by 4 synthetic instances** of those existing people. The resulting f1-scores related to those 2 scenarios are illustrated through Figure 6.2. This figure also allows to discuss the performance that can be obtained according to the initial number of images feeding the training set. Moreover, notice that the validation set is only composed of real data and the databases feeding those training sets are the all ones given in Table 5.1, except the *CASIA-WebFace*, since training sets based on larger face images sets are experimented here.

From Figure 6.2, it can be observed that the data augmentation process bring a slight improvement, mainly when few face images are available at training time. From 4000 pictures, the exploitation of synthetic data during the training phase doesn't really increase the performance anymore. Besides this, regarding the quantity of data, it's interesting to mention that the deep network benefits from the increasing size of the training set until about 15000 images and saturates thereafter, when evaluated on the verification task. Finally, from the Table on the right, it's interesting to notice that a small number of initial pictures can nevertheless lead to a sufficient triplets quantity, as theoretically developed in the previous chapter. That's why, at least partially, the increase of the f1-score according to the size of the face images



# of real pictures	# of real triplets	# Real and synthetic triplets
500	1766	3432
1000	2766	5176
2000	4766	8670
4000	8766	15736
10000	14460	21615
15000	19203	26366
20000	23946	31107
25000	26144	34129

Figure 6.2: f1-score according to the quantity of real (and synthetic) images ; Mapping between the number of pictures and the number of triplets

set supporting the training set is not huge. Another possible explanation is the quite simple architecture of the Siamese Network that is less suitable to deal with a very large quantity of data.

Another very interesting scenario that has been experimented is to train a model based only on synthetic data to see if they can be relevant enough to learn from. This last scenario was explored based on 3435 synthetic pictures. That lead to a quite good model reaching a f1-score of 76%.

Besides exploiting the synthetic images during the training phase, those can be used to perform the classification task, by augmenting the probe. This scenario is experimented in Section 6.2.

6.1.4 Network architecture

Now the data have been discussed, let's move to the description of the Siamese Network itself. First, a comparison of the different architectures using different losses is exposed Through Figure 6.3, where the f1-score is the highest one that was obtained over the 100 epochs supporting the training phase (and the quantity of data is the default one, given in subsection 6.1.2).

Seeing those last results and here in a context where the target is leading a research work on one-shot learning techniques (and not implementing an industrial tool reaching the highest performance possible), the basic architecture is employed and the training is directed by the triplet loss. This last choice regarding the loss can appear as inconsistent seeing Figure 6.3 that shows that the basic architecture performs better when training is directed by the cross entropy loss. However, let's remind that the performance evaluated here are on the verification task, while the final target is achieving good result on the face recognition task. That's why the triplet loss is selected, since it will appear later as providing better embeddings to support the second phase where faces are compared and matched based on their feature representation.

Finally, notice that Figure 6.3 is exposed to get only a global idea about performance that could be reached with other architectures and losses. Some of the 9 scenarios (the ones related to

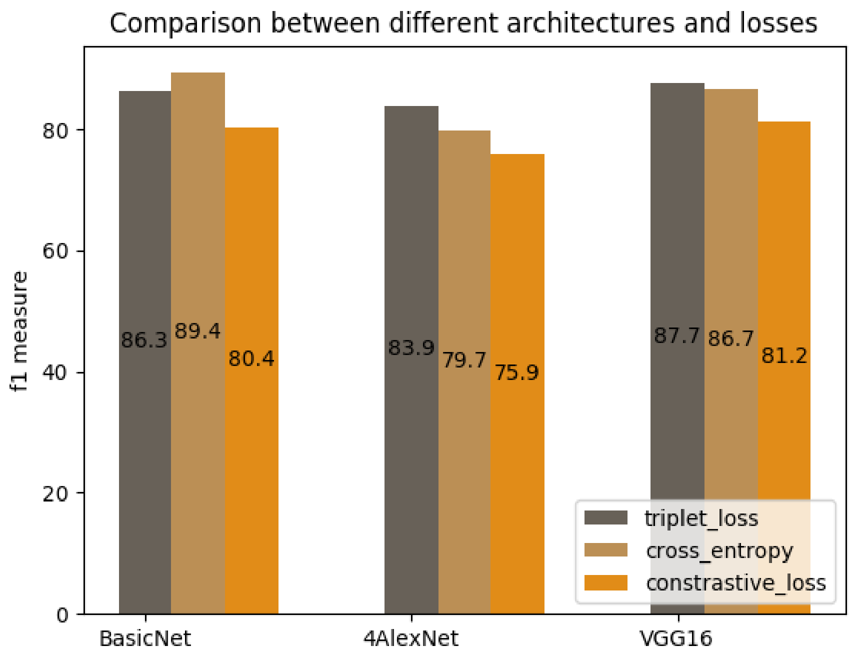


Figure 6.3: f1-score related to the face verification task, for different architectures and losses

