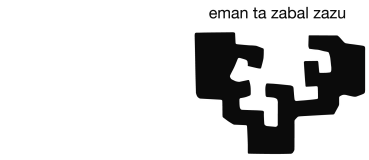

Máster Universitario en Ingeniería Computacional y Sistemas Inteligentes

Master Thesis

Image-based Family Verification in the wild

Oscar Serradilla Casado

Director:
Fadi Dornaika

Co-director:
Ignacio Arganda Carreras

2017

Abstract

Facial image analysis has been an important subject of study in the communities of pattern recognition and computer vision. Facial images contain much information about the person they belong to: identity, age, gender, ethnicity, expression and many more. Visual kinship recognition is a new research topic in the scope of facial image analysis which is essential for many real-world applications (e.g., kinship verification, automatic photo management, social-media application, and more). However, nowadays there exist only a few practical vision systems capable to handle such tasks. We propose a flexible pipeline composed by modules that assembled together improve the results obtained individually. Our optimized pipeline is based on deep feature extraction, feature selection, multi-metric learning and classifier blending. Our kinship verification system improves state-of-the-art results that do not use external data for the two public databases KinFace-I and KinFace-II.

Acknowledgements

I would like to thank my two tutors for directing this thesis. Specially, I would like to thank Ignacio Arganda Carreras for his support in the development and contribution of ideas and Fadi Dornaika for his guidance and contributions with his expert knowledge.

I would also like to thank my family and girlfriend for encouraging and supporting me during the thesis.

Contents

List of Figures

Table index

1 Introduction

Facial image analysis has been an important subject of study in the communities of pattern recognition and computer vision. Facial images contain much information about the person they belong to: identity, age, gender, ethnicity, expression and many more. For that reason, the analysis of facial images has many applications in real world problems such as face recognition, age estimation, gender classification or facial expression recognition.

Visual kinship recognition is a new research topic in the scope of facial image analysis. It is essential for many real-world applications (e.g., kinship verification [12] [13] [14], automatic photo management, social-media applications, and more). However, nowadays there exist only a few practical vision systems capable to handle such tasks. Hence, vision technology for kinship-based problems has not matured enough to be applied to real-world problems. This leads to a concern of unsatisfactory performance when attempted on real-world datasets.

Kinship verification is to determine pairwise kin relations for a pair of given images. It can be viewed as a typical binary classification problem, i.e., a face pair is either related by kinship or it is not [8][15][16][17]. Prior research works have addressed kinship types for which pre-existing datasets have provided images, annotations and a verification task protocol. Namely, *father-son*, *father-daughter*, *mother-son* and *mother-daughter*.

1.1 Objectives

The main objective of this Master work is the study and development of feature selection and fusion for the problem of family verification from facial images.

To achieve this objective, there is a main tasks that can be addressed: perform a comparative study on face descriptors that include classic descriptors as well as deep descriptors. The main contributions of this Thesis work are:

1. Studying the state of the art of the problem of family verification in images.

2. Implementing and comparing several criteria that correspond to different face representations (Local Binary Patterns (LBP), Histogram Oriented Gradients (HOG), deep descriptors).

3. Implementing a processing pipeline that can include feature selection, metric learning architecture for feature fusion (e.g., the metric used in [18]), Discriminant embedding, and classifier blending.

4. Evaluation of the performance on two public databases: KinFaceW-I and KinFaceW-II.

1.2 Related work

The kinship verification literature is mostly divided into two main types of approaches: feature-based and model-based solutions. Feature-based methods extract discriminative features that represent the original facial images to perform supervised or unsupervised learning. Model-based methods learn discriminative models to verify kin relationship from face pairs.

In the sequel, we describe some of the most relevant and recent feature-based methods. Fang *et al.* [19] used feature extraction and selection methods. They evaluated a set of low-level image features for kinship verification problem and used the most discriminative ones to build a K-Nearest-Neighbors (KNN) classifier. They also evaluated human performance on this problem. Zhou *et al.* [20] proposed a new spatial pyramid learning-based (SPLE) descriptor for face representation and applied support vector machine (SVM) for kinship verification. Guo *et al.* [21] developed a descriptor called DAISY, which was adapted to the problem of kinship verification from facial images that represented the salient features. They also developed a dynamic scheme to stochastically combine familial traits. Zhou *et al.* [22] presented a Gabor-based Gradient Orientation Pyramid (GGOP) descriptor representation method for kinship verification from facial images and used discriminative SVMs for classification. Kohli *et al.* [23] proposed an approach for feature representation termed as filtered contractive deep belief networks (fcDBN). The proposed descriptor representation encoded relational information present in images using filters and contractive regularization penalty. Kohli *et al.* [24] presented a kinship classification algorithm that used the local description of the pre-processed Weber face image as self-similarity representation. Yan *et al.* [25] proposed a prototype-based discriminative feature learning (PDFL) method for kinship verification. They also proposed multiview PDFL method to learn multiple mid-level features that could characterize better the kin relation of face images for kinship verification. Lu *et al.* [26] proposed a compact binary face descriptor (CBFD) feature learning method for face representation and recognition. They also proposed a coupled CBFD (C-CBFD) method by reducing the modality gap of

heterogeneous faces at the descriptor level to make their method applicable to heterogeneous face recognition. Lu *et al.* [27] proposed a discriminative multimanifold analysis (DMMA) method by learning discriminative features from image patches. Dehghan *et al.* [28] proposed an algorithm that fused the features and metrics discovered via gated autoencoders with a discriminative neural network layer that learned the optimal features to delineate parent-offspring relationships. Dibeklioglu *et al.* [29] proposed a method that employed facial expression dynamics by using features that described facial dynamics and spatio-temporal appearance over smile expressions; with the objective of recognizing kinship by resemblance of facial expressions.

Regarding the most relevant and recent model-based methods: Lu *et al.* [8] proposed a new neighborhood repulsed metric learning (NRML) method for kinship verification. To make better use of multiple descriptors to extract complementary information, they also proposed a multiview NRML (MNRML) method that sought a common distance metric that provides a linear projection for the multiple features that can be fused at the classification level. Yan *et al.* [15] proposed a new discriminative multimetric learning method for kinship verification via facial image analysis. They extracted multiple features using different face descriptors to jointly learn multiple distance metrics. Hu *et al.* [13] proposed a new large margin multi-metric learning (LM^3L) method for face and kinship verification in the wild. It jointly learned multiple distance metrics under which the correlations of different feature representations of each sample were maximized. Zhou *et al.* [17] proposed a new Ensemble similarity learning (ESL) for this problem. First, a sparse bilinear similarity function was used to model the relative characteristics encoded in kin data. Then, ESL learned from kin dataset by generating an ensemble of similarity models with the aim of achieving strong generalization ability. Xia *et al.* [14] developed a transfer subspace learning based algorithm in order to reduce the significant differences in the appearance distributions between children and old parents facial images. They also proposed an algorithm to predict the most likely kin relationships embedded in an image. Xia *et al.* [30] proposed an extended transfer subspace learning method aiming at mitigating the enormous divergence of distributions between children and old parents. The idea was to utilize an intermediate distribution close to both the source and target distributions to bridge them and reduce the divergence.

