Explaining Siamese Neural Architectures from Local Perturbations on the Embedding Layer: Methodological Proposal and Experimental Validation on Facial Recognition

Pablo Costa Contreras

Tutorizado por:

Dra. Marilyn Bello García

Dr. Pablo Mesejo Santiago

Memoria de Trabajo Final de Máster.

Máster Universitario en Física y Matemáticas.

Año académico 2021-2022

Universidad de Granada.

DECLARACIÓN DE ORIGINALIDAD

Don Pablo Costa Contreras con NIE Y4924608T garantiza al firmar este documento que en la realización del TFM que lleva por título “Explaining Siamese Neural Architectures from Local Perturbations on the Embedding Layer: Methodological Proposal and Experimental Validation on Facial Recognition ” se han respetado todos los derechos de otros autores a ser citados cuando se han utilizado sus resultados o publicaciones.

Acknowledgements

I would like to thank Marilyn and Pablo for believing in me although I did not have a formal background on the topic when beginning this project. Without their valuable comments and support, this work would not have been possible.

Abstract

Despite the growing interest in the explainability of deep neural networks during the last years, there is little work on the topic that focuses on siamese network architectures. The existing studies either lack proposals for applying it to image recognition, where this model has its most important applications, or do not consider any method to assess the explanations' quality objectively. In this work, we propose a method that consists in locally perturbing the feature space of the embedding layer of the network to recognize what areas of the images being compared are the most important in determining the inferences of the model. We test this method on a network trained to recognize if two images of faces belong to the same person. Then, we modify a method employed in deep neural networks to assess the quality of explanations, making it applicable to siamese network architectures. Compared to existing explainability methods on image recognition, the advantages of our proposal are that it is simpler, depends on fewer parameters, requires less data for a single explanation, can be applied to a wider variety of cases, and provide explanations of better quality. This suggests further works for testing our proposal in different siamese neural network models and kinds of datasets, and thoroughly compare its performance with future explainability methods for siamese network architectures.

Contents

1 Introduction **1**

1.1 Problem description 1

1.2 Motivation 2

1.3 Goals 3

1.4 Structure of the Report 3

2 Theoretical Background **4**

2.1 Deep Neural Networks 4

2.1.1 Convolutional neural networks 5

2.1.2 Siamese neural networks (SNN) 7

2.1.3 Loss functions to train SNN 9

2.2 Explainability methods in artificial intelligence 12

2.2.1 Categorization 13

2.2.2 Evaluating the quality of visual explanations 13

3 State of the Art **15**

3.1 Explainability of Siamese Networks 15

3.1.1 An Explanation Method for Siamese Neural Networks (EMSNN) 15

3.1.2 Self-learn to Explain Siamese Networks Robustly 18

3.2 Limitations of existing methods 19

4 Proposed Method **21**

4.1 Explaining from Local Perturbations on the Embedding Layer 21

4.1.1 Step 1: Training the decoder to reconstruct images from embedding vectors 22

4.1.2 Step 2: Perturbing the embedding features according to their relevance . 22

4.1.3 Step 3: Inferring the most important image features 24

4.2 Assessment of the quality of explanations 25

4.3 Improvements over existing methods 26

5 Experiments **27**

5.1 Siamese network implementation 27

5.1.1 Architecture 27

5.1.2 Training and Testing 29

5.2 Decoder implementation 30

5.2.1 Architecture 30

5.2.2 Training and Testing 30

5.3 Results and Discussion 31

5.4 Evaluating the quality of the explanations 34

5.5 Comparison with EMSNN 36

5.5.1 Reconstruction Network 36

5.5.2 Use of a prototype 37

5.5.3 Randomly perturbing the features by selecting the k-top of the ranking . 38

6 Conclusions **40**

6.1 Future works 41

List of Figures

2.1 Structure of a single layer perceptron. 5
2.2 Structure of a multilayer perceptron with four hidden layers. 5
2.3 Example of a convolution. 6
2.4 Example of pooling. 7
2.5 Full structure of a CNN 7
2.6 A simple structure of an SNN architecture 8
2.7 SNN with a convolutional architecture for image processing. 9
2.8 Triplet loss training. 10
2.9 Number of publications from 2000 to 2021 related to the query "Explainable AND Artificial AND Intelligence". A total of 1857 publications are displayed. Search performed using Scopus on the 15th of August, 2022. 12
2.10 Relative decrease of quality on each iteration of the assessment method for LRP and SA [1]. 14

3.1 Training of the autoencoder, step 1. 16
3.2 Training of the autoencoder, step 2. 17
3.3 Examples of explanations by the method given by Utkin et al. 17
3.4 Application of the method on a SNN that determines how similar are two different clients of a bank. 18

4.1 Step 1 of the explainability method 22
4.2 Step 2 of the explainability method. 23
4.3 Step 3 of the explainability method. 24
4.4 Method for assessing the quality of explanations. 25

5.1 Original Inception-ResNet-v1 architecture. 28
5.2 Detailed structure of the Stem of the Inception-ResNet-v1 network. 28
5.3 Detailed structure of Inception-ResNet-A module in Figure 5.1.1. 28

5.4 Detailed structure of Reduction-A module in Figure 5.1.1. 28
5.5 Detailed structure of Inception-ResNet-B module in Figure 5.1.1. 29
5.6 Detailed structure of Reduction-B module in Figure 5.1.1. 29
5.7 Detailed structure of Inception-ResNet-C module in Figure 5.1.1. 29
5.8 Some examples of the images used for implementing the method. 31
5.9 Original images and their corresponding reconstructions. 32
5.10 Results provided by the proposed method when the network recognizes two faces of the same person as similar. 33
5.11 Results provided by the proposed method when two faces of the different persons are recognized by the network as dissimilar. 34
5.12 Relative decrease in similarity in the images classified by the network as belonging to the same class. 35
5.13 Perturbation of superpixels in similar instances. 35
5.14 Average relative increase in similarity in the images classified by the network as belonging to a different class 36
5.15 Perturbation of superpixels in dissimilar instances 36
5.16 Experiments for assessing the application of a prototype as a reference. 37
5.17 Results provided by the proposed method when using a k-top feature ranking. . . 38
5.18 Comparison of the decrease in similarity score when perturbing by a relevance measure or by randomly perturbing just a number of the features ranked as most important 39

List of Tables

5.1 Decoder architecture. 30
5.2 *Autoencoder architecture* 37

Chapter 1

Introduction

1.1 Problem description

Siamese neural networks (SNN) [2] are a special type of deep neural network (DNN [3]) that allow comparing different instances by giving a measure of their distance. The particular structure of SNN, which consists of two identical neural networks joined at the output, confers them the ability to achieve this task [4]. Also, by defining a threshold, these networks can be used to classify the compared instances as belonging to the same or different classes if their distance is below or above that value. This characteristic endows them with countless applications in several different areas such as signature verification (its first application [2]), face recognition [5], natural sciences, medicine [4], among others.

Despite these outstanding results, SNNs, as well as every other type of deep neural networks, are said to be black boxes, i.e., their strength in modeling complex interactions makes their internal dynamics almost impossible to explain [6]. In particular, it is not easy to intuitively and quantitatively understand the outcome of their inference, i.e., for a single input data point, what caused the trained neural model to arrive at a certain output. This is especially important in applications such as medicine, where model confidence must be guaranteed [7, 8]. Moreover, the strive for predicting accurately sometimes causes the network to infer spurious correlations caused by overfitting or implicit dataset bias, such as gender and race [9]. In this context, explainable artificial intelligence may be useful to unveil this behaviour [10–12].

Explainability of machine learning models has increasingly developed as a research field, and several methods have been proposed in the last years [6, 7]. This has given rise to so-called Explainable AI (XAI) [6, 13], i.e., systems with the ability to explain their reasons for making decisions. Several approaches have been proposed to understand and interpret neural network reasoning: some of them attempt to build more transparent models that help to understand the networks' internal mechanics. In contrast, others complement the neural network model with a post-hoc interpretation stage, conferring useful information that highlights underlying patterns and relevant aspects of the input when inferring [14].

As a direct benefit, explanations help humans to understand the trends underlying the dynamics of the network, making it easier to identify the correlations that a network established in its learning process (including the undesired ones). This might suggest new ways of working with neural networks, also providing information about the applicability of the results. In addi-

tion, a recent research by Rieger et al. [9] reports having used explanations from human domain knowledge to improve the performance of a deep learning model, enabling it to ignore spurious correlations, correct errors, and generalize to different types of dataset shifts, giving an objective measure of this effect. This also suggests the possibility of incorporating the explanations given by algorithmic methods to improve machine learning performance.

1.2 Motivation

Although the development of different explanation methods has flourished during the last years [7, 15], methods that focus specifically on SNN are scarce. This is a problem considering the multiple applications given to this model in several fields, and its outstanding performance in tasks such as face recognition and one-shot image recognition (correctly making predictions given only a single training data instance) [4, 5, 16]. The only proposals that we could find in the scientific literature are the ones from Chen et al. [17] and Utkin et al. [18]. The first one is only suitable for graphs and tabular data, and describes a method to generate masks for the inputs that contain most of the information necessary for the network to attain the correct inference. The second one consists of constructing a prototype embedding for each existing category in the image set and deriving an explanation based on the similarity between the embedding vector of an image and the corresponding prototype.

Some of the limitations of the existing methods are:

- The authors only show examples of applications in relatively low-complexity data, such as tabular data [17] and images of handwritten digits [18] from the MNIST dataset [19]. However, SNN architectures have their most relevant applications in more complex datasets like those used for face recognition.
- The method proposed in [18] can only explain why two images are similar (i.e., belong to the same category) and not why they are dissimilar (i.e., belong to different categories).
- The method proposed in [18] requires a set of images of the same category to construct the prototype. This is a special problem in face recognition tasks [20] where it might be hard, if not even legally conflictive, to require several identified images of the same person for some applications of image recognition [21].
- The only method that works with images [18] does not provide an objective measure to evaluate the quality of explanations.
- The method in [18] depends on several parameters (7), which makes it hard to reproduce its results.

This motivates us to propose a new method that overcomes the limitations mentioned above. Besides, we believe that it is important to apply explainability methods in SNN on the data in which this architecture has its most relevant applications, as it is face recognition. Furthermore, we believe that it is crucial to provide an objective measure to evaluate the quality of explanations in siamese architectures, something that will also give a reference for comparing the different methods that might be proposed in the future for this particular model.

1.3 Goals

Considering the stated before, the objectives of this work are:

- Review the state of the art of explainability methods in SNN.
- Propose an explanation method for SNN that overcomes the discussed limitations of the methods available in the literature.
- Modify an existing method for assessing the quality of explanations to make it applicable in SNN architectures, using it to evaluate the quality of our proposal.
- Apply the proposed method to a SNN architecture trained in face recognition.
- Compare the results of our proposal with the ones obtained by following the method described in [18].

1.4 Structure of the Report

This report is divided into six chapters. The first chapter is this Introduction, in which the motivations and goals of our work are described. Chapter 2 explains the concepts needed to understand this report, which fall into the categories of neural network and explainability methods. In Chapter 3, the state of the art methods for explaining SNN inferences are described. Chapter 4 describes a new method for this task, whose results are shown and discussed in Chapter 5. Finally, in Chapter 6, the contributions of our proposed method are discussed, and future works that it could inspire are suggested.

Chapter 2

Theoretical Background

In this chapter, the core concepts for understanding this work are explained. It is divided into two main sections: the first one deals with deep neural networks, describing them in a general way and then the relevant architectures for the explainability method proposed in chapter 4, such as convolutional neural networks and SNN. The second part deals with the topic of explainable artificial intelligence describing its history, motivation, and assessment methods for the quality of explanations.

2.1 Deep Neural Networks

Artificial neural networks (ANNs) [3, 22, 23] are composed of a collection of connected units, nodes or artificial neurons that emulate biological neural systems. Most neural networks are *feedforward*, which means that neurons of one layer only pass information to neurons on subsequent layers. In this way, the first layer is directly connected with an input whose values process and pass to the following layer. Then, the neurons in this layer act similarly, sending their results to the next layer on and on until it reaches the final one, which then gives the output of the network [3].

The single-layer perceptron is the earliest and simplest type of feedforward neural network [24]. This model consists in taking the dot product of an input $x \in \mathbf{R}^N$ and a set of weights $w \in \mathbf{R}^N$ with a bias $b \in \mathbf{R}$ evaluating the value obtained according to an activation function. This basic process is illustrated in Figure 2.1.

The classic single-layer perceptron has limitations because it can only solve linearly separable problems [25]. The multilayer perceptron (MLP), that consists of multiple layers of neurons interacting essentially in the same way as the linear perceptron (depending on the weights connecting the neurons, and the activation function defined) due to its highest complexity can be used to approximate any continuous function [26]. Deep neural networks are a special kind of the MLP model [24]. The structure of the multilayer perceptron is depicted on the top of Figure 2.2.

The approximation of functions is attained by training. In this process, deep neural networks are supervised to fit a set of examples in which the correct output values are known. To do this, a loss function is defined that compares each value predicted by the network with the true

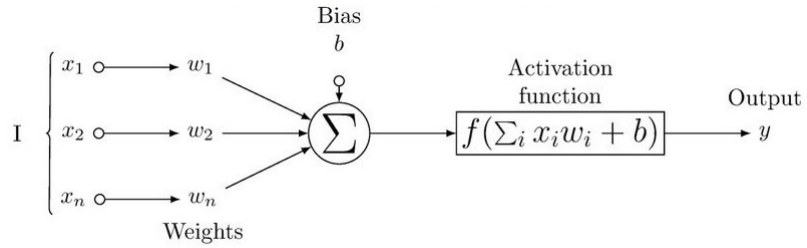

Figure 2.1: Structure of a single layer perceptron. The $x_1, x_2, ...x_n$ correspond to the input of the perceptron. These values are multiplied by the corresponding weight w_i, and the sum of the products with the addition of the bias b are passed to an activation function, that gives the output y of the perceptron. Extracted from [26]

.