the 4AlexNet and VGG16 Net typically) were only broadly optimized. Typically for instance, a more complex architecture may require a higher learning rate. Or in the same way, the number of final features should be higher or lower when using some specific loss with some specific architecture. This kind of consideration wasn't deeply investigated. Moreover, exploiting very complex architectures where a limited quantity of data is available is potentially less relevant.

6.1.5 Loss Function

Here, the focus is on the triplet loss function tested in different conditions to try to improve the learning process. On top of Table 6.2, the performance of the best scenario is illustrated, simply consisting in employing the triplet loss where the distances are derived from the features that have been normalized just before.

Regarding the best epoch piece of information, it can be relevant to get an idea about the learning process itself. Typically, here, a quite low best epoch shows how poor the learning process was.

It was also tried to be combined with the cross entropy loss function to see if those 2 losses couldn't be complementary. From Table 6.2, it can be observed that there's no improvement brought by this combination.

Besides this, the performance of the triplet loss defined from features that are not normalized is illustrated as well to highlight how much the normalization can positively impact on the performance. It's also interesting to say that this improvement can be observed only when using the basic architecture. A quantitative view of this improvement is illustrated in Table 6.2 (where all scenarios were experimented in the same hyperparameters setting). Nevertheless, notice that the performance illustrated here reflects both the quality of the final feature representation and the quality of the strategy to derive the threshold supporting the verification task. Potentially, this low f1-score may mainly arise from an improper threshold definition.

Finally, another experimented procedure is considered. It consists in, once the feature element-wise distance is computed, learning weights related to each element-wise distance and finally derive the distance used in the triplet loss (or, in other words, adding a final linear layer). Unfortunately, no improvement could be brought when including a last linear layer.

Loss	**f1 score**	**Best Epoch**
Triplet Loss	87.68%	64
Triplet Loss + Cross Entropy	81.52%	25
Triplet Loss without feature normalization	59.59%	6
Triplet Loss + Distance followed by a Linear Layer	80.99%	22

Table 6.2: Performance using the triplet loss in different conditions

6.1.6 Autoencoder Use

Finally, let's have a look at how helpful the pretraining phase can be. When the quantity of training images is very reduced (500 images typically), the model pretrained through an autoencoder definitely performs better. However, when meeting the general conditions stated at the beginning of this chapter, similar final performances are obtained when evaluating both the nonpretrained and the pretrained model. This is quantitatively illustrated through Table 6.3.

Training Procedure	**8000 images**	**500 images**
Not Pretrained	86.68 %	68.78 %
Pretrained	86.71 %	73.72 %

Table 6.3: Performance comparison when using an autoencoder to pretrained the embedding network

Besides this, when shifting the attention to the evolution of the loss and the f1-score on the validation set during training, it can be observed that the pretrained model seems better only at the beginning of the training process, which makes sense since it has already acquired some knowledge about how to represent a face while the other model starts from scratch. This is illustrated through Figures 6.4 and 6.5.