There are different reference databases for the field of kinship verification from facial images publicly available. Namely, KinFace-I and KinFace-II databases [8], Cornell Kinface [19] and UB KinFace database [14]. Several authors also proposed their own databases such as the IIITD kinship database [24], UvA-NEMO Smile[29] or WVU kinship database

[23].

2 Artificial Neural Networks (ANN)

Artificial Neural Networks are an important technology used to extract features from images in this proposal. This section describes what Neural Networks are and their applications, based on the book by Nielsen *et al.* [1]. The most relevant parts of the book have been summarized and a few paragraphs extracted as stated in the book. Most of the images and all the equations have also been taken from the book.

2.1 Perceptrons

So as to understand what a neural network is, the concept of perceptron should be learned. Perceptrons were developed in the 1950s and 1960s by Frank Rosenblatt, inspired by earlier work by Warren McCulloch and Walter Pitts.

Perceptron are linear discriminators that take several inputs and produce a single binary output. Figure 1 shows a perceptron with three inputs (x_1, x_2 and x_3) and the output:

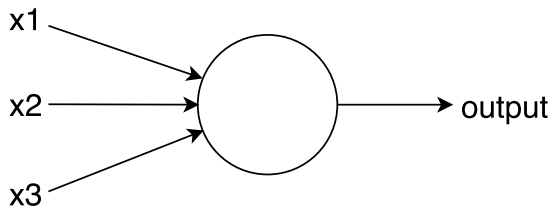

Figure 1: Neuron with three inputs and an output. Figure from [1].

The output is computed by using weights which are real numbers that express the importance of each input for the output. The neuron's output is determined by the sum of the multiplication of each input by its weight. If this sum is higher lower or equal to a specific threshold, the output value is 0 and 1 otherwise. This is formally expressed by the equation (1)

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases} \tag{1}$$

This threshold was replaced by the term bias, $b \equiv -\text{threshold}$, resulting in the following equation (2)

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \tag{2}$$

Perceptron networks can be created by assembling perceptrons, see Figure 2.

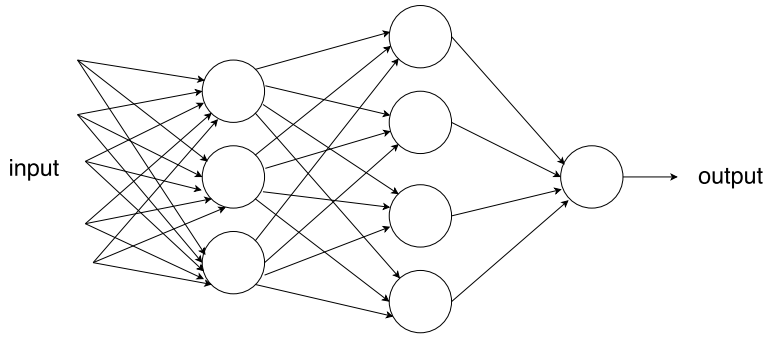

Figure 2: Perceptron network with three neurons as input, four intermediate neurons and one neuron as output. Figure from [1].

These networks can be used as binary classifiers to model linear functions or pattern recognition learning. To do so, learning algorithms are used, which set the network's weights and biases so that they outputs the desired result.

The problem with perceptron networks is that when one of their weights changes a little bit (in training stage), the behavior of the network can change a lot, so it is very difficult to tweak in order to model a specific behavior. This problem is solved using non-linear neurons.

2.2 Sigmoid neurons

In sigmoid neuron networks, unlike perceptrons, a small change in any weight of the network or bias causes a small change in the output of the network. Its scheme is the same as the perceptron neuron. It has inputs (x_1,x_2...), weights and an output, see Figure 1.

However, the sigmoid neuron can also output any value between 0 and 1, defined by the equation (3), where $z = w \cdot x + b$

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}. \tag{3}$$

2.3 Feedforward networks

Feedforward networks are divided in layers (see Figure 3). The left layer is the input layer, where the input data is fed to the network. The right layer is the output layer, which contains the output values computed by the network for the input in the input layer. All the intermediate layers are called hidden layers, which connect the input and the output layers. Feedforward networks do not have loops, all the information flows forward.

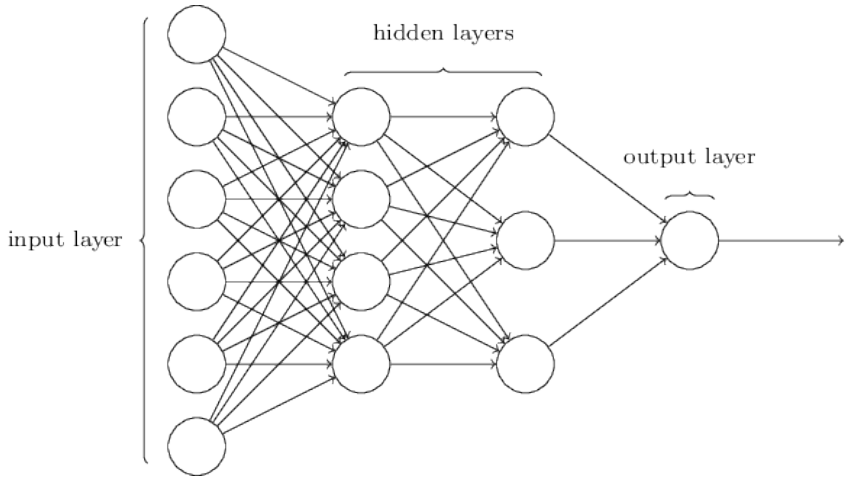

Figure 3: Neural Network formed by the input layer, two hidden layers and the output layer. Figure from [1].

Feedforward networks with non-linear activation functions that have at least two hidden layers can model any function or behavior. This is because they can create the universal approximator if the intermediate layers have enough neurons [31].

2.3.1 Gradient descent and cost function

In order to create the learning algorithm, first the cost or objective function must be defined. This indicates how far the network is to obtain the desired output. One example cost function is the *mean squared error* (MSE), defined by the following equation:

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2. \tag{4}$$

w denotes the collection of all weights in the network, b all the biases, n is the total number of training inputs, a is the vector of outputs from the network when x is input,

y(x) is the desired vector (regression values or probability distribution) and the sum is over all training inputs, x.

The aim of the training algorithm is to minimize the cost $C(w,b)$ as a function of weights and biases. Usually, this cost is minimized using *gradient descent* algorithms.