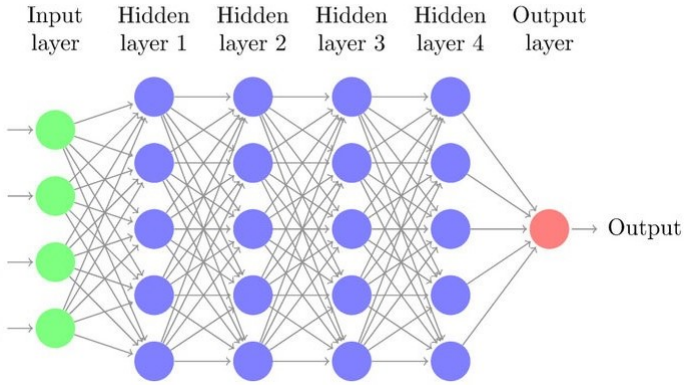

Figure 2.2: Structure of a multilayer perceptron with four hidden layers. Every neuron in these layers, represented by a blue node, acts in the same fashion that the linear perceptron depicted in Figure 2.1. Extracted from [26]

.

ones. In this way, the loss function measures how long the desired output is from the current prediction in terms of statistical error [3, 22]. The network uses this information to modify its internal structure (the magnitudes of the weight that connects each neuron) through an algorithm called *Backpropagation* [27]. This is used to compute the gradient of the loss function concerning the weights of the network to modify them later to minimize loss and increase the accuracy of predictions. In this calculation, to avoid redundancy, the algorithm computes by the chain rule the gradient one layer at a time, iterating backward. Once trained, the network can be used to predict the values of instances that were not known before.

2.1.1 Convolutional neural networks

Convolutional Neural Networks (CNN) are a special kind of deep neural network [28, 29]. They are very similar to classic neural networks in the sense that they consist on a set of neurons arranged in a feedforward fashion. The main difference is that a hidden layer neuron in a

CNN is only connected to a subset of neurons in the previous layer, something known as *sparse connectivity*. These layers are also called *convolutional layers* and act as filters, something that gives the network the ability to learn features implicitly [30, 31]. An example of a convolution is illustrated in Figure 2.3. The sequential organization of the convolutional layers also provide a hierarchical organization to the different feature extracted. For example, the trained filters of the first layer would probably capture a set of edges or color blobs of an image, a posterior layer is going to interpret these edges and blobs as shapes, and more advanced ones might recognize parts of an object [30].

Besides, convolutional layers, pooling layers are also common in CNN. The purpose of these layers is to reduce the spatial resolution of the feature maps and thus achieve spatial invariance to input distortions and translations, also known as subsampling [31]. This also makes the computational cost of the networks less expensive by diminishing the number of neurons in the next layer. In Figure 2.4 two common methods to pool are described.

In general, the last layer of a CNN is a fully connected layer, whose function is to interpret the feature representations and perform the function of high-level reasoning [31]. The full structure of a CNN is depicted in Figure 2.5 The ability to do feature extraction make this networks specially good for working with images, where nearby pixels are highly correlated, something that can be better captured by the sparse processing described above [30].

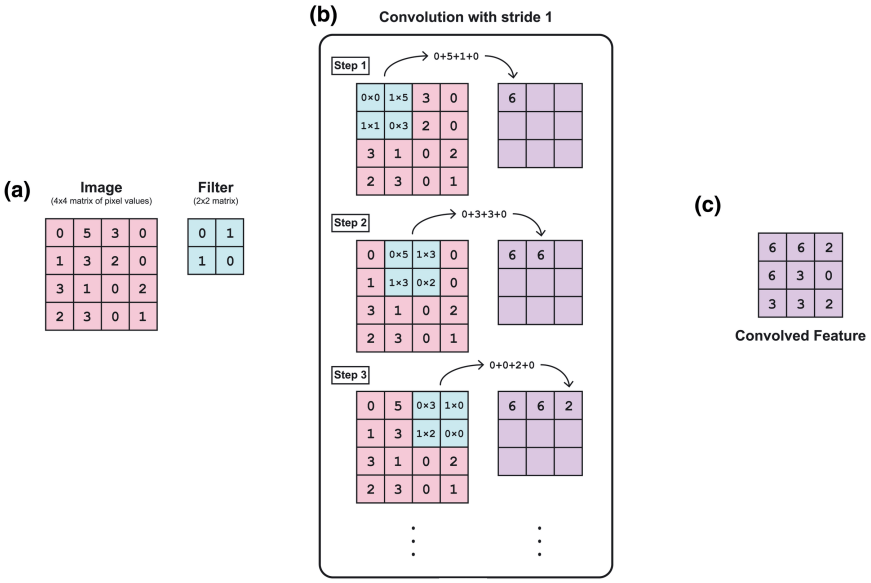

Figure 2.3: Example of a convolution. The inputs of the neurons are depicted in the pink grid, and the convolution (also called filter) in the light blue grid in (a). On each step, the weights on the filter multiply the corresponding input (b), and the sum of that values is the output shown in the convolved feature grid (c). Extracted from [32].

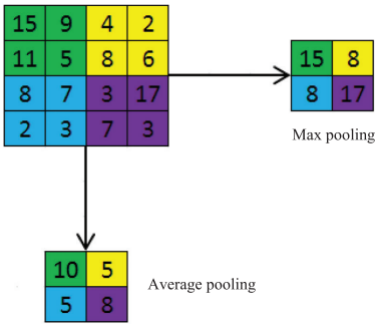

Figure 2.4: Example of pooling. When doing pooling, neighbourhoods of neurons are defined for being processed together. Average pooling outputs the average value of the neighbourhood, while max pooling only keeps the highest value. Extracted from [31].

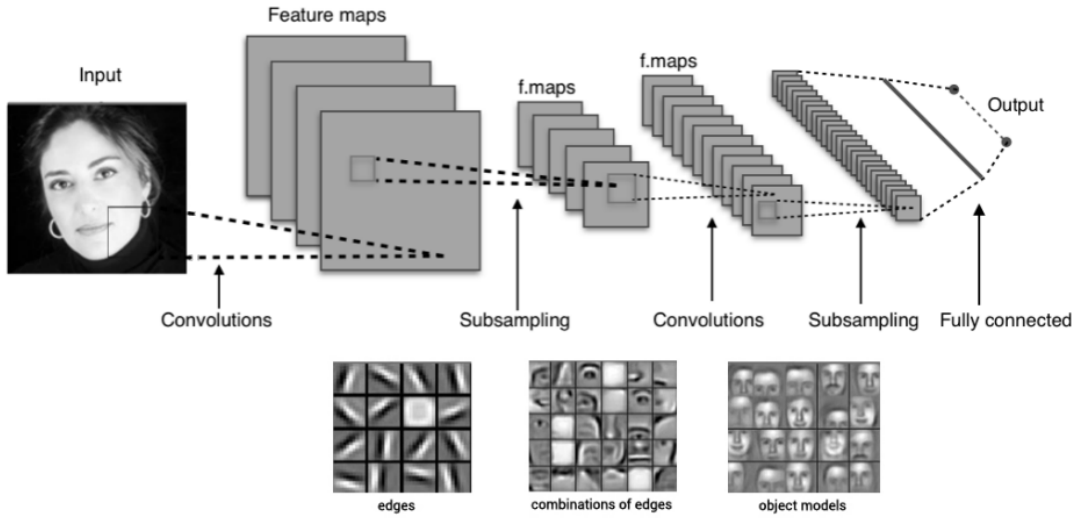

Figure 2.5: Full structure of a CNN. At the top, an image is given as input to the network. On each layer, several convolutions or pooling layers (subsampling) are applied until the final fully connected layer is reached, giving at last the output of the network. At the bottom, the expected features captured by each convolutional layer (or feature map) are depicted. The complexity and hierarchy of the features detected increase as the network gets deeper. Extracted from [33].

2.1.2 Siamese neural networks (SNN)

Being able to evaluate, by making a comparison, how similar or dissimilar are two objects is a tool with countless applications in many different areas of study [4]. To do this, it is necessary to have a metric for quantifying the proximity of two objects and that the aspects to contrast are consistent with our goal [4]. That is a problem when comparing images because a comparison can not work if it is done between objects containing different data types, meanings, or both. For example, comparing the pixel of two images would give information of poor applicability.

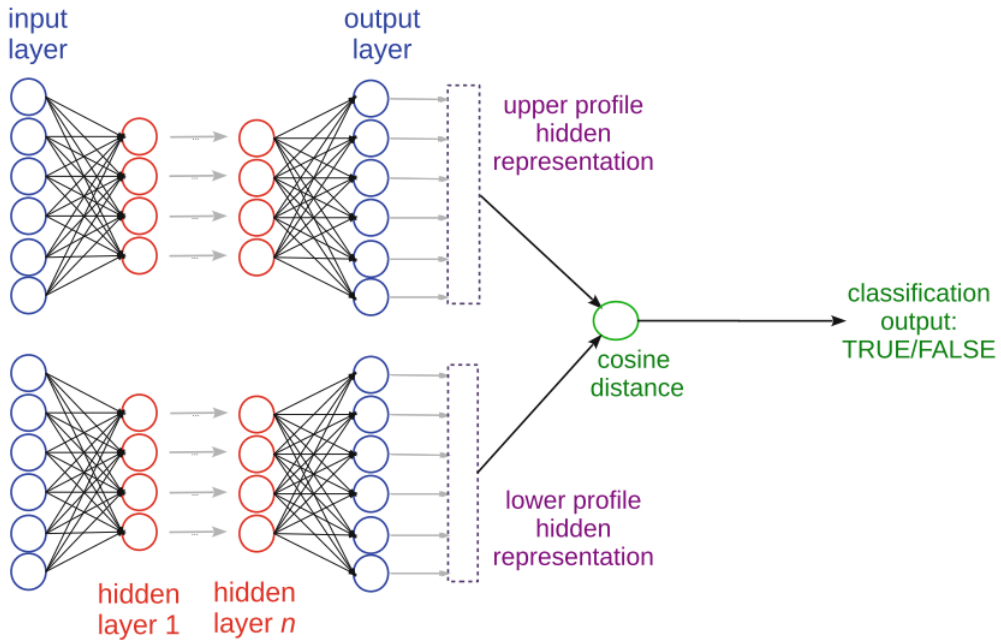

Figure 2.6: A simple structure of an SNN architecture. The input is processed in a feedforward fashion, from left to right. In this example, the cosine distance is used to measure the similarity of the embeddings (i.e., hidden representation), although a different measure could be used. The similarity value is contrasted with a threshold to give an output. The inputs belong to the same class (True) if it is greater than a threshold. Otherwise, they belong to different classes (False). Extracted from [4].

Therefore, in most cases, it is compulsory to find a method that processes and projects all the information in the objects to meaningful features in which a distance metric can later be safely and significantly applied, according to our objectives.

SNN offer a structured method to cope with this challenge. In particular, this architecture consists of two identical feedforward networks, whose last layers (which are to be referred to as the embedding layers) are joined by the application of a distance measure. Through this measure, the neural network states that the two profiles are different or similar according to a threshold fixed by the user. The algorithm then labels the data instance as positive or as negative, respectively. The output value can finally be compared with its corresponding ground-truth value [4]. Figure 2.6 shows the structure of a simple SNN from a fully connected network that serves as a feature extractor.

It is worth noting that the structure given to the network (what is in between the input and the embedding layer) can vary widely depending on the application and the nature of the input. For example, in the case of working with images, using Convolutional Neural Networks (CNNs) as networks for feature extraction is recommended [30]. A diagram of a SNN with a convolutional structure is depicted in Figure 2.7.

SNNs were first introduced by Bromley et al. [2] to recognize if two handwritten signatures belong to the same person. After that, several applications have been given to the model in

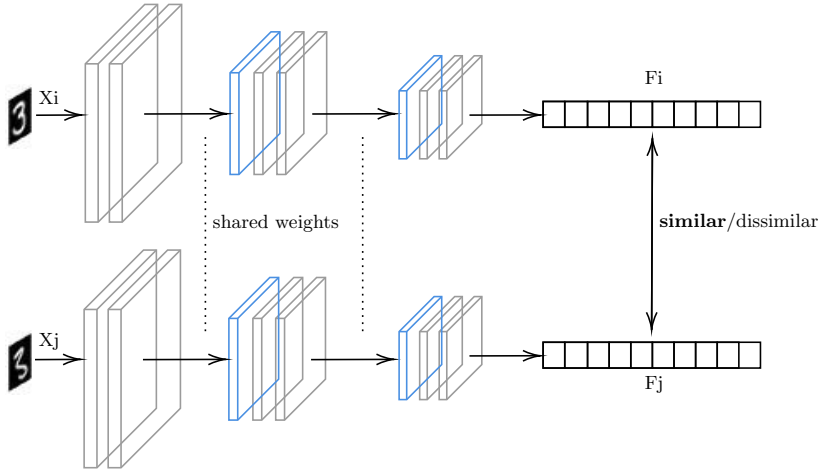

Figure 2.7: SNN with a convolutional architecture for image processing.

various problems. In a recent survey [4], more than one hundred applications are described in the following fields: audio and speech signal processing, biology, chemistry, physics, geometry, image analysis, medicine, robotics, sensor-based activity recognition, software development, text mining, and video analysis.