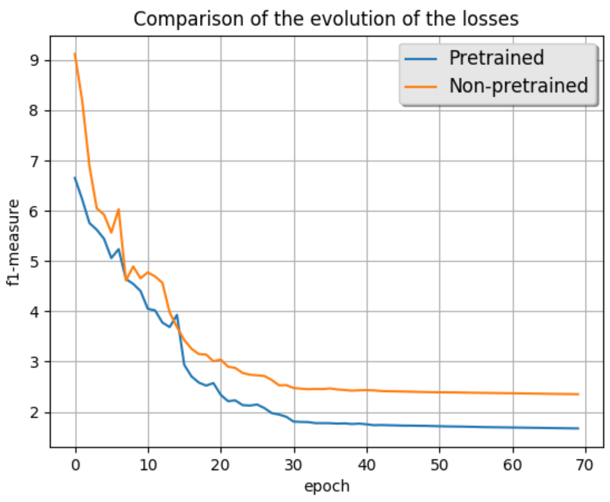


Figure 6.4: Loss Evolution with and without pretraining, on the validation set

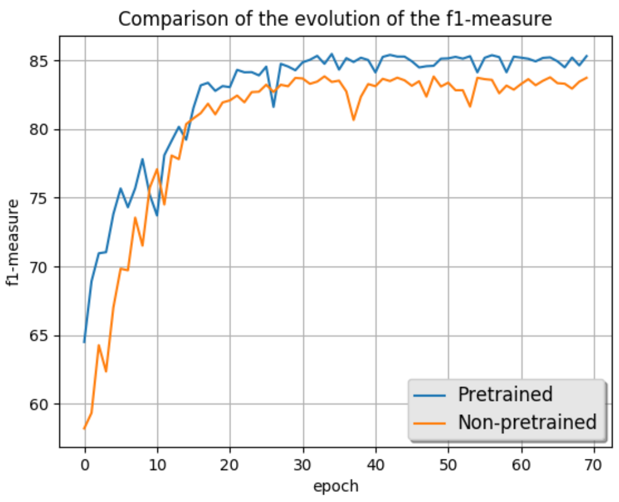


Figure 6.5: f1-score Evolution with and without pretraining, on the validation set

6.2 Classification Evaluation

Now the Siamese Network, responsible for deriving the best possible embedding, has been deeply investigated and tested over different scenarios, let's consider the last classification phase. This last section is quite important since it reflects the performance of the final module whose role is leading the face recognition task itself. Of course, that achievement is supported (and then affected) by the modules implemented before, composing the face recognition system.

6.2.1 Basic Conditions

To handle all the experiments on the classification task, the ***Face Scrub Cropped***, the ***cfp*** and part of the ***CASIA-WebFace*** databases are used since their data were not exploited to train the Siamese Network $\mathcal{S}$ that supports the classification task. In this way, those experimental conditions are closer to the reality.

This Siamese Network selected here is a **Basic Net** that was trained based on a **triplet loss**, with features normalization at the end and without any final linear layer after the feature distance computation. The performance of this model on the face verification task is 87% once evaluated on a testing set based on another identities than the ones that were used for training.

As briefly mention before, it could be noticed that the best performance on the verification task that were evaluated over the Siamese Networks that were trained was reached when employing the cross entropy loss. However, on the face recognition task, better accuracies were obtained when using the distance-based ranking approach rather than the voting system, which doesn't fit well with a cross-entropy based model, as illustrated in Table 6.4.

This last table first shows how relevant the triplet loss is for the second phase relying on people distance-based ranking and second how the distance-based ranking system performs better compared to the voting system that is "less nuanced" since it relies on a boolean prediction rather than a quantitative value.

Noticing this, despite slightly lower performance on the verification task for a triplet loss based model, this last type of model is selected and exploited over a distance-based ranking approach to lead the classification task.

	Voting System	**Distance-based System**
Cross-entropy based Model	$[47, 53]$ %	$[27, 32]$ %
Triplet-loss based Model	$[44, 49]$ %	$[58, 64]$ %

Table 6.4: 1-Top accuracy on a gallery made up of 50 people and a set of probes composed of 40 pictures

Besides this, regarding Table 6.4, the quite low results can appear as appealing. In front of this, let's highlight the fact that the evaluation metric employed here is the strictest one, considering a single final prediction, which is hard for this current system. For later, other evaluations metrics are employed.

Finally, regarding the parameters related to the classification task, in the cases where those parameters are not discussed of course, the gallery is imposed to be balanced so that each person has **8 picture instances**. The **size of the gallery is set to 50 and the number of probes is 40**. Moreover, each scenario is **repeated 7 times**, with different galleries and probes, to derive a more reliable final performance result and to get an idea about the gap from one test to another, in the same conditions.

6.2.2 Distance Ranking Approach

In the particular case where the classification task is based on a model whose training was directed by a distance-based loss, it's interesting to notice that in both distance ranking and voting approaches, the embeddings of the 2 face images are computed and the distance between them is derived. The difference is that in the distance ranking approach, this distance itself supports the final ranking while in the voting system, the elements of the distance vector are averaged and the result is compared to the threshold t that was computed at the end of the training phase to output a boolean answer. Knowing this, it can be guessed that there's no advantage in using the voting system. Indeed, the threshold isn't helpful here in a face identification task since it just leads to loose information, turning a distance into a boolean value resulting from a comparison. So directly using the distance is better.