The gradient descent algorithm is a general concept for minimizing cost functions. In the literature of neural network learning, one can use also gradient stochastic descent methods that is very suited to online learning that uses one training examples or a mini batch of training examples.

Imagine the cost function as if it were a set of mountains and valleys and the objective is to find its lowest point, denoted as global minimum. It can be reached by selecting the best variables setting. See the simplification of only two variables and the cost function value (height) in Figure 4

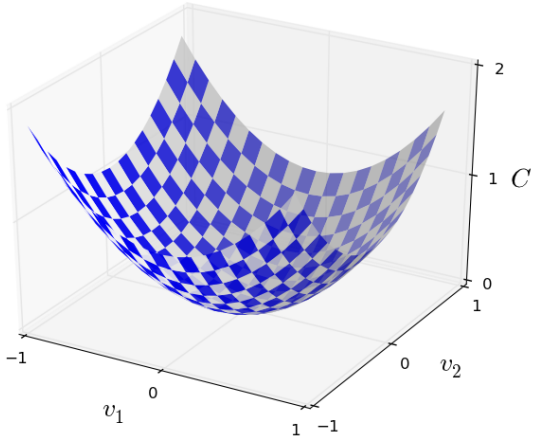

Figure 4: Cost function as a valley. *v1* and *v2* variables are represented as width depth respectively while C cost is represented by height. Figure from [1].

An initial network starts from any part of the cost function and it should find a minimum. To do so, the learning algorithm gives a small step in each variable's value. It behaves as a ball rolling from a mountain to the lowest point. The way to represent a small move Δv_1 in the v_1 direction, and a small move Δv_2 in the v_2 direction with regard to cost C is given by the following equation:

$$\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2. \tag{5}$$

Δv_1 and Δv_2 are chosen to make ΔC negative. The gradient vector of C is given by: $\Delta v \equiv (\Delta v_1, \Delta v_2)^T$ in equation (6):

$$\nabla C \equiv \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}\right)^T. \tag{6}$$

Previous equations allow to define a very important equation (7), which allows to choose Δv so as to make ∇C negative:

$$\Delta C \approx \nabla C \cdot \Delta v. \tag{7}$$

Here, a new concept arises called learning rate. It is the size of the step taken to find the global minimum, which is defined by the following equation:

$$\Delta v = -\eta \nabla C, \tag{8}$$

The final equation that defines the way of moving for the gradient descend is as follows:

$$v \rightarrow v' = v - \eta \nabla C. \tag{9}$$

All in all, the backpropagation work is to repeatedly compute the gradient ∇C, and then to move in the opposite direction, "falling down" the slope of the valley.

Returning to the scope of neural networks, the gradient descent method can be applied as follows: the previously called "position" now has components w_k and b_l, and the gradient vector ∇C has corresponding components $\partial C/\partial w_k$ and $\partial C/\partial b_l$. Writing out the gradient descent update rule in terms of components we obtain

$$w_k \rightarrow w_k' = w_k - \eta \frac{\partial C}{\partial w_k} \tag{10}$$

$$b_l \rightarrow b_l' = b_l - \eta \frac{\partial C}{\partial b_l}. \tag{11}$$

By repeatedly applying this update rule, the cost function can decrease until, hopefully, finding a minimum value. This is the way gradient descent method is used to learn a neural network.

2.3.2 Backpropagation

Backpropagation is the algorithm needed to compute the gradient of the cost function [32]. In addition to being a fast algorithm to learn neural networks, backpropagation gives detailed insights about how changing their weights and biases change the overall behavior of the network. The rest of this section is about how the algorithm works.

The following notation will be used in the rest of the section: w^l_{jk} denotes the weight for the connection from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer. See Figure 5

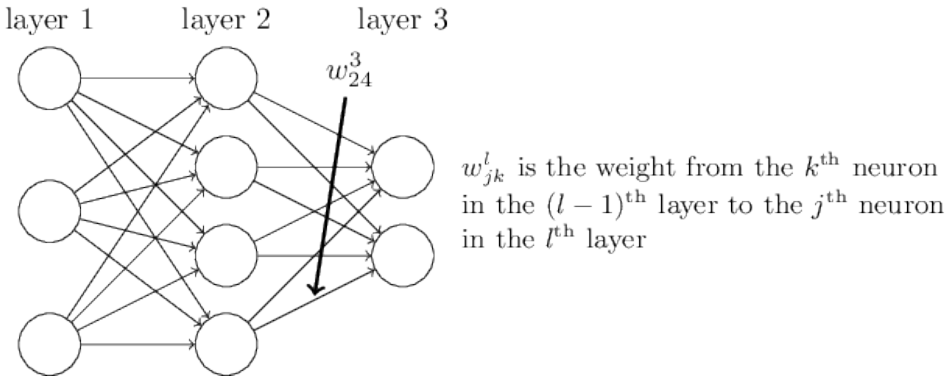

Figure 5: Neural network notation: layers weights and connections. Figure from [1].

Extending the notation, the activation a^l_j of the j^{th} neuron in the l^{th} layer is related to the activations in the $(l-1)^{\text{th}}$ layer by equation (12)

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right), \tag{12}$$

The last expression can be represented in matrix form so that calculus are done faster, in the following way: w^l is defined as a weight matrix for each layer, l. The entries of the weight matrix are just the weights connecting to the l^{th} layer of neurons, that is, the entry in the j^{th} row and k^{th} column is w_{jk}^l. Similarly, for each layer l there is a bias vector b^l. The components of the bias vector are the values b_j^l, one component for each neuron in the l^{th} layer. The activation vector a^l is formed by the activations a_j^l. $\sigma(v)$ is defined as $\sigma(v)_j = \sigma(v_j)$.

The equation (12) can be rewritten in matrix form, see equation (13). It gives a global view of thinking about how activations of one function relate to the activations in the previous layer: applying weight matrices to the activations, adding the bias vector and them applying the σ function

$$a^l = \sigma(w^l a^{l-1} + b^l). \tag{13}$$

The goal of backpropagation is to compute the partial derivatives $\partial C/\partial w$ and $\partial C/\partial b$ of the cost function C with respect to any weight w or bias b in the network. In our example, the cost function is given by equation (4).