Most of the applications happened to be in images. However, there are some studies that focus specifically on face recognition [5, 34–38], in which are records in accuracy, and benefits on representation efficiency. One of the most impressive applications of SNN is one-shot image recognition, a framework that correctly makes predictions given only a single training data instance. This was first described in a paper by Koch et al. [16], where a convolutional neural network containing a SNN was satisfactorily trained to recognize images on the Omniglot dataset.

The most important state-of-the-art architectures on face recognition are DeepFace [38], Sphere Face [39] and FaceNet [5]. Because our proposed method is tested in this last architecture, it is described in Chapter 5.

2.1.3 Loss functions to train SNN

The most important characteristic of a loss function for training a SNN is that it penalizes the distance of the embeddings of the same class being large, and the ones of different classes being small. Next, some of the most popularly used loss functions in this architecture are described.

Triplet loss function

As indicated by its name, in this loss function training instances to be fed to the network are evaluated by triplets: the first image is used as an anchor (x^a), the second one (referred to as *positive* x^p) belongs to the same class as the anchor, and the third one (*negative* x^n) is from a

different class [5, 40]. The loss function is defined as:

$$L_{triplet} = \sum_{i}^{N} \left[\|h_i^a - h_i^p\| - \|h_i^a - h_i^n\|_2^2 + \alpha \right]_+ \tag{2.1}$$

where h_i is the embedding associated with the instance x_i (i.e., the output of the network), N is the number of training instances , and α is a real number parameter. Equation 2.1 penalizes all the cases where the euclidean distance between the anchor and the negative instance is smaller than α to that of the anchor and the positive case. In mathematical terms, the equation achieves:

$$\|h_i^a - h_i^p\| + \alpha < \|h_i^a - h_i^n\| \tag{2.2}$$

The expected effects of learning by this function are illustrated in Figure 2.8.

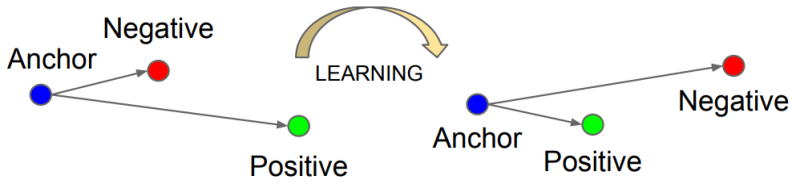

Figure 2.8: Triplet loss training. After training with the triplet loss function, the distance between the anchor and the positive case become smaller than the one between the anchor and the negative case. Extracted from [5].

Contrastive loss function

Contrastive loss was proposed by Hadsell et al. [41], and trains by processing pairs of input samples. The basic idea behind it is to penalize differently when the instances in the pair are from the same or different classes. In particular, with this loss the network is trained to produce similar feature embeddings if the target classes are the same and less similar feature embeddings in the converse case. Mathematically, contrastive loss function is defined by the following equation:

$$L_{contrastive} = \sum_{i,j} (1 - z_{ij}) \|h_i - h_j\|_2^2 + z_{ij} \max(0, \alpha - \|h_i - h_j\|_2^2) \tag{2.3}$$

where $z_{ij=1}$ when the instances are from different classes and $z_{ij} = 0$ when they are from the same. Accordingly, in the first case, the first term of the equation is cancelled and the loss function would try to make the distance between the embedding as small as possible until it reaches a value of α. On the other hand, when the pair is from the same class, the second term is cancelled and the loss function will force the distance of the two embeddings to be as close as possible, thus having a similar effect to the triplet loss function explained above.

Softmax loss and Center loss

Although contrastive loss and triplet loss functions are more widely used for training SNN, according to Parkhi et al. [37], softmax loss (that is more common in classifier networks) makes

training significantly easier and faster. To do this, it is necessary to put a fully connected layer with the same dimensions as the number of classes in the dataset connected to the last embedding layer (this extra layer is only for training and is removed for making inferences). Each feature in this new layer is related to a different class in the dataset and, when trained with the softmax loss, the network is induced to activate the feature that corresponds to the class of the input image. In notational terms, for the class $k = 1, ..., N$ (where N is the total number of classes), for each training image $x_i,\ i = 1, ..., M$ the network outputs a score vector $f(l_i) = v_i \in \mathbf{R}^N$ that is compared to the ground-truth class identity vector $e_{c_i} \in \mathbf{R}^N$. This vector e_{c_i} has all its entries equal to zero except the k entry that corresponds to the class of x_i, which is equal to 1.

As said, the described comparison is made according to the softmax loss function of the form:

$$L_{softmax}(W) = -\sum_{i=1}^{M} \log \left(\frac{e^{\langle e_{c_i}, v_i \rangle}}{\sum_{q=1,...,N} e^{\langle e_q, v_i \rangle}} \right) \tag{2.4}$$

To make the embedding vectors of the same class be close to each other and far from the ones in other classes, the softmax loss function is complemented with the center loss function. This last function attains this by learning a center for features of each class and penalizing the distances between features and their corresponding class center [36].

The expression for the center loss function is defined in Equation 2.5.

$$L_{center} = \frac{1}{2} \sum_{i=1}^{M} \|v_i - c_k\|_2^2 \tag{2.5}$$

where c_k denotes the k class center of the embedding layer features. Ideally, each class center should be updated according to the changes in the corresponding embeddings during training. Unfortunately, this would be highly impractical because it would be necessary to consider all the training set and average the features of every class in each training step. To solve this problem, each class center is estimated heuristically updating them per mini-batch. Also, to avoid large perturbation due to possible mislabelled samples, a scalar term α is added to control the learning rate of the centers:

$$\begin{gathered} c_k^{t+1} = c_k^t - \alpha \cdot \Delta c_j^k \\ \Delta c_k = \sum_{i=1}^{m} \frac{\delta(x_i \in \text{class } k) \cdot (c_j - v_i)}{1 + \sum_{i=1}^{m} \delta(x_i \in \text{class } k)} \end{gathered} \tag{2.6}$$

where m is the size of mini-batch, and the δ function is 1 when the condition evaluated is attained and 0 otherwise.

Considering both loss functions, the total loss is equal to:

$$\begin{aligned} L_{total} &= L_{softmax} + \lambda L_{center} \\ &= -\sum_{i=1}^{M} \log \left(\frac{e^{\langle e_{c_i}, v_i \rangle}}{\sum_{q=1,...,N} e^{\langle e_q, v_i \rangle}} \right) \frac{\lambda}{2} \sum_{i=1}^{M} \|v_i - c_k\|_2^2 \end{aligned} \tag{2.7}$$

Where λ is a parameter that modulates the relative importance of the center loss compared to the softmax loss.

Summing up, the total loss function learns to recognize different classes while fostering inter-class dispersion and intra-class compactness in the embedding layer. In the end, this makes the SNN achieve our goal in an indirect way, i.e. the embedding of the same classes are close to each other and far from the ones of different classes when a distance function is applied to them.

2.2 Explainability methods in artificial intelligence

The term explainable applied to an artificial intelligence engine is first mentioned by Van Lent et al. [42] where they describe a system able to explain the behavior of simulation game entities. However, the first instance of explainable AI (XAI) systems dates back to the 1970s [43].

Since the rise of deep learning in the last decade, the explainability of machine learning models has increasingly developed as a research field [6, 7]. Figure 2.9 illustrates the remarkable resurgence of explainable AI term research interest using Scopus trends. A possible cause for this phenomenon is the increasing influence of AI across multiple areas and its crucial impact on decision-making processes. The inability to provide detailed information about the reasoning that leads to AI decisions and predictions, as well as its application's social, ethical, and legal issues, demands special effort in developing new and effective explainability methods [6].

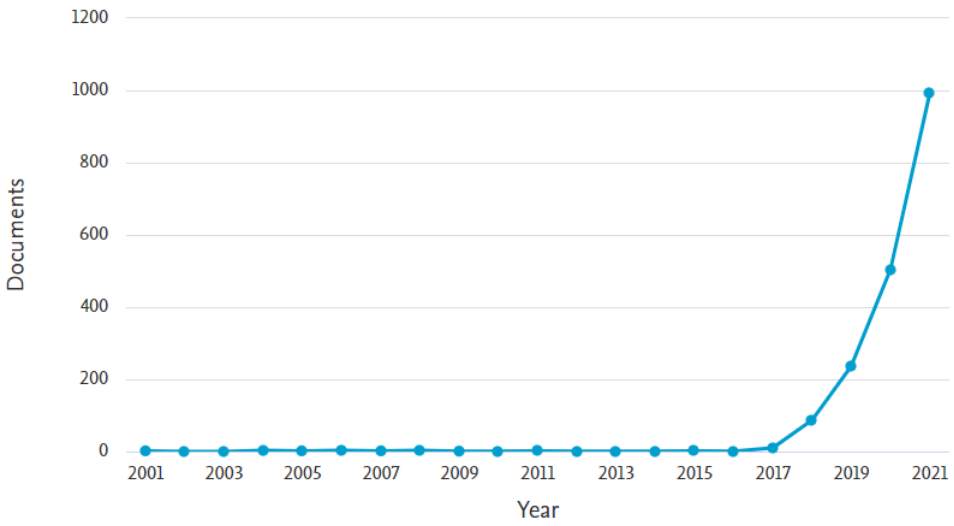

Figure 2.9: Number of publications from 2000 to 2021 related to the query "Explainable AND Artificial AND Intelligence". A total of 1857 publications are displayed. Search performed using Scopus on the 15th of August, 2022.

XAI is defined as a system with the ability to explain its rationale for making decisions. If an intelligent system resulting from a machine learning process is able to solve a problem and explain its solution, the confidence of its users increases, which contributes to the credibility of AI [6, 13].

Traditionally, deep learning explanation frameworks emphasize their application over input domains that are naturally visualizable and understandable, such as images or text, seeking representations such that an arbitrary input explanation can be visualized and presented to end-

users [8]. Commonly, such explanations are in the form of a superimposed image on top of the original input data instance to highlight what features of the input "explain" the decision.

2.2.1 Categorization

Explanation methods can be categorized as transparent or *post-hoc* techniques. Transparent methods explain how the machine learning model makes inferences, while post-hoc methods confer helpful information for the user without elucidating the internal mechanism of the model [14]. Some ways in which post-hoc methods can provide this information are [14].

- **Text explanations**, by training models that uses words for describing inferences.
- **Explanations by example**, that consist of reporting which examples the model considers to be most similar (this is similar to interpretations given by humans, where a decision is explained by referring to a previously known case).
- **Visualization**, in which an image or parts of the input are provided for explaining the inferences.

This kind of method can be are classified as [44]:

- **Model-agnostic**: if they work only with the inputs and outputs of a black-box model, and therefore can be applied to any architecture.
- **Model-specific**: if the tools used for interpretations are specific to a particular model or group of models.

On the other hand, explanations are *global* if they can give interpretations applicable to all the predictions and not just to one specific instance and *local* if they focus on specific sections of each instance, something that can be shown using masks or saliency maps highlighting the regions that, if changed, would influence the output most [14, 44].

2.2.2 Evaluating the quality of visual explanations

An objective measure is needed to assess the quality of produced explanations. Samek et al. [1,45] proposed a quality measure based on perturbation analysis. The method is based on the following three ideas:

- The perturbation of input variables that are highly important for the prediction leads to a steeper decline of the prediction score than the perturbation of the input dimensions which are of lesser importance.
- When the explanation method provides a score for the importance of an input variable. These can be sorted according to this score.
- It is possible to iteratively perturb input variables and track the prediction score after every perturbation step. The average decline of the prediction score (or the decline of the prediction accuracy) can be used as an objective measure of explanation quality (a large decline indicates that the explanation method successfully identified the relevant input variables).

In [1] the authors applied this method to compare explanations obtained by two existing explainability methods on a GoogleNet architecture used for image classification. The methods are SA (sensitivity analysis) [6], and LRP (Layer-wise Relevance Propagation) [46]. Both of them measure how important is every pixel in the classification of the image. First, the image is divided into 9x9 patches, and the importance of every patch is determined by summing the relevance of its pixels. At every perturbation step, the image patch with the highest value is replaced by random values sampled from a uniform distribution. The classification accuracy is measured on the perturbed image. From the ideas behind the method, it follows that if the explanations are good, classification quality should decrease in a logarithmic fashion, as the most important pixels are being perturbed first.

Figure 2.10 shows the results of comparing the methods. Since the prediction score decreases much faster when perturbing the images using LRP heatmaps than when using SA heatmaps, LRP provides better explanations than SA.

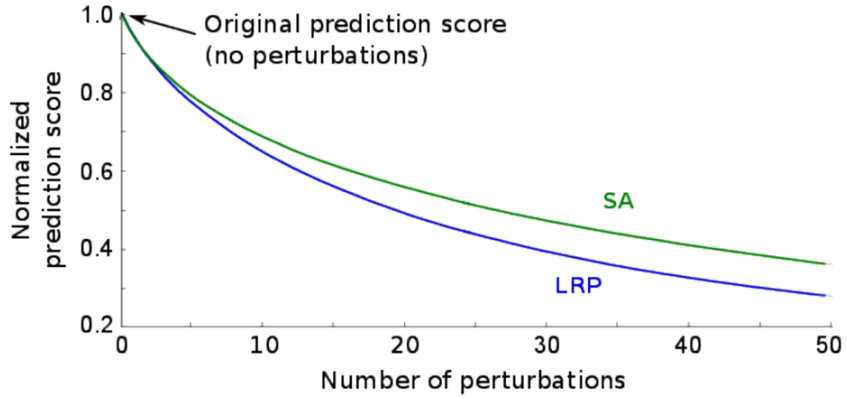

Figure 2.10: Relative decrease of quality on each iteration of the assessment method for LRP and SA [1].