6.2.3 Evaluation Metrics

Various approaches can be used to evaluate the performance over the classification task:

- **The Top-1 accuracy-based** where only the identity on top of the ranking resulting from the comparisons $(\mathcal{P}, \mathcal{G}_{ij})$ is predicted and valuable.
- **The Top-N accuracy-based** where the N first identities on top of the ranking are predicted and checked so that the final predicted identity is marked as correct if the expected identity is among the N top ones.
- A variant of the Top-N accuracy-based metric (the **"Top-Multiple accuracy-based"**), where 2 differences can be highlighted compared to Top-N one:
  - The N isn't a priori fixed, it varies from one probe to another **according to the certainty of the ranking**, evaluated based on a threshold. In particular, in the voting approach, it may occur that K identities have the same score. Similarly, in the

distance ranking approach, it may occur that the K smallest distances are very close, closer than some threshold[2]. Then, N is set to K.

– When computing the performance itself over different probes, first the correctness of the multi-prediction is checked. If it's correct, the performance is increased by $\frac{1}{N}$ (rather than 1 as in the 2 previous cases), so that the final performance reflects the uncertainty of the algorithm.

The advantage of this last evaluation approach is that it brings more nuances compared to the 2 first ones where the Top-1 accuracy-based may be too "strict" in some cases and the "Top-N accuracy-based" may be to "broad" in some other cases where the gap between the top identity and the other is significant. If there's no specification, this last metric is used to evaluation the algorithm performance.

At this step, notice that the Equal Error Rate isn't finally used to evaluate the system since finally the boolean approach isn't employed at all.

6.2.4 Gallery Size

Now the conditions for the assessment of the face recognition system have been described, let's have a deeper look at the performance with respect to some parameters.

First, Figure 6.6 exposes the performance that can be reached for different gallery sizes. To support this Figure, 2 scenarios were considered. A first one where the probe has a single corresponding instance and a second one where there are 2 instances representing the probe, so that more comparison are performed and better result can be obtained. Of course, in practice, getting 2 instances per probe isn't always affordable.

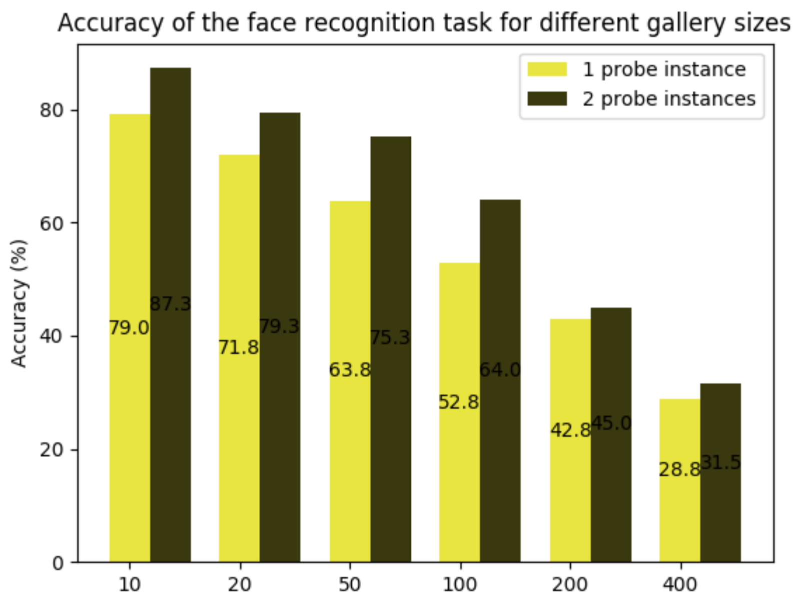


Figure 6.6: Top-multiple accuracy of the face identification over galleries of different sizes

[2]This threshold is set to a proportion (0.3 typically) of the average of the gaps from one distance to another. Imagine, for a given probe, you get {Luc:1, Paul:1.1, Lise:3, Sarah:7}, the average gap is 1.6, leading to a threshold of 0.5. Seeing this, $K = 2$ and Luc and Paul are predicted.