Two assumptions must be made about the cost function. The first one is that it can be written as an average $C = \frac{1}{n}\sum_x C_x$ over cost functions C_x for individual training examples x. In the case of quadratic cost function, it can be represented as $C_x = \frac{1}{2}\|y - a^L\|^2$; where L denotes the final layer of the net. The second assumption made is that the cost can be written as a function of the outputs from the neural network (see Figure 6)

This way, the quadratic cost function can be written as

$$C = \frac{1}{2}\|y - a^L\|^2 = \frac{1}{2}\sum_j (y_j - a_j^L)^2. \tag{14}$$

Backpropagation is about understanding how changing the weights and biases in a net-

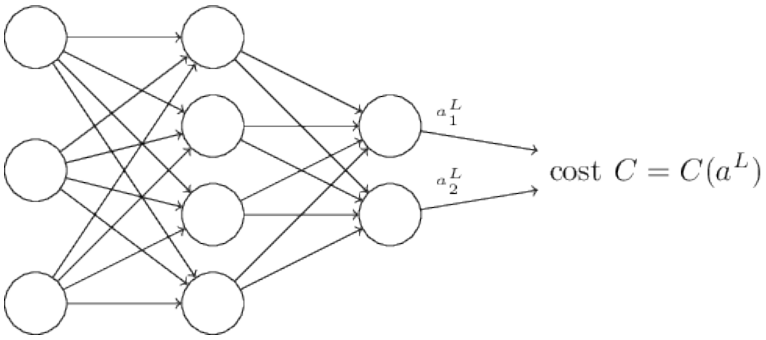

Figure 6: Cost represented as output of a neural network. Figure from [1].

work changes the cost function. Ultimately, this means computing the partial derivatives $\partial C/\partial w^l_{jk}$ and $\partial C/\partial b^l_j$. But to compute those, first an intermediate quantity δ^l_j, which is the error in the j^{th} neuron in the l^{th} layer, must be computed.

Backpropagation starts from the last layer and propagates till the first layer recursively. All in all, backpropagation consists on four equations that can be interpreted using the notation presented in this section. The equations are the following: (15), (16), (17) and (18). In order to see their proof and development, consult [1], or this tutorial [33].

$$\delta^L_j = \frac{\partial C}{\partial a^L_j}\sigma'(z^L_j). \tag{15}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l), \tag{16}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j. \tag{17}$$

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j. \tag{18}$$

In the equation (16), $\odot$ means element wise product.

2.3.3 Convolutional Neural Network (CNN)

Convolutional Neural Networks are a type of feed-forward networks. They are biologically inspired by the animal visual cortex and have applications in image and video recognition, recommender systems and natural language processing. They had a big impact in those scopes, improving state of the art results [3].

CNNs are designed to recognize visual patterns directly from pixel images with minimal preprocessing. They can recognize patterns with extreme variability (such as handwritten characters), and with robustness to distortions and simple geometric transformations. They work by applying $K \times K$ convolutional filters w throughout the image (like a sliding window), encoding the information of the middle pixel by using values of pixels on its neighborhood (its surrounding KxK pixels including itself). Figure 7 shows that process graphically.

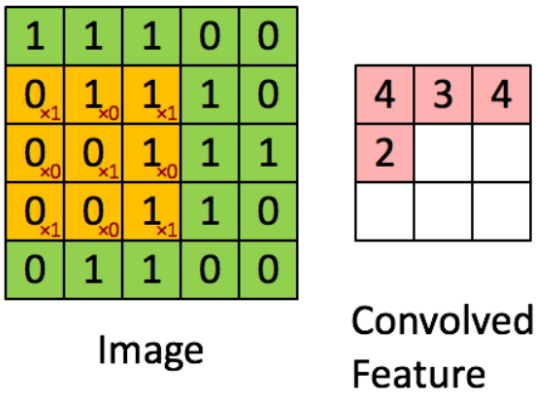

Figure 7: Convolution kernel applied pixel-by-pixel to the input image. The yellow squares represent the neighborhood that, when applied the kernel, create the number 2 of the convolved feature on the right. The image to be convolved is a binary image (its values are the big values that appear on all the cells of the image) and the weights of the kernel (appear as x0 or x1 on the yellow cells but they are also applied to the rest of the image by sliding it through the image) are also binary. The result of the convolution of the first 4 neighborhoods is recorded on the convolved feature table, remaining the last 5 neighborhoods. Figure from [2].

Normally, a convolutional layer consists of a linear convolution followed by a non-linear activation function (such as sigmoid or Rectified Linear Unit (ReLU), which is one of the most used because solves the problem of the vanishing gradient while using back-propagation) and a max-pooling; which selects the biggest intensity value of small pixel neighborhoods, thus reducing the size of the filters output. Normally, after several convolutional layers, fully connected layers are applied. When performing classification, the

last fully connected layer can be a softmax, making the output of the network represent the probability of each class (because the sum of network's output values is 1). Figure 8 shows a typical CNN architecture.

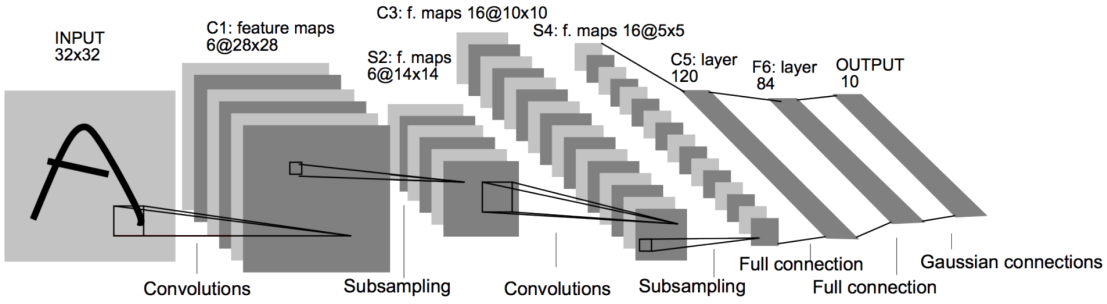

Figure 8: Lenet CNN architecture. Figure from [3].

2.3.4 Deep learning

Deep learning is the name given to the neural networks with more than 3 hidden layers. Currently it is a very powerful tool to accomplish state-of-the-art results in many fields. Thanks to Graphical Processing Units (GPUs), programs that split the data and use this components obtain a good way to train complex neural networks.

The advantage of deep learning is that, even if a neural network can be the universal approximator using only two hidden layers, adding more hidden layers can result in less neurons and time to process and train.

3 Face descriptors

This section will briefly describe some image descriptors that are very often used for extracting face features. We present two traditional hand-crafted descriptors: LBP and HOG. However, as recently handcrafted approaches have been outperformed by the last generation of Convolutional Neural Networks, we also present the two networks we used: VGG-Face and VGG-F. These pre-trained networks provide deep descriptors that can later be used for classification.

3.1 Local Binary Patterns (LBP)

Local Binary Patterns (LBP) [34], [4] is a descriptor that has been widely used in image processing for texture and face analysis. Its main advantage is that it is robust against monotonic gray-scale changes, so it is robust against illumination variations. Moreover, it is computationally simple, what makes it suitable for real-time image processing.