Chapter 3

State of the Art

This chapter describes the existing methods of explainability for SNN. At the end, there is a discussion about limitations of these methods, that are considered when we reflect upon the improvements of the proposal of this work in the next chapter.

3.1 Explainability of Siamese Networks

As far as we know, the only explainability methods specialized for SNN available in current literature are those formulated by Utkin et al. [18] and Chen et al. [17]. Of them, the only one that was applied to explain images is the one proposed by Utkin et al. [18]. Both methods are described below.

3.1.1 An Explanation Method for Siamese Neural Networks (EMSNN)

This method was proposed by Utkin et al. [18] and is going to be referred to as EMSNN, based on the acronym of the original paper, whose title is the same as the one of this subsection. The method is model-agnostic and explains by highlighting what are the most important image features for considering that an instance belongs to a certain class. To do this, the following method is applied:

- Build an autoencoder [47] whose hidden bottleneck layer has the same dimension as that of the embedding vector. The autoencoder is trained using a loss function that has two main terms. The first one influences the autoencoder to make the output reconstruction similar to the original input image (L_{recon_a} in equation 3.1). On the other hand, the second term attempts to make the bottleneck layer representations resemble the embedding vector given by the SNN when that image is given to it as input (L_{close}). Mathematically:

$$\begin{aligned} L_{autoencoder}(W) &= \gamma L_{recon_a}(W) + \mu L_{close}(W) + \lambda R(W) \\ &= \gamma \sum_{i=1}^{n} \|x_i - \tilde{x}_i\|_2^2 + \mu \sum_{i=1}^{n} \|h_i - \tilde{h}_i\|_2^2 + \lambda R(W) \end{aligned} \tag{3.1}$$

  where W are the network weights, x and h are the values of the original input image and embedding vector respectively, and $\tilde{x}$ and $\tilde{h}$ are the reconstructed output and bottleneck

layer of the autoencoder. R is a regularization term, while γ, μ, and λ are parameters for varying the relative importance of the different terms of the equation. This training step is depicted in Figure 3.1The only part of the autoencoder used in the following steps is the decoder (i.e., the network between the bottleneck layer and the output) to reconstruct images from embedding vectors.

- Generate a prototype of each category by taking the average of the sum of all embedding vectors belonging to it according to the equation:

$$c_k = \frac{1}{n_k} \sum_{i:y_i=k} f(x_i) = \frac{1}{n_k} \sum_{i:y_i=k} h_i \tag{3.2}$$

where k is a specific category, c_k is its prototype, n_k the number of instances that belong to k, and $f(x_i)$ the output of the SNN when taking image x_i as input; this output is equivalent to h, the embedding vector. This step, and all the followings, are depicted in Figure 3.2.

- Determine, for the instance that is going to be explained, what are the most important embedding features. These are considered to be the ones that are closer to the corresponding one of its prototype.

- Perturb the most important embedding features randomly.

- Use the decoder to obtain a reconstruction of the image from its perturbed embedding vector. The perturbation and reconstruction process is repeated several times to find the pixels that undergo, on average, the largest changes compared to the original input.

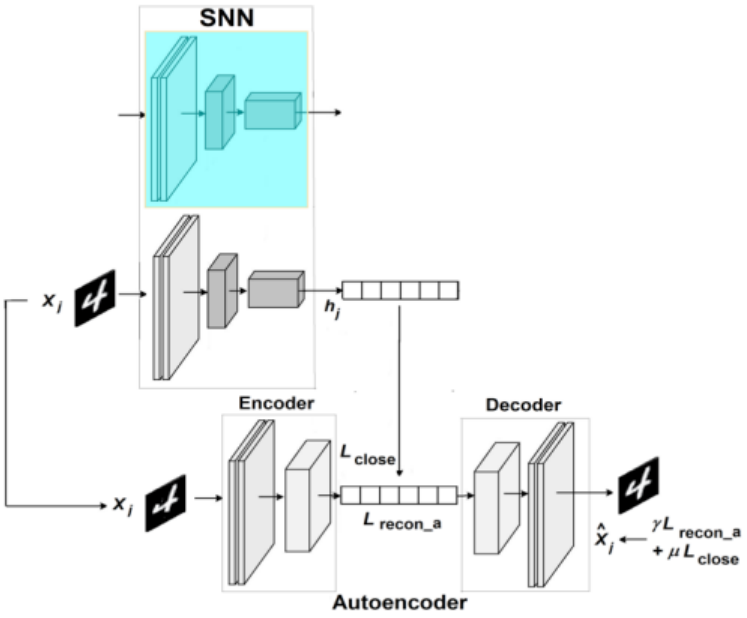

Figure 3.1: Training of the autoencoder, step 1. Extracted from [18].

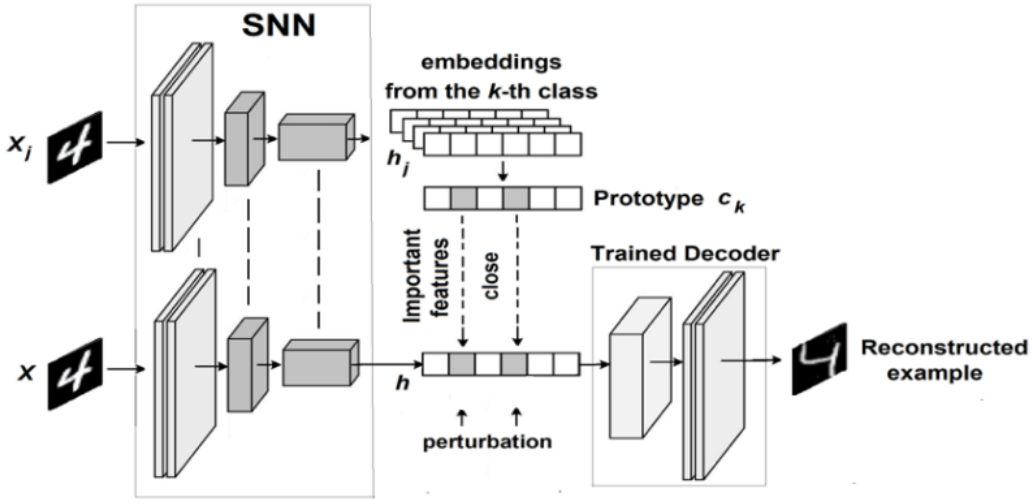

Figure 3.2: Training of the autoencoder, step 2. Extracted from [18].

The process above relates the most important features in the embedding to those of the original input image by assuming that those pixels that suffered the greatest perturbations are "encoded" in the most important features.

To analyze their results, the author showed what are the areas that suffered the greatest perturbations, and superimposed them on the original images (Examples in Figure 3.3). According to the authors, these areas are what make that number similar to those belonging to the same category.

Figure 3.3: Examples of explanations by the method given by Utkin et al. [18] From left to right, the first column represents the original inputs, the second the reconstructions obtained by the autoencoder; in the third one the areas that suffered on average the greatest changes are highlighted in yellow, and on the last one those areas are shown isolated from the rest of the image [18].

3.1.2 Self-learn to Explain Siamese Networks Robustly

This method is proposed by Chen. et al [17] and is also model-agnostic and post-hoc. In the original paper, the method is applied to tabular data and graphs, so an application to images may require adaptations. An example of an application is illustrated in Figure 3.4.

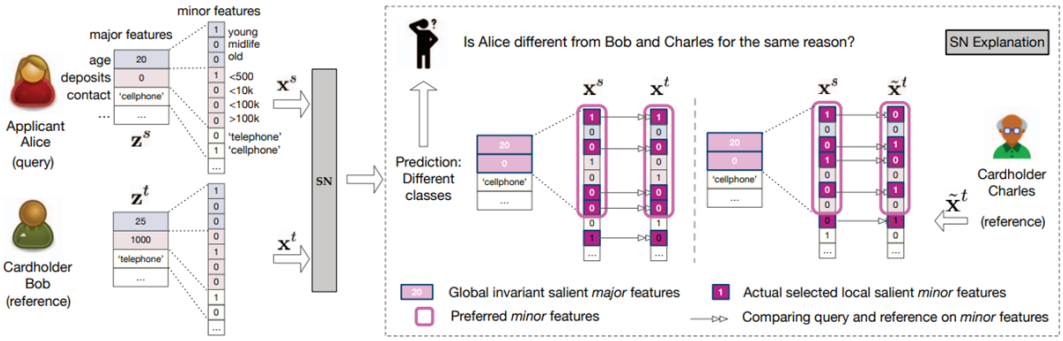

Figure 3.4: Application of the method on a SNN that determines how similar are two different clients of a bank. First, data are converted using one-hot encoding. Then, global important features are selected (pink cells). The method creates masks by selecting the most important local minor features (dark purple cells with white numbers) according to the criteria described in the text (simplicity, faithfulness, conformity, and not missing relevant features). On the right, the test, Alice, is compared with two different references, Bob and Charles, obtaining different local explanations in each case. Extracted from [17].

For our purposes, it is not necessary to give a detailed description of the method, but only to describe the characteristics of the explanations obtained. The aim of the method is finding masks (m) for the original inputs that have high *simplicity*, *faithfulness*, *conformity* and that do not miss important features. Formally, the characteristics are defined in the following way:

- **Simplicity** means that masks select a small amount of highly significant features. A way of measuring this is with the ℓ_1 norm of the mask m.

- **Faithfulness** means that the masks select salient features that preserve the SNNs' output $f(x_i, x_j)$. Faithfulness is evaluated by feeding the SNN with the masked instances $f(x_i \otimes m_i, x_j \otimes m_j)$ and measuring the differences between the original and the masked outputs with a loss function, such as a cross-entropy loss.

- The method allows to select globally important features (those that are relevant regardless of references), that can be determined by the user by considering previous knowledge in the topic. Based on that, **conformity** means that the global salient features overlap with the local salient ones (those that are relevant for a particular comparison). This gives robustness to the method so that a particular explanation for an instance doesn't vary too widely depending on the instance used as a reference for the comparison. As an example, in the application shown in Figure 3.4, global features are "age", "deposits" and "contact", but only the first two are considered as important. Then, for a specific comparison, different minor features inside these global features are considered for the mask.

- For measuring **how much the masks miss relevant features**, a counterfactual loss is defined:

$$\ell(f(x_i \otimes (1 - m_i), x_j \otimes (1 - m_j)), f(x_i, x_j)) \tag{3.3}$$

  This loss function gives information about what the prediction would be if the features not incorporated in the mask were selected.

3.2 Limitations of existing methods

Because the second method described is not specially designed to work with images, the following reflection focuses on the limitations we could detect in EMSNN.

First, we think that the method designed for reconstructing images from SNN embeddings is unnecessarily complex. We think that the part of the autoencoder that takes an image to create a bottleneck representation similar to that of the embedding should be omitted, because the goal of all this process could be attained more directly if the embeddings were immediately fed as input to the reconstruction network. In this way, the bottleneck layer would not just be *similar* to the embedding, it would be *the same*. Moreover, this method only explains when two instances are similar, not taking into account what are the most important features of the image when they are considered to be different by the network, something that is another possible behaviour of it.

Also, we believe that using a prototype doesn't reflect the nature of the inferences made by the SNN, because the model uses just two instances to make a comparison, and all the images of the class are considered in the prototype. Besides, the output of the networks is a measure of similarity, and not a direct categorization, thus saying that all the images in the prototype are from the same class could be a conflictive statement. A specific problem related to this is that it would be hard to decide how to deal with possible inconsistencies between the instances inside a category. For example, in some cases instance 1 and instance 2 of the same class might be evaluated correctly as similar, as well as instances 2 and 3, although instances 1 and 3 could be incorrectly classified as belonging to a different class.

Another problem with using a prototype is that its value depends on the instances available at the moment, and if new instances of the class are included in the dataset, then the prototype should be calculated again and, for more rigor, the previous inferences should even be repeated. Also, the prototype could suffer from a lack of quality if just a few instances belonging to a certain class were available. This offers a special problem when working with images of people, where there exist privacy issues, and could even be legally conflictive to require several identified images of the same person for some applications of image recognition [21].

Also, we think the method considers a lot of parameters, and the values given to them could affect the behaviour of the explanations obtained. The next list summarizes all the parameters involved in it:

- There are three to define how the reconstructed network is trained.
- There is one for choosing how many features of the embedding will be selected as the most important.

- There are two parameters in the perturbation process: one for defining the magnitude of the random perturbation, and another one for deciding how many perturbed embeddings will be used to determine the average changes in the reconstructions.
- There is one for considering how many pixels of the image will be considered the most important ones.

Chapter 4

Proposed Method

In this chapter, the steps of our proposed explanation method called "Explaining from Local Perturbations on the Embedding Layer" are described. Then, there is a description of the procedure followed to assess the quality of the obtained explanations. Finally, we expose the advantages of this method compared to the existing ones on siamese architectures described in the previous chapter.

4.1 Explaining from Local Perturbations on the Embedding Layer

Our method is post-hoc, model-agnostic, and uses local visualizations to provide information about the most important areas of the image when making inferences. Its goal is to recognize what is the meaning of the most important features that the network is taking into account to consider whether two instances are similar or dissimilar enough to belong to the same or different classes.

The big picture of our proposal comprises the following steps:

1. Train a network called *decoder* to reconstruct the original image from the SNN embedding.
2. For each comparison, choose an instance to be the *reference* and the other one to be the *test*. Assign a relevance measure to every feature of the test embedding based on their distances to the ones of the reference embedding. Then, perturb every feature of the embedding in a magnitude according to its relevance.
3. Fed the original and the perturbed embedding to the decoder, to get their associated reconstructions. Assess what are the most significant differences between the image reconstructed from the original embedding and the ones reconstructed from the perturbed embedding. The pixels that suffered the most significant changes are considered the most important ones.