From Figure 6.6, as expected, it can be noticed that a second instance is very helpful, especially for low gallery, increasing the accuracy from 3 to 12%. Besides this, regarding the accuracy values themselves, they quickly drop when increasing the gallery size, encouraging to consider the set of the Top-N identities as final prediction. From that point, the second Top-N metric is used for galleries containing various number of people and the probe having a single instance. The results are illustrated through Table 6.5. For each computed accuracy, their standard deviation is mentioned to get an idea about the reliability of the those result.

Gallery Size	**Top-1**	**Top-3**	**Top-5**	**Top-10**	**Top-20**
20	68 ±3.3%	86±4.4%	95±0%	100±0%	100±0%
50	60±2.5%	76±4.3%	84±2.5%	97±2.4%	100±0%
100	48±4%	66±1.9%	75±3.8%	90±3%	94±3.3%
200	36±2.6%	64±2.8%	69±3.8%	84±1.2%	93±1.4%
400	24±2.5%	40±2.5%	50±3.3%	65±3.3%	87±1.4%

Table 6.5: Top-N accuracies for different gallery sizes

From this last Table, much better results are depicted, compared to the previous ones. Typically, reaching a top-10 accuracy for a gallery size of 200 when a limited quantity of data has supported the training phase is acceptable. Of course, those results could be improved by going back to the first phase to improve the feature representation.

6.2.5 Data Augmentation with synthetic instances

Finally, as briefly mentioned before, besides exploiting synthetic instances to augment the training set, synthetic data can also be used during this second phase to augment the probe and, in this way, reinforce the matching process. A similar strategy has been experimented and illustrated in Figure 6.6, except that in that case only one real instance has been added. Here, 3 additional synthetic instances represent the probe and the resulting performance are illustrated through Figure 6.7.

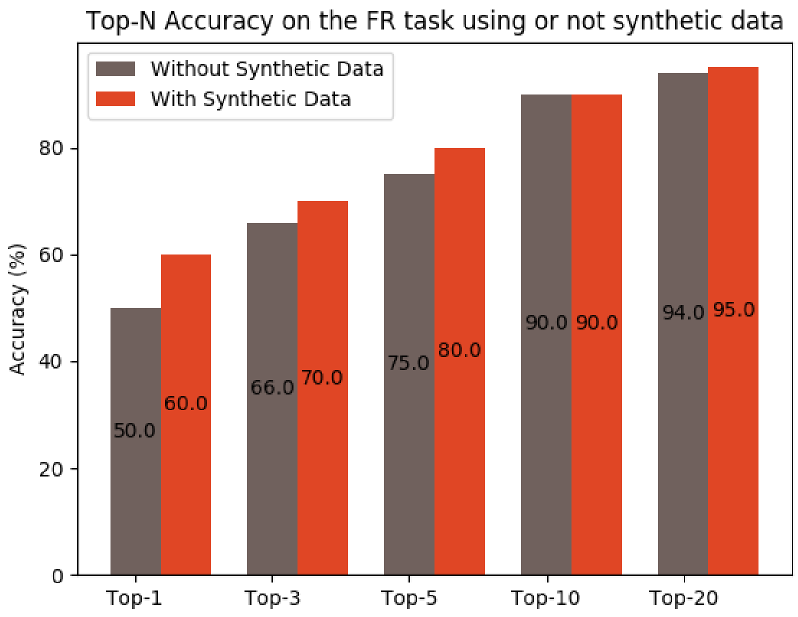


Figure 6.7: Comparison of the Top-N accuracies when using or not synthetic data, evaluated on a gallery made up of 100 people

From Figure 6.7, it can be observed that a slight improvement is brought by using synthetic images to augment the probe. However, this improvement is still less significant than the ones shown in Figure 6.6 when real instances augment the probe. This can be explained by the current limited quality of the synthetic images that really could be improved in the future.

6.2.6 Style GAN Encoding Exploitation

In addition to this, the distance derived from the encodings z is also tested to be used to support the ranking since it's expected to represent, at least partially, the physionomic face characteristics.

The resulting accuracy was around 37%, showing that, in some cases, the encoding z could be relevant enough. However, the performance remain pretty bad, which makes sense since the training process of the perceptual model deriving z wasn't optimized to perform well on the face identity differentiation.

Chapter 7

Retrospective

Before concluding, some words are dedicated to put in perspective all that work, highlighting the positive aspects, the difficulties that have been encountered, the weaknesses of the current solution and some extra ideas that could be explored.