This descriptor works on gray images. It divides an image into several small regions (neighborhoods of pixels) from which the features are extracted. For each neighborhood, the central pixel is encoded by comparing its value to the value of its neighbors, creating a binary cell that contains 0 in the places were it is smaller and 1 otherwise. Figure 9 shows the steps to extract LBP descriptor from an image.

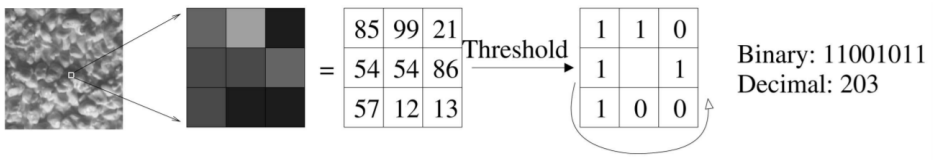

Figure 9: LBP traditional descriptor scheme. From left to right: input image from which LBP is extracted; 3x3 pixel window of the input image; intensity values of those pixels (usually between 0 and 255); image obtained by applying the threshold (assigning 1 to the pixels that have an intensity value equal to or larger than the pixel in the middle and 0 otherwise); decimal value of the last picture encoded in binary and transformed to decimal (in the case of these pixels it indicates that their LBP value is 203). Figure from [4].

Those binary cells represent the LBP descriptor for their neighborhood, so by concatenating these features for all the neighbors of the image, the final LBP descriptor representation is obtained. LBPs is the histogram of the generated codes over the image.

3.2 Histogram of Oriented Gradients (HOG)

Histogram of Oriented Gradients (HOG) [5] is a descriptor that has been widely used in image processing for object detection and recognition. Its main advantage is that it invariant to geometric and photometric transformations although vulnerable to object orientation.

This descriptor works on gray images. It divides an image into several small overlapping regions (neighborhoods of pixels) from which the features are extracted. For each neighborhood, the centered horizontal and vertical gradients are computed, together with their orientation and magnitude. These are the data that will represent each cell as an histogram,

so they are calculated all over the image and concatenated to form the HOG descriptor representation of the image. See Figure 10

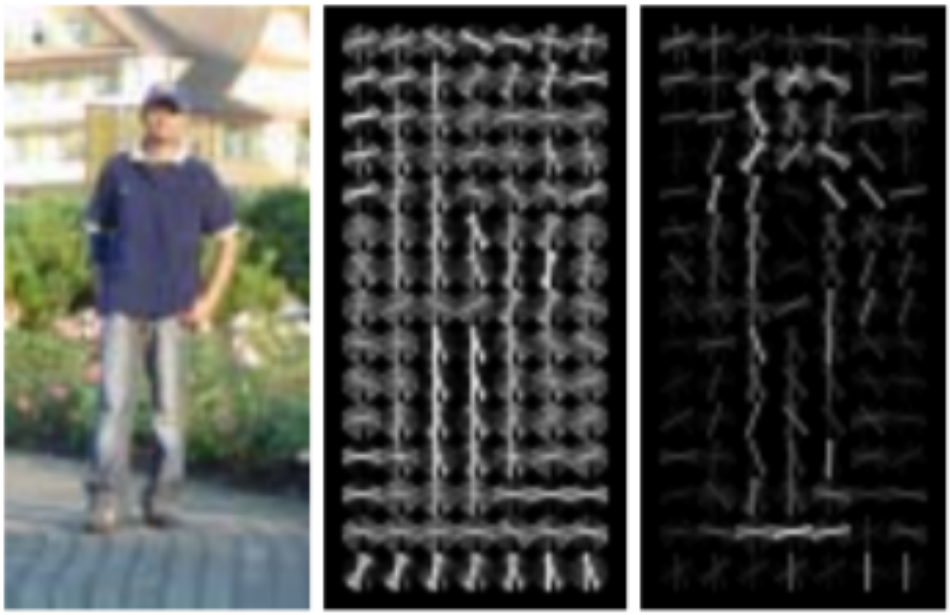

Figure 10: HOG traditional descriptor scheme. The left image contains a picture of a person which is aimed to be detected. The center picture is the computed rectangular HOG (R-HOG, one of the two main block geometries for computing the HOG image descriptor) descriptor of the left picture. The right picture is the R-HOG descriptor weighted by respectively the positive and the negative SVM weights presented in the paper [5], which is the original source of the picture.

3.3 VGG-Face descriptors

This pre-trained deep Convolutional Neural Network provides descriptors specifically trained for face recognition [6].

The architecture of the network is formed by 11 blocks, each containing a linear operator followed by one or more non-linearities such as ReLU and max pooling (see Figure 11). The first 8 such blocks are linear convolutional, while the last 3 are fully connected with ReLU activation function. The input image is RGB and its size is 224×224 pixels. The first 2 fully connected layers contain $4,096$ neurons and the last one reduces the dimensionality to $2,622$ (number of classes of the training dataset). The network was trained in 2.6M images. The data was further augmented by flipping the image left to right with 50% probability, but did not perform any color channel augmentation.

The hyperparameter setting that was used to train this network is the following: optimiza-

layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
type	input	conv	relu	conv	relu	mpool	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv
name	–	conv1_1	relu1_1	conv1_2	relu1_2	pool1	conv2_1	relu2_1	conv2_2	relu2_2	pool2	conv3_1	relu3_1	conv3_2	relu3_2	conv3_3	relu3_3	pool3	conv4_1
support	–	3	1	3	1	2	3	1	3	1	2	3	1	3	1	3	1	2	3
filt dim	–	3	–	64	–	–	64	–	128	–	–	128	–	256	–	256	–	–	256
num filts	–	64	–	64	–	–	128	–	128	–	–	256	–	256	–	256	–	–	512
stride	–	1	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	2	1
pad	–	1	0	1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	1

layer	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
type	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	softmx
name	relu4_1	conv4_2	relu4_2	conv4_3	relu4_3	pool4	conv5_1	relu5_1	conv5_2	relu5_2	conv5_3	relu5_3	pool5	fc6	relu6	fc7	relu7	fc8	prob
support	1	3	1	3	1	2	3	1	3	1	3	1	2	7	1	1	1	1	1
filt dim	–	512	–	512	–	–	512	–	512	–	512	–	–	512	–	4096	–	4096	–
num filts	–	512	–	512	–	–	512	–	512	–	512	–	–	4096	–	4096	–	2622	–
stride	1	1	1	1	1	2	1	1	1	1	1	1	2	1	1	1	1	1	1
pad	0	1	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

Figure 11: VGG-Face architecture. Figure from [6].

tion is by stochastic gradient descent using mini-batches of 64 samples and momentum coefficient of 0.9; dropout of 0.5 after the first two fully connected layers and weight decay of $5 \cdot 10^{-4}$; initial learning rate 10^{-2}, which is decreased by a factor of 10, when the validation error stopped decreasing. The layers were initialized from a Gaussian distribution with a zero mean and variance equal to 10^{-2}. Biases were initialized to zero.