Because a perturbation on a feature might correspond to a change in an area of the reconstructed image, by following the described method it is possible to infer what features of the image are encoded in the most important embedding features. This gives valuable information

to clarify what the SNN is taking into account when making its inferences. In the following sections, each one of the steps mentioned above is described in detail.

4.1.1 Step 1: Training the decoder to reconstruct images from embedding vectors

We use a neural network (the *decoder*) to associate the features of the embedding vectors given by the SNN to the features of the corresponding input image. To do this, it is necessary to obtain first all the embedding vectors of the images in the train set by processing them in the SNN. Then, the embedding vectors (denoted h) are given to the decoder as input, and the network is trained to make the output ($g(h_j) = \tilde{x}_j$) as similar as possible to the image related to the embedding (x_j). This is done by training using the following loss function:

$$L_{decoder}(W) = \sum_{j=1}^{M} \|x_j - \tilde{x}_j\|_2^2 \tag{4.1}$$

where M is the number of instances of the train dataset, and $\|\cdot\|_2$ is the euclidean norm. One step of this process is illustrated in Figure 4.1.

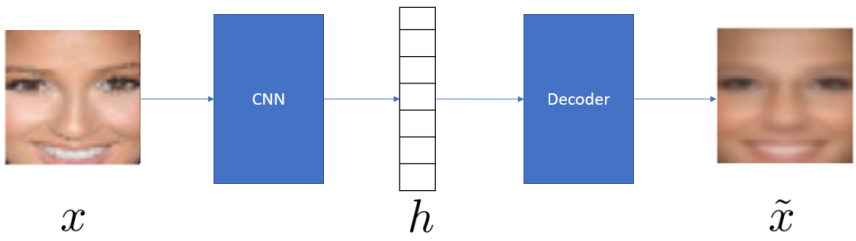

Figure 4.1: Step 1 of the explainability method. Example of a training step for the decoder. Only one branch of the SNN is illustrated in the figure. This is depicted by the CNN block, recalling the convolutional structure of this architecture.

4.1.2 Step 2: Perturbing the embedding features according to their relevance

The relevance of each feature is determined by directly comparing the embeddings of two different instances. For doing this, the distance between each feature is calculated by taking the absolute value of the subtraction:

$$d_i = |h_i^r - h_i^t| \tag{4.2}$$

where $i = 1, ..., N$, the number of features. In this equation, h^r and h^t correspond to the $\mathbb{R}^N$ embedding vector of the reference and test, respectively.

When the network categorizes two images as being similar, the most important features are those that are closer; in the case that they are dissimilar, they are the ones that are further.

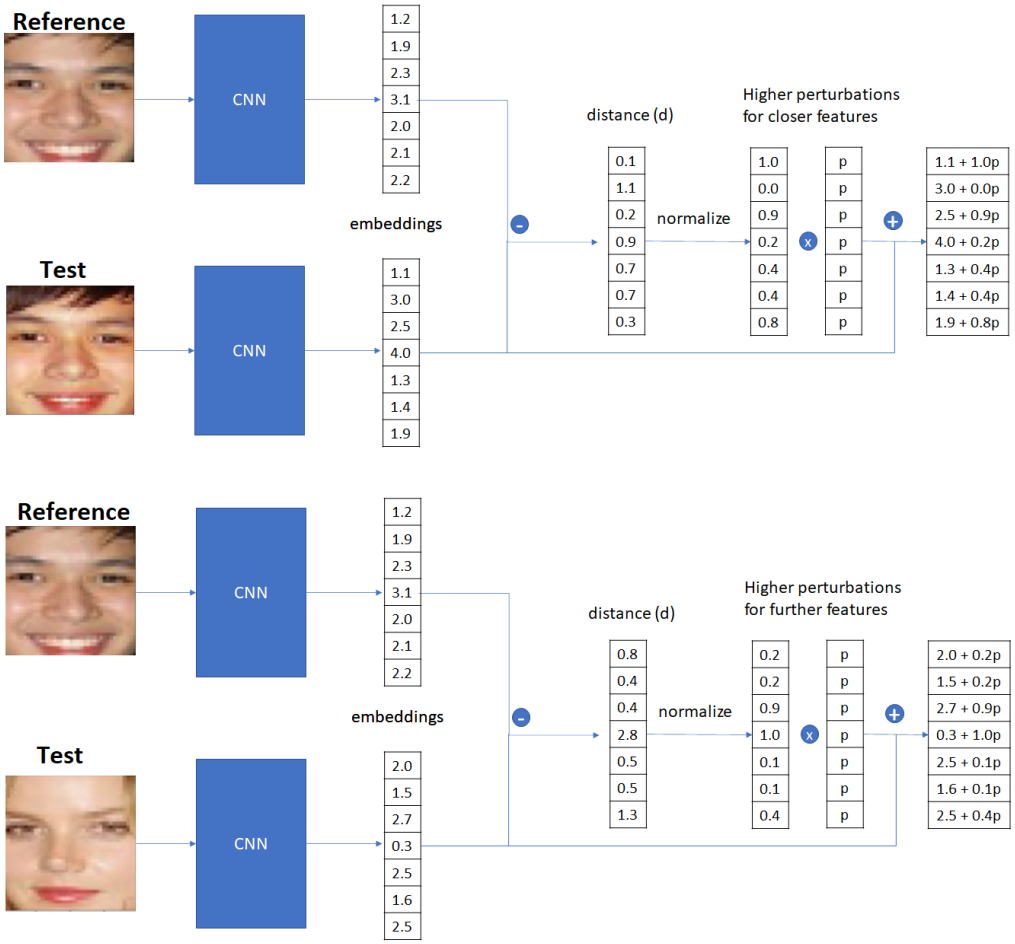

Figure 4.2: Step 2 of the explainability method. The example of the top represents the method when it is applied to pairs of similar images, while in the example of the bottom the images are from different categories. First, the embeddings are obtained by processing them in the SNN with a convolution architecture (CNN). The magnitude of the perturbation applied to the test embedding depends on the distance between the corresponding features: when instances are similar, the perturbation is higher for closer features, and the converse is true when the images are dissimilar. The values in this figure are for illustrative purposes and are not real.

To reflect this, the distances are normalized to take values between 0 and 1 according to the following equation:

$$\bar{d}_i = \frac{d_i - min(d)}{max(d) – min(d)} \tag{4.3}$$

and then the relevance of the features is determined according to the $r \in \mathbb{R}^N$ vector whose components are given by:

$$r_i = \begin{cases} |1 - \bar{d}_i| & \text{if instances are similar} \\ \bar{d}_i & \text{if instances are dissimilar} \end{cases} \tag{4.4}$$

for $i = 1, ..., N$.

Then, choose a value for the parameter $p \in \mathbb{R}$ that defines the magnitude of perturbation, and perturb the test embedding according to equation 4.5.

$$h^p = h^t + r * p \tag{4.5}$$

where h^p refers to the perturbed embedding.

By following the described step, when the instances are similar the features that are closer are perturbed to a greater extent, and the converse is true when the instances are dissimilar. Both cases are shown in Figure 4.2.

4.1.3 Step 3: Inferring the most important image features

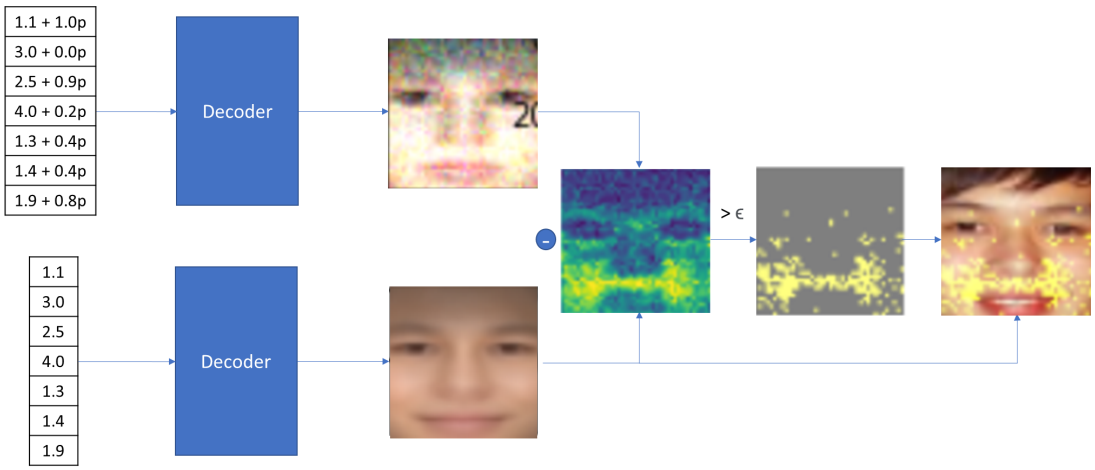

Figure 4.3: Step 3 of the explainability method. The perturbed embedding and the original test embedding are given as input to the decoder trained as described in step 1 to obtain the associated reconstructions. The differences between the pixels in the reconstructions define a heat map, where the highest values reflect the pixels that changed the most with the perturbation process of step 2. These pixels are considered the most important ones for defining the SNN inference. A mask is built by selecting only the pixels whose change is greater than a threshold value ϵ. Finally, this mask is superimposed on the original image.

The image features associated with the perturbed embedding features are those whose reconstruction shows the greatest changes when contrasted against the original reconstruction. To get the most important image features, we proceed in a similar fashion to how we determined the most important embedding features in step 2.

First, using the decoder, get the associated reconstruction of the original and the perturbed embeddings of the test instance ($g(h) = \tilde{x}$ and $g(h_p) = \tilde{x}_p$ respectively). Then, the distance $D \in \mathbb{R}$ between the pixels of both reconstructions is calculated:

$$D_i = |\tilde{x_i} - \tilde{x}_{pi}| \tag{4.6}$$

For improving clarity and interpretability, it is possible to select a threshold for visualizing just a part of the most important image features. In order to do this, it is convenient to apply the same normalization to D to that described in Equation 4.3 in step 2 to get the normalized distance $\bar{D}$. Then, select a threshold $b \in [0, 1]$ and the most important features $\tilde{D}$ are those above that threshold, as indicated by equation 4.7

$$\tilde{D}_i = \begin{cases} 1 & \text{if } \bar{D}_i > \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

This process is illustrated in Figure 4.3.

4.2 Assessment of the quality of explanations

Figure 4.4: Method for assessing the quality of explanations. The image is divided into a grid of superpixels, and the superpixels with the greatest relevance score are perturbed first. On each perturbation, the resulting image is given to the SNN to calculate its similarity with the original reference image. When the instances are dissimilar, the reference image is also perturbed in the same superpixels that the test image before measuring the similarity again.

To assess the quality of the explanations we do some modifications to the method proposed by Samek et al. [45], described in the chapter 2, to apply it to SNN.

First, the image is divided into a square grid, and the relevance of each square superpixel is estimated by summing the relevance of each pixel defined by equation 4.6. Then, on each iteration of the method, the superpixel with the highest relevance is selected and the values of its pixels are replaced with a random number taken from a normal distribution with mean 0, and variance σ. It is expected that if the method works well, the similarity will decrease when applying it to similar instances, and it will increase when applying it to dissimilar ones, as the most important areas on each inference are being perturbed.

To achieve this, both cases are treated in a different way. When assessing a pair of images of the same category, only the test image is perturbed and the reference is kept without changes. On the other hand, if we are dealing with images of different categories, the corresponding superpixels of both images (the reference and the test) are perturbed. This is done because one image with a perturbation and the other one without it are, in most cases, more dissimilar than the original images without perturbations. In this way, it would be impossible to measure an increase of similarity by perturbing dissimilar instances, as desired (this claim was tested by doing experiments). Figure 4.4 shows this method applied to a pair of similar instances.

4.3 Improvements over existing methods

Considering the ideas exposed in the last section of the State of the Art chapter about the limitations of the existing methods, the improvements of our proposal are the following:

- The neural network used for reconstruction from embeddings has less complexity than the one in EMSNN. because all the previous structure that is connected to the bottleneck layer is omitted.
- For explaining an inference, only two instances are needed, the same used by the SNN on each inference (EMSNN required a set of instances for building a prototype).
- This method gives explanations for both possible cases: whether the model is categorizing the instances as similar or as dissimilar.
- Only two parameters are needed to define our method: one determining the intensity of perturbations, and another one used as a threshold to select the most important image pixels that are shown in the visualization.
- We implemented a method for quantitatively assessing the quality of the explanations given by our proposal.

Chapter 5

Experiments

In the first part of this chapter, we describe important information regarding the implementation of our method, such as the architectures of the neural networks used, and the datasets used for training and testing. Then, we present and discuss relevant results by applying the method described in the proposal.

5.1 Siamese network implementation

5.1.1 Architecture

As noted in Chapter 2, SNN consist essentially of two identical neural networks connected by their outputs. The chosen structure for this network is something that depends on our objectives and on the nature of the input. Because we are going to work with images, the best choice is to use a convolutional neural network [30]. In the implementation of our proposal, the chosen structure is based on the FaceNet model. This model is based on the Inception-ResNet-v1 architecture [48]that has been previously used in SNN to identify if the faces in two different images are from the same person [5]. Figure 5.1.1 shows the Inception-ResNet-v1 original architecture (all images from 5.1.1 to 5.7 were taken from [48]) The modifications that are done to originate the specific FaceNet architecture used are:

- The Input, whose dimension is changed to 160x160x3.
- The Dropout layer, where the keep probability is diminished from 0.8 to 0.4.
- The last layer, where the softmax activation is replaced with a linear activation layer of dimension 512.