7.1 Helpful and Positive Points

First, let's go through the main positive points related to this work:

- Concerning the subject whose start point was quite broad ("research on deep learning"), an interesting and concrete application was selected to support this research work.
- Over all this work, there was always lots of freedom regarding the ways to explore and techniques to investigate. Moreover, the face recognition problem is quite famous and has been tackled over many works, offering a wide investigation area and possible documentations.
- Seeing that last point, it led to some diversity in the solution implementation. Many different modules had to be designed over this work to tackle particular subtasks. It includes the data processing, the network architecture optimization, the classification part, the experimentation phase based on scenarios definition, the synthetic data generation...
- A final good global solution framework has been designed.
- That work was an experience in the research domain.

7.2 Difficulties and Points to Improve

Besides those positive points, let's go now over the difficulties and all the points which could have been improved:

- Related to that freedom mentioned before, the reverse side is that the clarification of the objectives wasn't straightforward. This phase was very progressive, and objectives were (permanently) slightly adapted, namely in terms of performance expectation, seeing how the effective work was going.

  Typically, the initial target was designing a model competing with human performance (but still based on few training data). That appeared as quite ambitious. Also, the minimum affordable quantity of training face images was very difficult to estimate at the very beginning of this work, without any experimentation phase.

Moreover, new ideas were coming on way, bringing additional objectives (like the exploitation of synthetic data typically).

- The huge quantity of (hyper)parameters and implementation choices can be also pointed here, leading to face a very large number of possible scenarios. To deal with that, a very good organization was required to keep track of all the parameter settings and the training sets corresponding to each scenario. Related to that aspect, as briefly mentioned before, the optimization process was focused around the BasicNet architecture, meaning that no deep optimization process was led when using the AlexNet or the VGG16 Net, which could nevertheless have been relevant in the perspective of reaching higher performance.
- Regarding the performance of the overall system, despite a long experimentation phase, it remains limited for large galleries and could still be improved, by getting a better feature representation basically. The easiest way would be to increase the quantity of training data and using more complex network architectures but it would lead outside the boundaries of the one-shot learning setting. In this setting, more investigation on the exploitation of synthetic data would be the best approach.
- From an implementation point of view, the solution had to be implemented so that it could ideally work running on both cpu and gpu to be tested locally and then on the machines of the University.

  Facing that, first, regarding the use of Pytorch exploiting cuda, some issues had to be overcome.

  Also, many versions incompatibilities were encountered, consuming time to solve that. The fact of working on several machines (the local one and the ones of the university) was sometimes quite tedious.

7.3 Future Work

Finally, this whole current solution could certainly be improved by cleanly completing the implementation of the "Adversarial Siamese Network" (illustrated through 5.16) relying on synthetic data generation.

Besides this some other points could have been relevant to investigate are listed here:

- The face recognition task could have been complexified by using pictures with lateral positions for instance or with background including more disturbing elements.
- Despite the "In the Wild" characteristic claimed in some datasets exploited here and the effort provided to maximize the diversity over the datasets supporting this work, those remain quite specific, taken almost exclusively by professional photographers in well lit environments. It could be very interesting to submit that current algorithm to very usual pictures taken in a crude way.
- The face feature representation still could be improved in the perspective of the face recognition task by:
  - Experimenting different cuts in the network and exploiting the resulting embedding representation
  - Experimenting other architectures

- Other methods to classify (i.e. recognize) a face could be studied, some relying on Support Vector Machines or decision trees taking as input feature vector, typically.
- Again regarding the classification phase, it could be speed up and maybe improved in terms of performance by computing the average (or the median) over the feature representations related to each pictures j attached to each identity i in the gallery. In this way, each identity in the gallery would be represented through a single embedding (expected as generalizing at best the current person) and for each probe, only i comparison are performed.
- More investigation on the synthetic data generation could lead to better final performance, as mentioned before. To do that, a more sophisticate method could be designed to find new other latent direction $\mathbf{w}$ leading to consistent synthetic faces (i.e. where all the physionomic features are preserved and the external ones altered). Typically, exploiting tagged data can be very helpful for this purpose.
- It could be interesting to apply all that procedure to another task, like symbols recognition typically and compare the performance that could be obtained.