In order to use this network we removed the last layer, which performed dimensionality reduction for classification of the training dataset, thus obtaining a $4,096$ dimension feature vector. We also had to resize the input images of the database to have the required 224×224 dimensions.

3.4 ImageNet VGG-F descriptor

This pre-trained deep Convolutional Neural Network provides a descriptor specifically trained for image recognition and object detection [7]. The architecture of the network is formed by 8 learnable layers: 5 convolutional and the last 3 fully connected (see Figure 12). The input image is RGB and its size is 224×224 pixels. The first 2 fully connected layers contain $4,096$ neurons and the last one reduces the dimensionality to 1000 (number of classes of the training dataset). The network was trained on the ILSVRC-2012 database. Data augmentation was applied in the form of random crops, horizontal flips, and RGB color jittering.

Arch.	conv1	conv2	conv3	conv4	conv5	full6	full7	full8
CNN-F	64x11x11 st. 4, pad 0 LRN, x2 pool	256x5x5 st. 1, pad 2 LRN, x2 pool	256x3x3 st. 1, pad 1 -	256x3x3 st. 1, pad 1 -	256x3x3 st. 1, pad 1 x2 pool	4096 drop-out	4096 drop-out	1000 soft-max

Figure 12: VGG-F architecture. Figure from [7].

The hyperparameter setting that was used to train this network is the following: momen-

tum 0.9; weight decay $5 \cdot 10^{-4}$; initial learning rate 10^{-2}, which is decreased by a factor of 10, when the validation error stopped decreasing. The layers were initialized from a Gaussian distribution with a zero mean and variance equal to 10^{-2}.

In order to use this network we removed the last layer, which performed dimensionality reduction for classification of the training dataset, thus obtaining a $4,096$ dimension feature vector. We also had to resize the input images of the database to have the required 224×224 dimension.

4 Feature processing

This section explains the tested feature processing techniques. Some of them are supervised and others unsupervised.

4.1 Feature selection

Feature selection techniques have been used to reduce the dimensionality of the feature vectors while maintaining the relevant information, with the objective of improving classification performance. The tested feature selection techniques are explained in the following list:

- Variance selection: this technique consists on analyzing the variance of all the features and removing the ones that are bellow a set threshold. The reason why this method should work is that it removes the variables that hardly change, so not only they insert noise in the feature vector but they also do not give information about the class. It is an unsupervised technique because it does not take into account the class information in its process.

- Fisher score equation 19[35] (concatenated pair): first, feature vector of each pair (formed by a parent and a child) must be concatenated, creating a new feature vector that represents the pairs. This concatenated feature vector is used to calculate the Fisher score, using each pair's class. To calculate the score of each descriptor, we have used the function *spider_wrapper* of the framework *Feature Selection Library*, specifying *fisher* as feature selection method. Then, the descriptors are ranked using the score from highest to lowest. This feature selection method has a parameter to configure how much *percentage* it must crop the original feature vector. Using the

ranking of the descriptors and relating it to the parents and children (before merging) descriptors, each member of the pairs selects its best *percentage* of descriptors. These will be the descriptors selected for the final feature vector. This is a supervised technique because it takes into account the class of each pair in the calculation of the score. For family verification problem, we have two classes: related, and not related.

- Fisher score (subtracted pair): first, the feature vectors of each pair (formed by a parent and a child) must be merged, creating a new feature vector that represents the pairs. In order to perform merging, the absolute value of the subtraction of the pair is calculated. Then, the merged feature vector is used to calculate its Fisher score, using each pair's class. To calculate the score of each descriptor, we have used the function *spider_wrapper* of the framework *Feature Selection Library*, specifying *fisher* as feature selection method. Then, the descriptors are ranked using the score from highest to lowest. This feature selection method has a parameter to configure how much *percentage* it must crop the original feature vector. Using the ranking of the descriptors, the best *percentage* of descriptors are selected for the final feature vector. This is a supervised technique because it takes into account the class of each pair in the calculation of the score.

- Fisher score center and subtraction: this procedure is the same as the previous but before calculating the score, the feature is rescaled: centered by subtracting its mean and divided by its standard deviation.

$$f_i = \frac{|\mu_{i,1} - \mu_{i,2}|^2}{\sigma_{i,1}^2 + \sigma_{i,2}^2} \tag{19}$$

where $\mu_{i,1}$ and $\sigma_{i,1}^2$ refer to the mean and variance respectively of the i^{th} feature of class one (i.e. positive) and $\mu_{i,2}$ and $\sigma_{i,2}^2$ represent the same for class two (i.e. negative).

4.2 Feature projection

Feature projection techniques have been used to project the extracted features to a new space in which intra-class distance is minimized while maximizing inter-class distance,

therefore facilitating the classification task. The tested feature projection techniques are explained in the following list:

- Principal Component Analysis (PCA)[36] is an unsupervised feature extraction method, useful when the dataset has redundancy. It generates principal components from the input data, maximizing its variance and compacting the information in less variables. These components are obtained by computing the eigenvectors that have highest eigenvalues of data's correlation matrix.

- Neighborhood Repulsed Metric Learning (NRML) [8]. This neighborhood separation metric for kinship verification is motivated by the fact that, interclass samples (without a kinship relationship) with higher similarity usually lie in a neighborhood and are more easily misclassified than those with lower similarity. This metric aims to learn a distance metric under which the intraclass samples (with a kinship relation) are pulled as close as possible and interclass samples lying in a neighborhood are repulsed and pushed away as far as possible simultaneously, as shown in Figure 13. This way, more discriminative information can be exploited for verification.

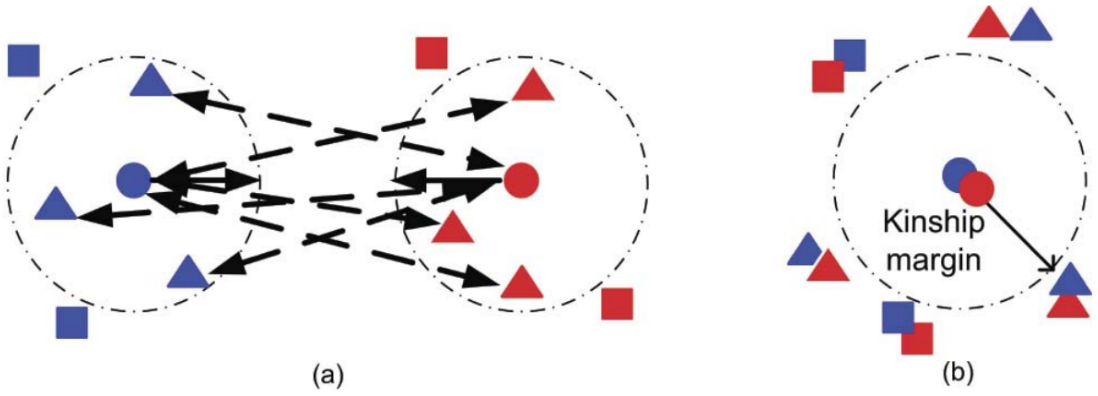

Figure 13: NRML neighborhood projection. (a) The original face images with/without kinship relations in the high-dimensional feature space. The samples in the left denote face images of parents, and those in the right denote face images of children, respectively. Given one pair of face images with kinship relation (denoted as circles), the triangles and squares denote face samples in the neighborhood and non-neighborhood, respectively. They aim to learn a distance metric such that facial images with kinship relations are projected as close as possible and those without kinship relations in the neighborhoods are pushed away as far as possible. (b) The expected distributions of face images in the learned metric space, where the similarity of the circle pair (with a kinship relation) is increased and those of the circle and triangle pairs are decreased, respectively. Figure from [8].