The embedding vectors that are the output of two of these identical networks are connected by a neuron that measures its cosine similarity given by Equation 5.1.

$$\text{cosine similarity} = \frac{h_i \cdot h_j}{\|h_i\|_2 \|h_j\|_2} \tag{5.1}$$

where h represents the embedding vectors. The value of the cosine similarity is between $[-1, 1]$, where proportional vectors have similarity 1, orthogonal 0, and opposite -1 [49]. Then, as ex-

plained in the previous section, by defining a threshold it is possible to choose if the compared instances are from the same or different classes.

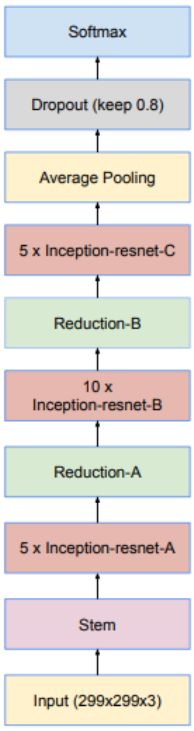

Figure 5.1: Original Inception-ResNet-v1 architecture.

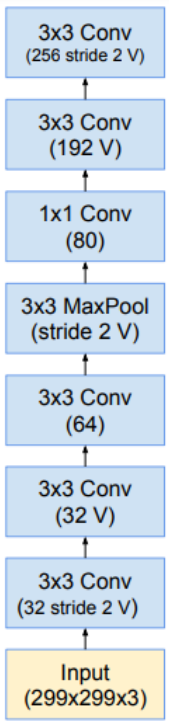

Figure 5.2: Detailed structure of the Stem of the Inception-ResNet-v1 network.

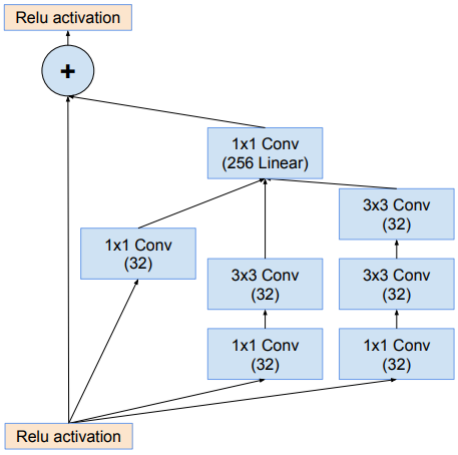

Figure 5.3: Detailed structure of Inception-ResNet-A module in Figure 5.1.1.

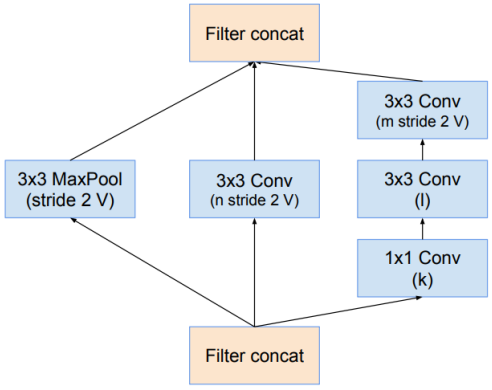

Figure 5.4: Detailed structure of Reduction-A module in Figure 5.1.1.

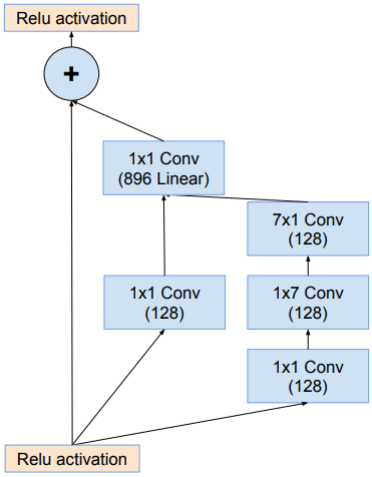

Figure 5.5: Detailed structure of Inception-ResNet-B module in Figure 5.1.1.

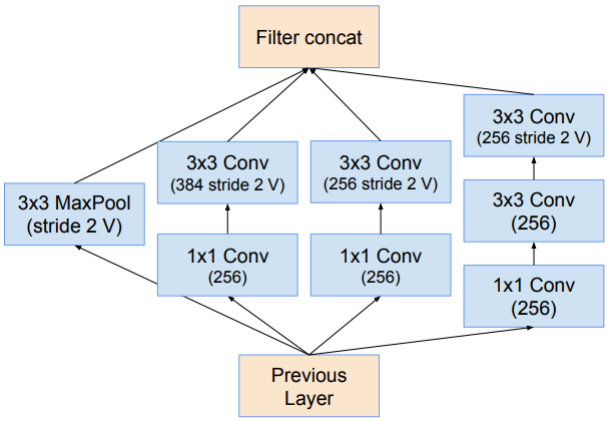

Figure 5.6: Detailed structure of Reduction-B module in Figure 5.1.1.

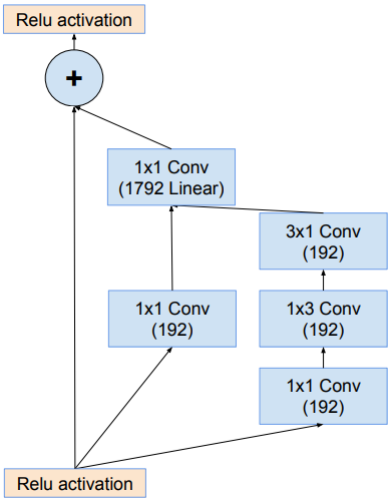

Figure 5.7: Detailed structure of Inception-ResNet-C module in Figure 5.1.1.

5.1.2 Training and Testing

The SNN model was training using the VGGFace2 dataset that consists of 3.31 million images of 9131 different subjects with an average of 362.6 images for each subject [50]. The images on it were selected to reach a very low label noise and a high pose and age diversity. To foster good generalizations, the images have a large variation in pose, illumination, ethnicity and profession of the subjects. In spite of that, the authors claim that the distribution of identities in the VGG-Face dataset may not be representative of the global human population, and, therefore, should

be used with the awareness that it might unintentionally reproduce societal, gender, racial, and other biases. This architecture reaches one of the highest accuracies achieved for an artificial engine on face recognition [5] (99.63 % on the LFW dataset [51]), something that makes it especially interesting for explaining its behaviour. Because of the structure of the explainability method, the model is also tested in the dataset used for training the decoder. The results are shown in the next section, where this dataset is described.

5.2 Decoder implementation

5.2.1 Architecture

We tried several different architectures for training the decoder. Because the loss obtained did not change considerably in our different trials, we decided to use the simplest architecture possible. This consists of a network in which the neurons of the input are fully connected with those of the output. Then, this output is reshaped so that it has the same shape that the images in our training dataset (40x40x3). This architecture is shown in Table 5.1. Because we are reconstructing from the embedding vector of the SNN, it is also important that the dimension of the input matches that of the embedding (512). In its implementation, the Adamax optimizer performs gradient descent with a learning rate of 0.001.

Layer	Output
Input	512
Dense	4800
Reshape	40x40x3

Table 5.1: Decoder architecture.

5.2.2 Training and Testing

We used the CelebFaces Attributes Dataset (CelebA), which contains more than 202,599 celebrity face images from 10,177 different identities. The images in this dataset cover pose variations, as well as differences in gender, race, age, among others [52].

The first 24000 images of the dataset are used for training and testing the decoder. We did not use all the images because we observed that the quality of the reconstructions stopped improving for a larger dataset, and this decision diminished the computational time required to train the network. A 40x40 square of the original images is cropped so that only the faces, and not the background, appear on them. Because the images are in RGB format, the final structure of the data is 40x40x3. Some examples of these images are depicted in Figure 5.8.

Because we need to get the embedding associated with these images, it is necessary to rescale them to a 160x160x3 shape retaining its quality to match the input of the siamese architecture used. The embeddings are saved for giving them later to the decoder so that this network learns to reconstruct the original faces from the embeddings by the method described in the proposal. One thousand images from the described set are selected for testing: these are also the images that are used later for applying the explainability method. The loss obtained on them is of 0.0295.

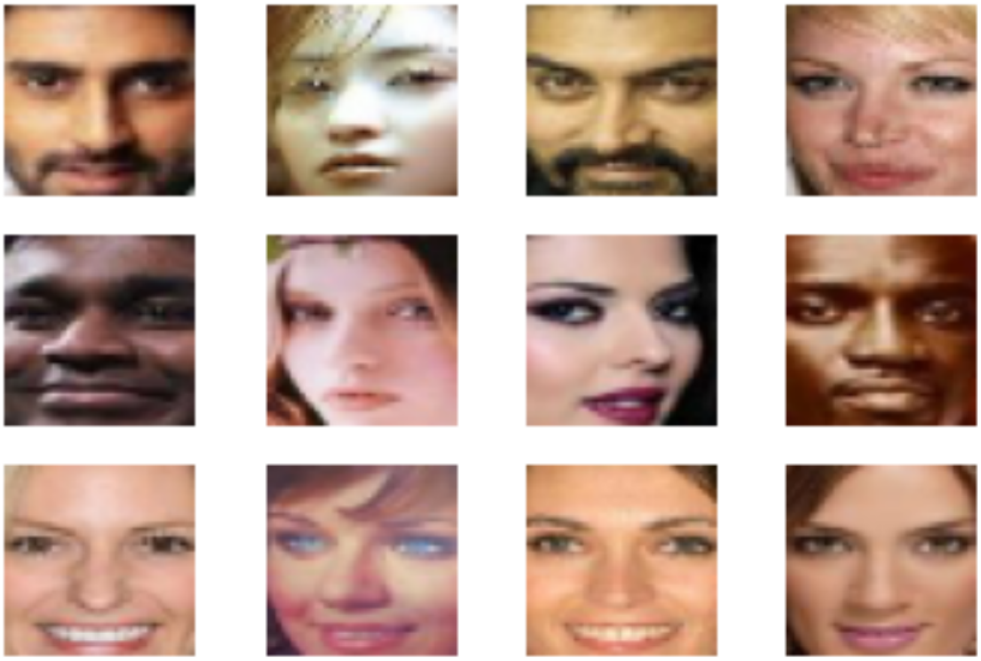

Figure 5.8: Some examples of the images used for implementing the method.

Because faces are of high complexity and the embedding vector given by the network is of relatively low dimensionality (512), the reconstructions obtained did not represent the details of the images well. Even though, we don't believe that that is a problem because our goal is not to get high quality reconstructions, but ones whose different areas correlate appropriately with those of the original input. Examples of original images and their corresponding reconstructions are shown in Figure 5.9.

For evaluating if the SNN did good inferences in the data used, its accuracy is measured on a subset of it. In order to do this, a threshold of 0.3 is selected, and 70000 comparisons are made, half on instances of the same class and the other half on instances of different classes. In the first case, an accuracy of 0.8959 % was attained, while in the second case, the accuracy was of 0.9122 %. Therefore, the total accuracy was of 0.9041%. Although the quality of the inferences decreased on the used dataset, it is still very high. Also, we are not interested in reaching a good performance on face recognition, but in explaining the behaviour of the network, thus this decrease is not in conflict with our objectives.

5.3 Results and Discussion

For defining the magnitude of the perturbation, we set the parameter $p = 0.3$ on Equation 4.5. We tried different values for the parameter, but because the resulting mask and the quality assessments did not change significantly we just show the results obtained with this value.

Some results of the experiments for similar instances are depicted in Figure 5.10. Because the masks highlight equivalent areas on the original image and on the reconstruction, this gives

Figure 5.9: Original images (left column) and their corresponding reconstructions (right column).

evidence of the suitability of our conjecture that reconstructions with very high quality are not needed to associate embedding features with image features.

For every pair of images being compared, we applied the method twice, using the former test as reference and the former reference as test in the second trial. This is reflected in Figure 5.10 in that every pair of rows has the same images in columns Reference and Test, but with changed positions. By doing this, the only thing that changes on each trial is which one of the two embeddings associated with these images is being perturbed and reconstructed. The magnitudes of the perturbations applied in both trials are identical, because the distances between the embedding features are the same independently on which embedding is used as reference. This suggests the interesting result that, at least when comparing similar images, the features of the two embeddings might be encoding almost the same image features, something that is reflected in the fact that the masks and the heat-maps obtained in both trials for the same pair are essentially identical (to see this, compare the columns Reference Highlighted, Reconstruction Highlighted, and Heat-map on each pair of rows of Figure 5.10).

The areas selected as the most important are generally the cheeks, the one between the mouth and the nose, and the eyebrows. This importance changes between each pair of images. For example, in the man of the first example, the eyes are more important than in the women of the other examples, and the mask that is between the superior lip and the mouth follows the shape of the moustache. Also, in the mask produced by the comparison of the last woman, there is a bigger area surrounding the eyes, and a greater area of the cheeks is highlighted, if contrasted against the other two results. In Figure 5.11 some results of the experiments explaining dissimilar instances are shown. The areas highlighted in these cases are quite similar to those of image 5.10, suggesting that almost the same parts of the faces are the most important whether assessing similarity or dissimilarity. For a given set of images, the masks and heat-maps changed more depending on which image is used as a reference, and therefore what the features are encoding might change more significantly in instances of different categories.