Chapter 8

Conclusion

To conclude, this work proposed a complete strategy to perform the face recognition task by exploiting one-shot learning techniques so that good final performance can be reached while using a limited quantity of data. More precisely, 3 sequential steps were defined. The first one referred to the data management, including the data collection, the data processing and the triplet definition, mainly. Next, deep convolutional Siamese Neural Networks were learnt and evaluated on the verification task. The results related to that task were discussed in order to finally select the best model for the last phase. During this last phase, the aim was to correctly identify a probe, based on a gallery containing labeled instances. To do so, this probe has to be compared to each gallery instance by computing the distance between their embeddings derived from the Siamese Network. Following that, all the identities in the gallery are ranked according to that distance and the predicted identity(ies) is/are the top one(s).

Regarding the performance that were obtained, first, the Siamese Network reached a f1-score of 87% on the verification task when trained based on 8104 face images and directed by a triplet loss function, with a basic architecture detailed in Table 5.2. Over this work and in this particular setting, relying on a more complex architecture didn't outperform much the current result, at least when only a limited quantity of training data is exploited. In the same way, no significant improvement could be brought when increasing a lot the size of the training set while remaining working with a quite simple architecture.

Next, the classification could reach good results when many identities could be predicted. In particular, for a gallery of 200 people, a top-10 accuracy of 84% could be obtained. However, for larger gallery, containing 400 identities for instance, that accuracy dropped by 20%.

To try to help increasing those last results, some extra modules were being defined. The first one implements an autoencoder used to initialize the weights of the Siamese Network. It looked relevant especially when few triplets were available (based only on 200 people each with 8 face instances typically) or when the number of epochs over which the Siamese Network was trained was limited. The second extra module is the one defining synthetic face instances to raise the training set and the probe. When increasing the training set, as before, it appeared useful only when little training data was available. However, when used to reinforce the probe during the last phase, the performance of the classification could be increased by 4% in the setting described above.

Finally, it's important to mention that many ways can still be explored, especially regarding the data augmentation process based on synthetic data generation and the iterative improvement process, where the main idea is retraining the Siamese Network based on synthetic images that currently can't be captured appropriately (referring to the Adversarial Siamese Network).

Bibliography

[1] M. Wang and W. Deng, “Deep face recognition: A survey,” September 2018.

[2] “Solution implementation.” `https://github.com/gbrieven16/oneShotLearning`.

[3] C. Finn, P. Abbeel, and S. Levine, “Model agnostic meta learning for fast adaption of deep network,” July 2017.

[4] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel, “Meta-learning for semi-supervised few-shot classification,” March 2018.

[5] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, “Matching networks for one shot learning,” December 2017.

[6] Q. Cai, Y. Pan, T. Yao, C. Yan, and T. Mei, “Memory matching networks for one-shot image recognition,” April 2018.

[7] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, “Sampling matters in deep embedding learning,” January 2018.

[8] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” 2015.

[9] A. Ng, “Face recognition neural style transfer (coursera.org),” 2017.

[10] G. Keren1, M. Schmitt1, T. Kehrenberg1, and B. Schuller, “Weakly-supervised one-shot detection with attention similarity networks,” June 2018.

[11] A. Puzanov and K. Cohen, “Deep reinforcement one-shot learning for artificially intelligent classification systems,” August 2018.

[12] C. Finn and M. Woodward, “Active one-shot learning,” February 2017.

[13] L. Chi, H. Zhang, and M. Chen, “End-to-end spatial transform face detection and recognition,” 2017.

[14] “Psychological image collection at stirling (pics).” `http://pics.psych.stir.ac.uk/2D_face_sets.htm`.

[15] “Georgia tech face database.” `http://www.anefian.com/research/face_reco.htm`, 1999.

[16] “Yale face database.” `http://vision.ucsd.edu/content/yale-face-database`.

[17] “Faces94.” `https://cswww.essex.ac.uk/mv/allfaces/faces94.html`.

[18] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments.” http://vis-www.cs.umass.edu/lfw/, October 2007.

[19] S. Sengupta, J. Cheng, C. Castillo, V. Patel, R. Chellappa, and D. Jacobs, *Frontal to Profile Face Verification in the Wild*, http://cslipublications.stanford.edu/HPSG/9/ February 2016.

[20] “Casia web face database.” https://drive.google.com/file/d/1Of_EVz-yHV7QVWQGihYfvtny9Ne8qXVz/view.

[21] “Stylegan implementation.” https://github.com/Puzer/stylegan-encoder/.

[22] “Siamese network implementation.” https://becominghuman.ai/siamese-networks-algorithm-applications-and-pytorch-implementation-4ffa3304c18.