- Multiview Neighborhood Repulsed Metric Learning (MNRML)[8]. This multiview

metric makes better use of multiple descriptors to extract complementary information, based on NRML. It seeks a common distance metric to perform multiple feature fusion, so as to improve the kinship verification performance.

- Linear Discriminant Embedding (LDE) [37]. The objective of this projection is to estimate a linear mapping that simultaneously maximizes the local margin between heterogeneous samples and pushes the homogeneous samples closer to each other. The expected effect of LDE framework on data is shown in Figure 14.

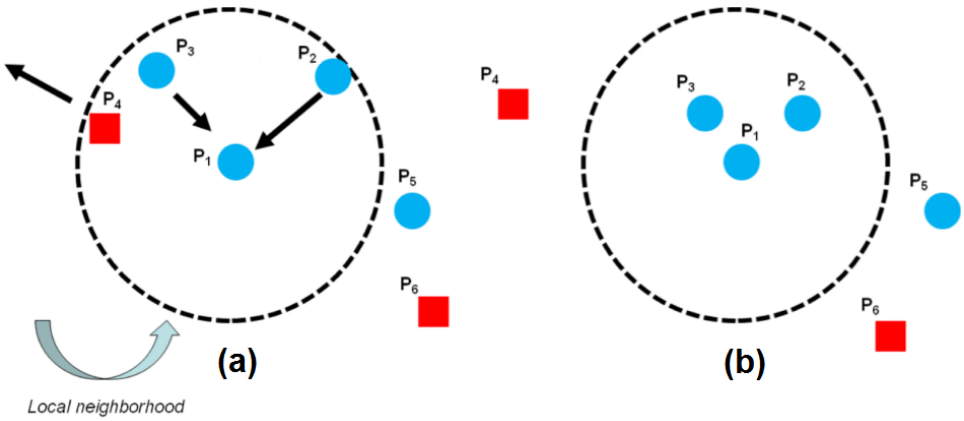

Figure 14: LDE Neighborhood. Figure from [9]: (a) Center point has three neighbors. The points with the same color and shape belong to the same class. The within-class graph connects nearby points with the same label. The between-class graph connects nearby points with different labels. (b) After LDE, the local margins between different classes are maximized, and the distances between local homogeneous samples are minimized. (a) Original space. (b) Expected mapped space using LDE

.

5 Classification

The objective of this thesis is to create a method that detects whether two images have a kinship relationship or not. In order to tackle this classification problem, two methods are proposed: cosine similarity and support vector machines.

5.1 Cosine similarity

Cosine similarity is a measure of similarity between two non-zero vectors. It is represented by the following equation

$$\cos\theta = \frac{\vec{v_p} \cdot \vec{v_c}}{\|v_p\| \, \|v_c\|} \tag{20}$$

where v_p represents the feature vector of parent for a pair and v_c represents the child of that pair.

We have used this metric to see how far the representation of the individuals (parent and child) are of each pair by each feature. Its value tends to 1 when v_p and v_c are similar and 0 when they are not similar. The best features should be the ones that have a value near 1 when the individuals of the pair belong to the positive class and near 0 if they belong to the negative class.

The cosine similarity value has been used to calculate ROC plots. Different threshold values (from 0 to 1) have been used to set the point at which pairs with values below or equal to the threshold belong to the negative class and pairs with values above belong to the positive class. Once each pair is assigned to a class, the number of Positive (P), Negative (N), True Positive (TP, number of positive pairs that have been classified as positive), True Negative (TN, number of negative pairs that have been classified as negative), False Positive (FP, number of negative pairs that have been classified as positive: FP = N-TN) and False Negative (FN, number of positive pairs that have been classified as negative: FN = P-TP) are counted. These values are used to calculate the True Positive Rate (TPR) 21 and False Positive Rate (FPR) 22, which are used to construct the Receiver Operating Characteristic (ROC) curve plot:

$$TPR = \frac{TP}{TP + FN} \tag{21}$$

$$FPR = \frac{FP}{FP + TN} \tag{22}$$

5.2 Support Vector Machine (SVM)

Support Vector Machine (SVM)[38] is a supervised machine learning method used for classification and regression. We are going to explain its use for classification as that is

what we have used it for.

SVMs are binary classifiers which consist on creating an hyperplane that separates the instances of the positive and negative classes. They are used to assign a class to unlabeled instances, regarding their position to the hyperplane. See Figure 15

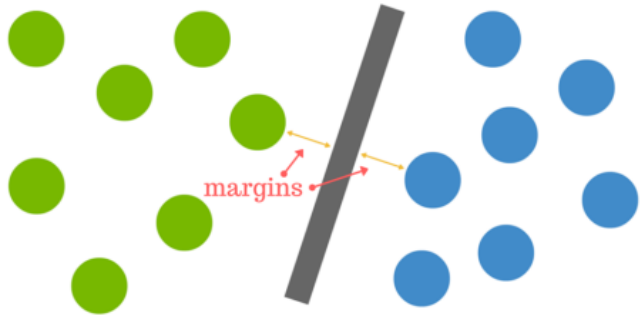

Figure 15: SVM scheme. Each dot indicates an instance and its color (green or blue) its class. The thick line that separates instances of the two classes is the SVM's hyperplane. The margin is the distance between the hyperplane and the nearest instance of either class. Figure from [10].

SVMs can have many different kernels. The most basic one is the linear one, which creates a straight hyperplane to separate the classes. The kernel that we have used has been the Gaussian kernel because it is a non lineal kernel, so it can model non linear data (this way we do not assume any behavior about our datasets).

In order to create a super-classifier that holds all the SVMs (one per feature), we blended their scores; which indicate the distance of each instance to the hyperplanes. It is in range $(-\infty, \infty)$, while its sign indicates the class to which it is assigned. The blending works as follows: in order to classify a new instance all the SVMs' scores are multiplied by a factor (that together sum one) and then all those values are summed. The sign of the sum indicates the class that the super-classifier indicates for that instance.