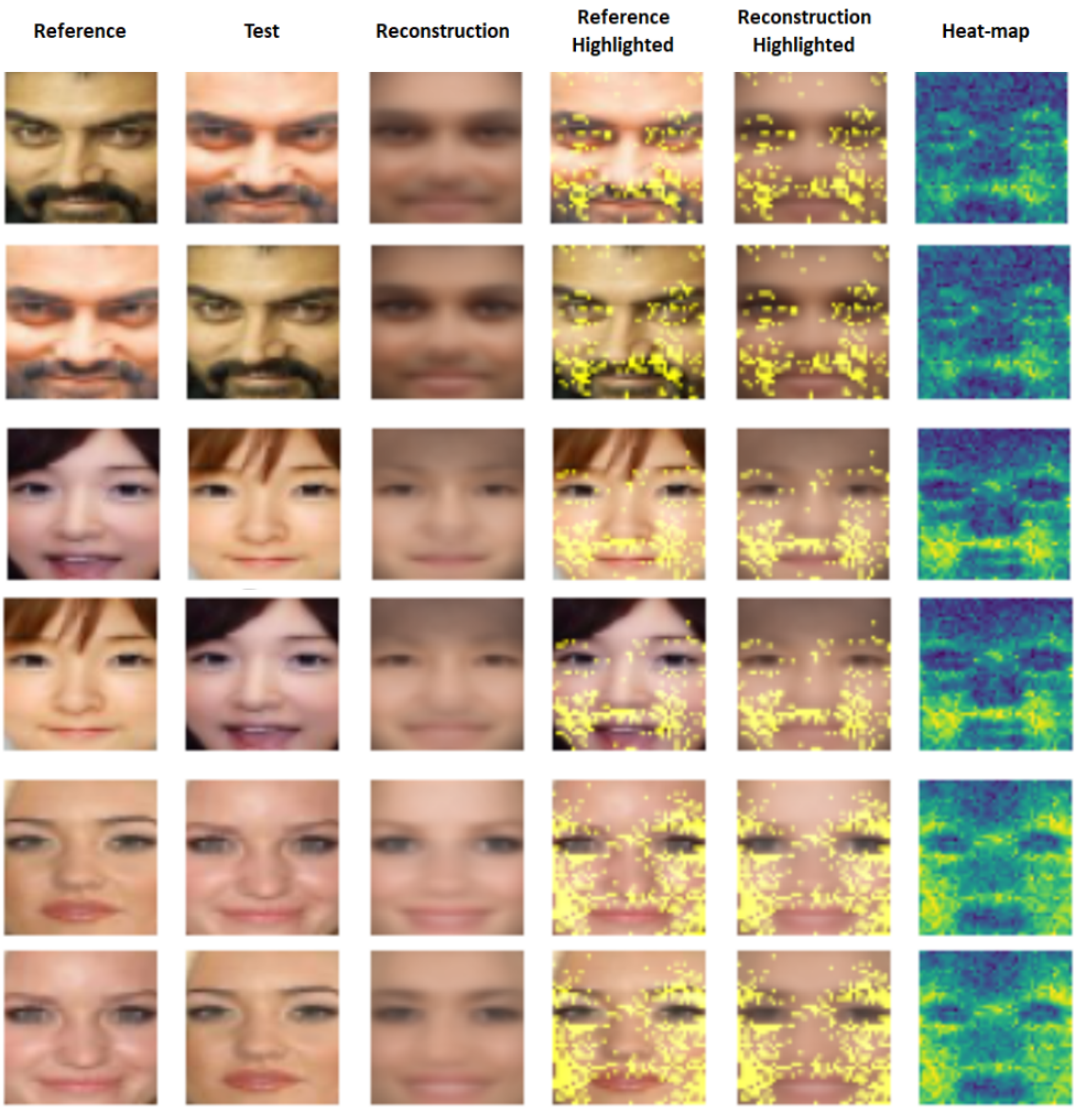

Figure 5.10: Results provided by the proposed method when the network recognizes two faces of the same person as similar. Each row represents a different case, where the 1st column is the reference, and the 2nd is the test. The reconstruction in the 3rd column is done based on the test embedding. In the 4th and 5th columns, the most important features are highlighted in the reference and the reconstructed image, respectively. These masks of highlighted pixels were generated by selecting the pixels whose normalized relevance value is greater than 0.7 (this means that in Equation 4.7 we set the parameter $\epsilon = 0.7$). In the Heat-map column, pixels with higher relevance are in yellow, and the ones with lower are in blue. The similarity of the images in the first pair of rows is 0.8436, in the second pair of rows is 0.7787, and in the third is 0.4549.

Figure 5.11: Results provided by the proposed method when two faces of different persons are recognized by the network as dissimilar. The similarity of the images in the first pair of rows is -0.1270, in the second pair of rows is -0.1965, and in the third is 0.1817.

5.4 Evaluating the quality of the explanations

Figure 5.12 shows the decrease of the normalized similarity score as superpixels in the image are being perturbed following the method described in the "Assessment of the quality of explanations" section of Chapter 4. As an example, Figure 5.13 illustrates how one instance is perturbed. The normalized similarity is calculated by fixing the similarity value of the non-perturbed images as 1, and representing the similarity of the perturbed images as a proportion of that original value. For example, a value of 0.8 means that the original similarity decreased on average by 20%. In all the graphs of this type shown in the present work, 100 different comparisons were

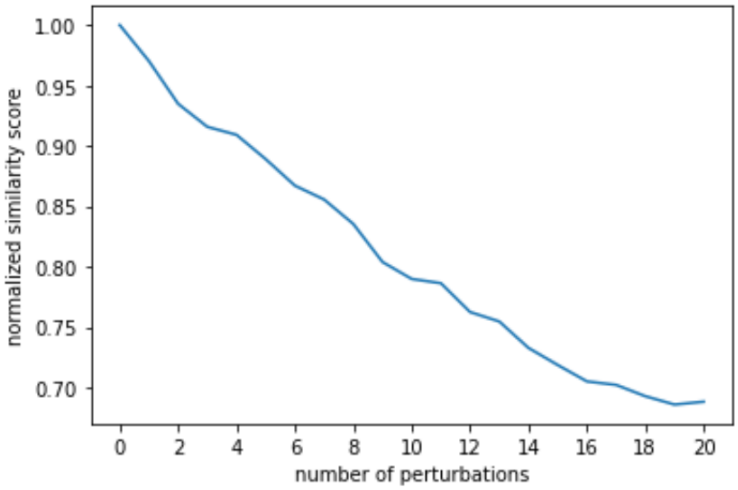

Figure 5.12: This curve shows the average relative decrease in similarity in the images classified by the network as belonging to the same class (this means, with a value of similarity above the threshold of 0.3) as the superpixels are perturbed in order according to their relevance.

Figure 5.13: Perturbation of superpixels in similar instances. The first image is the reference, while the second one is the test. The next images show the perturbation of the most important superpixels.

considered for calculating the average, and the selected size of the superpixels to be perturbed is 5x5.

The curve in Figure 5.12 has a similar shape to those shown in Figure 2.10, where the results for the original method proposed by Samek et al. [1] applied to LRP and SA explanations are depicted. This means that the behaviour exhibited is the expected if the method correctly explains the inferences of the network, i.e., the decrease of similarity is steeper at the beginning, as the most important superpixels are being perturbed at first. This very encouraging result suggests that our method is suitable for explaining SNN inferences for similar instances.

Figure 5.14 shows the curve that depicts the relative increase of similarity when perturbing the most important superpixels in the way described by the quality assessment method for dissimilar instances. The perturbation on each step is illustrated with an example in Figure 5.15. It is possible to see that although the similarity tends to increase on each perturbation, this increase is more irregular than the behaviour for similar instances shown in Figure 5.12.

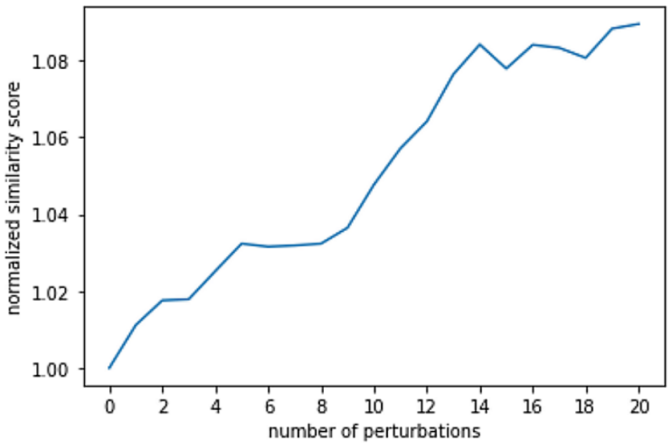

Figure 5.14: This curve shows the average relative increase in similarity in the images classified by the network as belonging to a different class (i.e., with a value of similarity below the threshold of 0.3) as the superpixels are perturbed in order according to their relevance.

Figure 5.15: Perturbation of superpixels in dissimilar instances. When assessing the quality of dissimilar instances, the method proceeds similarly to that depicted in Figure 5.13 for similar instances but perturbing both images in the pair, the reference, and the test.

5.5 Comparison with EMSNN

5.5.1 Reconstruction Network

To test how true is the statement of the improvement done by our proposed reconstruction network, we implemented an autoencoder similar to the one used in EMSNN. Its structure is shown in Table 5.2. In our results, the quality of the reconstructions and the similarity between the embedding and the bottleneck layer of the autoencoder varied widely depending on the parameters used. For example, a high value of γ in Equation 3.1 led to better reconstructions, but the bottleneck layers are highly different from the embeddings. We tried different parameter values until the loss given by $Lrecon_a$ in Equation 3.1 was similar to the loss obtained by our decoder. When this happened, the loss given by *Lclose* was 0.012. Therefore, when the same reconstruction quality was attained, there was still a difference between the embedding and the bottleneck layer, something that is not a problem in our proposed architecture that is directly

trained with the embeddings. As a remark, an advantage of using our proposed decoder instead of EMSNNs' autoencoder is that it is no longer necessary to define any parameters for training.

Layer	Output
Input	40x40x3
Flatten	4800
Dense (bottleneck layer)	512
Dense	4800
Reshape	40x40x3

Table 5.2: *Autoencoder architecture*

5.5.2 Use of a prototype

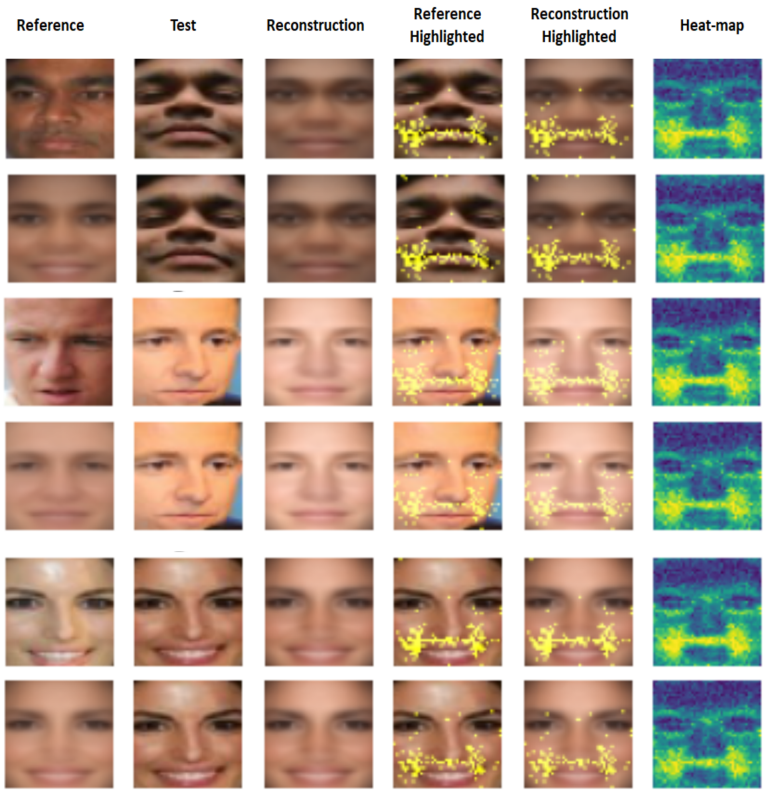

Figure 5.16: Experiments for assessing the application of a prototype as a reference. On odd rows, an arbitrary instance for the same class of the test was used as reference. On even rows, the prototype embedding was used as reference. In these cases, the reconstruction given by the decoder when using the prototype embedding as input is shown in the first column. Because a set of images is needed for creating the prototype, only instances with at least five other images of the same class available on the dataset were selected. The highlighted pixels are almost the same in both cases, suggesting that there is no significant advantage in using a prototype.

In these experiments, we kept all the steps of our method with the only difference that instead of using another image as reference we used the prototype of the class of the test image, as proposed in EMSNN. The prototype of a class is built by taking the average of all the embeddings of the instances belonging to that class. In Figure 5.16, the odd rows show the explanations of our method by comparing two images of the same class, while the even rows show the results obtained by comparing the test instance with its respective prototype.

The most remarkable result of this experiment is that the masks and the heat-maps obtained whether by comparing a test image with the prototype or with another instance of the same class are almost the same. Thus, this is another argument that sustains that using a prototype to implement the method is a disadvantage.

An explanation for this is that, as stated in the previous section, it seems that when instances are similar the image features are similarly encoded in the embedding features. This means that the information in the prototype might be encoded similarly to any of the embeddings belonging to the class.

5.5.3 Randomly perturbing the features by selecting the k-top of the ranking

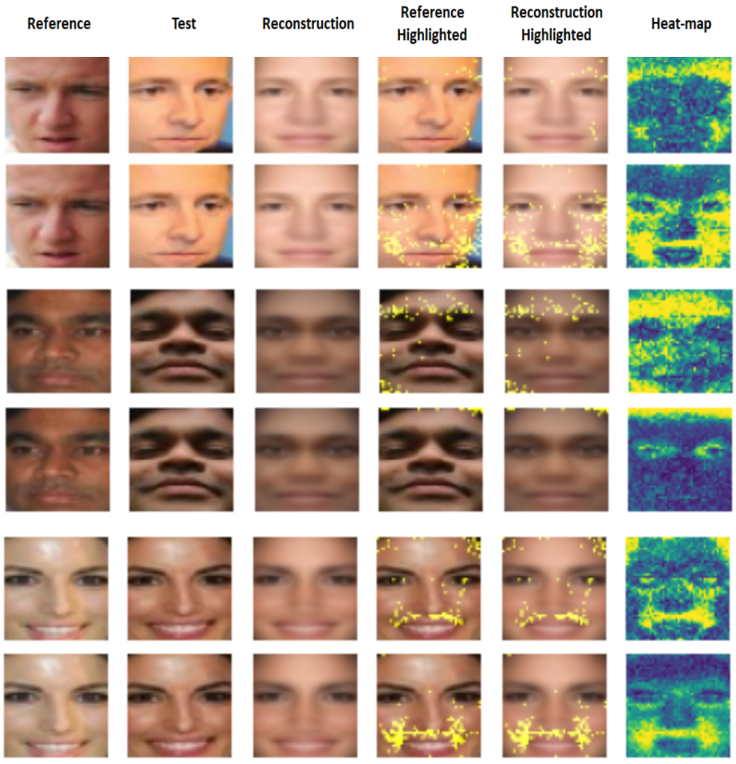

Figure 5.17: Results provided by the proposed method when using a k-top feature ranking. In odd columns, the 20 most important features are perturbed as indicated in EMSNN, while the top 100 features are selected in the even columns. It can be seen that the highlighted pixels differ significantly in both cases, showing the strong influence of the value assigned to this parameter.

Based on the feature distance, our method assigns a relevance measure according to which each feature is perturbed. In contrast, EMSNN selected a k number of the closest features to create N embeddings randomly perturbed on those most important features according to a normal distribution. We tried a variation of our proposal in which the perturbation step is done according to EMSNN. The resulting masks and heat-maps are shown in Figure 5.17 for two values of the parameter k: 20 and 100. The parameter N was set to 50. Results varying this parameter are not shown because we observed that the results did not change significantly above a certain value. The results can be contrasted with the ones obtained by following our original method, as the same images were used in the previously presented Figure 5.16. The results of this variation are not only different from those obtained by following the "pure" method. Still, they also vary widely depending on the value chosen for the parameter k, which is a severe problem regarding the method's reproducibility.

To measure which explanations were better, we compared the curves obtained by applying the quality assessment method in the original proposal and on its variation with $k = 100$. The resulting curves are shown in Figure 5.18. Our method performed significantly better, as the similarity on the other method decreases in a way that is not expected if the explanations were good, i.e., the decrease is less steep at the beginning and in general it becomes steeper with later perturbations.

Another disadvantage of applying the altered method is that it is slower than our original proposal, as the perturbation and reconstruction steps had to be repeated N times. In contrast, these steps are only performed once for each explanation in the original method.

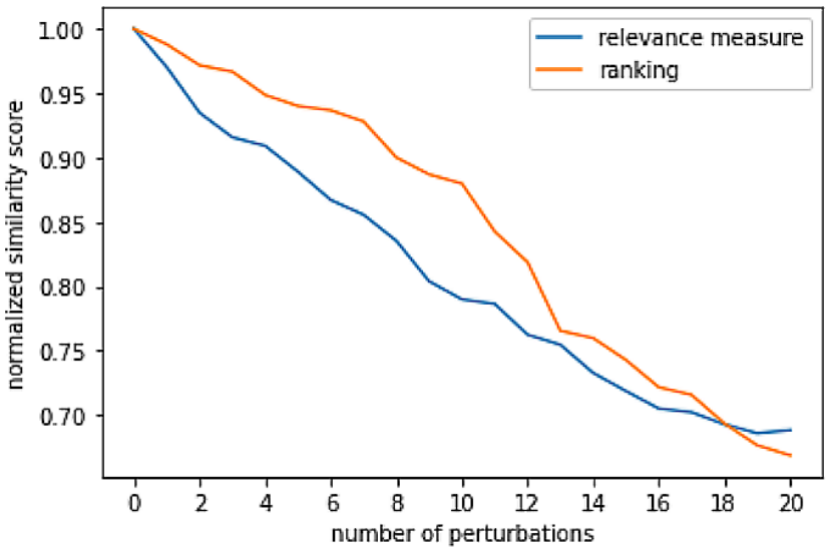

Figure 5.18: Comparison of the decrease in similarity score when perturbing by a relevance measure (as suggested by our proposal) or by randomly perturbing just a number of the features ranked as most important (as proposed in EMSNN). For the last case, the top 100 features were selected.

Chapter 6

Conclusions

Overall, it is possible to state that all the goals of this work were achieved. Considering separately each one of the objectives referred in Chapter 1, we conclude that:

- We reviewed the fundamental concepts for understanding deep neural networks in general, and CNNs and SNNs in particular. Also, we briefly presented the history, motivation, and core concepts of explainable artificial intelligence. Despite the importance of SNNs, and the current interest in explaining DNN inferences, we only found two methods in the literature that try to explain their inferences.

- We designed our explainability method by considering the advantages of the existing ones and carefully reflecting on their limitations to overcome them. **The main novelty of our method is that it can explain both possible cases: whether the images are considered to be similar or dissimilar by the SNN**. Compared to EMSNN, it is less complex, and also relies on fewer parameters.

- We adapted the method for quality assessment described in Chapter 2 to be applicable to SNN inferences. As a remarkable result, when explaining similar instances that were correctly classified by the network, we observed the desired behaviour in the assessment of its quality. This behaviour is also observed when explaining the correct inferences on instances of different classes but in a less apparent way, because the trend followed by the curve is not as clear as in the former case. Finding an explanation for this phenomenon, and ideally overcoming this problem, could be a matter for future works. The quality assessment method could also be useful for comparing the performances of future proposals of explainability in SNN.

- Our proposal is successfully applied to CelebA dataset. These data are of higher complexity than those used in the existing methods. The specific architecture in which our proposal was applied is FaceNet, even though, because it is post-hoc, it is general enough to be applied to other SNN models.

- Because it relies on fewer parameters, the results obtained by our method are easier to reproduce than those of EMSNN. Besides, by applying the quality assessment method, by using our proposal we observed a steeper descent in similarity in the first iterations. This behaviour is expected if the explanations are of better quality.

6.1 Future works

Possible works that our proposals could inspire are:

- Test the method in different SNN architectures and other kinds of image data. Of special interest would be applying it to numbers of the MNIST dataset so that the masks obtained could be compared with the ones in the paper by Utkin et al. [18], where EMSNN first appeared.
- Improve the representations obtained with the decoder, and evaluate its impact in the quality of the explanations given.
- Adapt this method to account for other types of data, such as tabular or graph data. By doing this, it would be possible to compare the obtained method with the one proposed by Chen et al. [17].
- Try modifications in our method to improve its quality. These modifications could be inspired by ideas from previous works. For example, as done in [17], global salient features could be used to constrain the features obtained in the mask.
- Propose different methods to assess the quality of explanations in SNN.

Bibliography

[1] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," *arXiv preprint arXiv:1708.08296*, 2017.

[2] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a siamese time delay neural network," *Advances in neural information processing systems*, vol. 6, pp. 737–744, 1993.

[3] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[4] D. Chicco, "Siamese neural networks: An overview," *Artificial Neural Networks*, pp. 73–94, 2021.

[5] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[6] A. Adadi and M. Berrada, "Peeking inside the black-box: a survey on explainable artificial intelligence (xai)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.

[7] S. Vollert, M. Atzmueller, and A. Theissler, "Interpretable machine learning: A brief survey from the predictive maintenance perspective," in *2021 26th IEEE international conference on emerging technologies and factory automation*, 2021, pp. 1–8.

[8] J. V. Jeyakumar, J. Noor, Y.-H. Cheng, L. Garcia, and M. Srivastava, "How can i explain this to you? an empirical study of deep neural network explanation methods," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4211–4222, 2020.

[9] L. Rieger, C. Singh, W. Murdoch, and B. Yu, "Interpretations are useful: penalizing explanations to align neural networks with prior knowledge," in *International conference on machine learning*, 2020, pp. 8116–8126.

[10] J. K. Winkler, C. Fink, F. Toberer, A. Enk, T. Deinlein, R. Hofmann-Wellenhof, L. Thomas, A. Lallas, A. Blum, W. Stolz *et al.*, "Association between surgical skin markings in dermoscopic images and diagnostic performance of a deep learning convolutional neural network for melanoma recognition," *JAMA dermatology*, vol. 155, no. 10, pp. 1135–1141, 2019.

[11] N. Garg, L. Schiebinger, D. Jurafsky, and J. Zou, "Word embeddings quantify 100 years of gender and ethnic stereotypes," *Proceedings of the National Academy of Sciences*, vol. 115, no. 16, pp. E3635–E3644, 2018.

[12] J. Dressel and H. Farid, "The accuracy, fairness, and limits of predicting recidivism," *Science advances*, vol. 4, no. 1, p. eaao5580, 2018.

[13] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information fusion*, vol. 58, pp. 82–115, 2020.

[14] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery." *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[15] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, and K.-R. Müller, "Explaining deep neural networks and beyond: A review of methods and applications," *Proceedings of the IEEE*, vol. 109, no. 3, pp. 247–278, 2021.

[16] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015.

[17] C. Chen, Y. Shen, G. Ma, X. Kong, S. Rangarajan, X. Zhang, and S. Xie, "Self-learn to explain siamese networks robustly," in *2021 IEEE International Conference on Data Mining*, 2021, pp. 1018–1023.

[18] L. Utkin, M. Kovalev, and E. Kasimov, "An explanation method for siamese neural networks," in *Proceedings of International Scientific Conference on Telecommunications, Computing and Control*. Springer, 2021, pp. 219–230.

[19] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[20] S. Chanda, A. C. GV, A. Brun, A. Hast, U. Pal, and D. Doermann, "Face recognition-a one-shot learning perspective," in *2019 15th International Conference on Signal-Image Technology & Internet-Based Systems*, 2019, pp. 113–119.

[21] D. Gibelli, Z. Obertová, S. Ritz-Timme, P. Gabriel, T. Arent, M. Ratnayake, D. De Angelis, and C. Cattaneo, "The identification of living persons on images: A literature review," *Legal medicine*, vol. 19, pp. 52–60, 2016.

[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[24] J. Zou, Y. Han, and S.-S. So, "Overview of artificial neural networks," *Artificial Neural Networks*, pp. 14–22, 2008.

[25] H. Ramchoun, Y. Ghanou, M. Ettaouil, and M. A. Janati Idrissi, "Multilayer perceptron: Architecture optimization and training." International Journal of Interactive Multimedia and Artificial Intelligence, 2016, pp. 26–30.

[26] Perez-Enciso and L. Zingaretti, "A guide for using deep learning for complex trait genomic prediction," *Genes*, vol. 10, pp. 553–567, 2019.

[27] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[28] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[30] N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," in *2017 international conference on communication and signal processing*, 2017, pp. 0588–0592.

[31] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[32] A. Hsiang, A. Brombacher, M. Rillo, M. Vautravers, S. Conn, S. Lordsmith, A. Jentzen, M. Henehan, B. Metcalfe, I. Fenton, B. Wade, L. Fox, J. Meilland, C. Davis, U. Baranowski, J. Groeneveld, K. Edgar, A. Movellan, T. Aze, and P. Hull, "Endless forms: 34,000 modern planktonic foraminiferal images for taxonomic training and automated species recognition using convolutional neural networks," *Paleoceanography and Paleoclimatology*, vol. 34, 07 2019.

[33] D. Grattarola, "Deep feature extraction for sample-efficient reinforcement learning," Ph.D. dissertation, Thesis, University of Lugano, 2017.

[34] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 539–546.

[35] L. Zheng, S. Duffner, K. Idrissi, C. Garcia, and A. Baskurt, "Pairwise identity verification via linear concentrative metric learning," *IEEE transactions on cybernetics*, vol. 48, no. 1, pp. 324–335, 2016.

[36] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European conference on computer vision*, 2016, pp. 499–515.

[37] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," 2015.

[38] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.

[39] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 212–220.

[40] X. Dong and J. Shen, "Triplet loss in siamese network for object tracking," in *Proceedings of the European conference on computer vision*, 2018, pp. 459–474.

[41] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 1735–1742.

[42] M. Van Lent, W. Fisher, and M. Mancuso, "An explainable artificial intelligence system for small-unit tactical behavior," in *Proceedings of the national conference on artificial intelligence*, 2004, pp. 900–907.

[43] J. D. Moore and W. R. Swartout, "Explanation in expert systemss: A survey," University of southern California Marina del Rey, Information Sciences Institute, Tech. Rep., 1988.

[44] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *2018 41st International convention on information and communication technology, electronics and microelectronics*, 2018, pp. 0210–0215.

[45] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller, "Evaluating the visualization of what a deep neural network has learned," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 11, pp. 2660–2673, 2016.

[46] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.

[47] M. Tschannen, O. Bachem, and M. Lucic, "Recent advances in autoencoder-based representation learning," *arXiv preprint arXiv:1812.05069*, 2018.

[48] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI conference on artificial intelligence*, 2017.

[49] A. R. Lahitani, A. E. Permanasari, and N. A. Setiawan, "Cosine similarity to determine similarity measure: Study case in online essay assessment," in *4th International Conference on Cyber and IT Service Management*, 2016, pp. 1–6.

[50] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *IEEE international conference on automatic face & gesture recognition*, 2018, pp. 67–74.

[51] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[52] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision*, December 2015.