6 Proposed approach

This section describes the proposed pipeline after performing many experiments to find the architecture that obtains the best classification results by combining components that individually improve it. Figure 16 shows an overview of the structure of our approach.

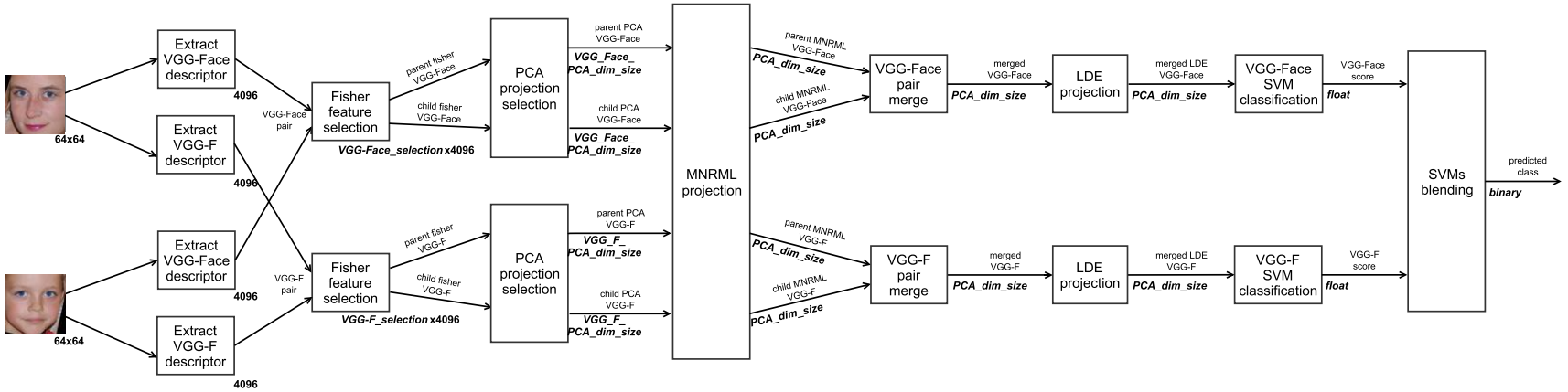

Figure 16: Proposed pipeline flowing from left to right, turning the pair of input images into a predicted class. Each block performs a change in its input and its output is connected to the input of another block. The text in bold indicates the size of the data in that particular part of the pipeline. In the scheme only VGG-Face and VGG-F are shown so that it is easier to understand but the pipeline architecture is the same for any number of descriptor types.

Its main parts are feature extraction (concretely using VGG-Face and VGG-F descriptors), fisher feature selection, PCA projection and selection, MNRML projection, pairs' data merging, individual SVM classifiers and SVM classifiers blending.

The following lines describe the behavior of the pipeline while giving details of its components. They go through the pipeline from the input images until obtaining their predicted class.

1. Image pair: each classification instance is formed by a pair of parent and child; this being the input of the pipeline. Each individual data is a 64x64 pixel image that contains the person's face.

2. Feature extraction: this part consists in extracting the most relevant information of the input images, representing it in features. In Figure 16, only deep features appear so that it is easier to follow, although many more features can be added following the established order.

3. Fisher feature selection: in order to remove irrelevant and noisy descriptors that do not contain information about the class, only relevant descriptors are chosen and the rest are removed. Different feature selection techniques have been tested and Fisher feature selection has been the one that provided the best results. It needs a training set of positive and negative pairs in order to rank the descriptors. First, the feature vectors of each pair must be merged (performing the absolute value of the subtraction). Once having the ranking, only the best (*feature_selection*) part of each feature is maintained.

4. PCA projection selection: it compacts its input data in less descriptors, thus reducing its dimensionality to *PCA_dim_size*.

5. MNRML projection: projects all the input features to a space in which the pairs that are not similar but related are pulled together and the ones that are similar but not related are pushed away. This projection works on the feature vectors of each pair.

6. Pair merging per feature: each pair's feature vector is calculated by performing the absolute value of the subtraction of the same feature for its individuals.

7. LDE projection: this projection also tries to pull together the instances that belong to the same class and push away the ones that belong to different class. This projection works on the pairs' merged feature vectors.

8. SVM classification per descriptor: each descriptor has its own SVM classifier that produces a score, whose sign indicates the assigned class.

9. SVMs blending: it combines all descriptor SVM classifiers into one. It merges the prediction of all the SVMs (there is one SVM per descriptor) by weighting them regarding their performance. To do so, the score of each individual SVM is multiplied by its coefficient computed by the MNRML technique (which indicates the relevance of that SVM comparing to the rest. Their sum is 1) and summed with the result of this calculus of all SVMs. The result is a number between $-\infty$ and ∞, which is converted into predicted class (0 if the number is negative or 0, or class 1 otherwise) per each instance.

   It worthy to mention that at the training phase, computing PCA and MNRML requires a set of positive pairs only. However, Fisher selection, LDE, and SVM requires positive and negative pairs.

7 Experimental setup

Our study concerns two hand-crafted descriptors and two deep descriptors. The deep descriptors were obtained by pre-trained CNNs. ImageNet VGG-F was trained on images of objects for the purpose of image categorization, while ImageNet VGG-Face for face identification (VGG-Face). With regards to descriptor sizes, the LBP has 243, HOG has 468, and the deep features $4,096$.

In order to extract VGG-Face and VGG-F deep descriptors, we have used MatConvNet, which is a Matlab Toolbox for implementing Convolutional Neural Networks for computer vision applications; we downloaded the pre-trained networks. For feature selection, we used Matlab's Feature Selection Library [39] [40] or own implementations.

7.1 Datasets

In our study, we have used the public database called Kinship Face in the Wild (KinFaceW), which contains unconstrained face images collected for studying the problem of kinship verification. It is composed of two datasets: KinFaceW-I and KinFaceW-II. They contain images of parents and children, together with metadata that specifies the pairs association and the class (whether there is a kinship relation between a pair of images or not).

The data is split in 5 folds so as to provide a testing framework that allows direct comparison between different approaches. There are four representative types of kin relations:

Father-Son (F-S), Father-Daughter (F-D), Mother-Son (M-S) and Mother-Daughter (M-D), respectively.

As explained in the official website [11], face images are collected from the Internet, including some public face images as well as their parents' or children's face images. Face images are captured under uncontrolled environments in the two datasets with no restriction in terms of pose, lighting, background, expression, age, ethnicity, and partial occlusion. All the images have been cropped to the face size, aligned and resized to 64 $\times$ 64 pixels.

The difference of KinFaceW-I and KinFaceW-II is that face images with a kin relation were acquired from different photos in KinFaceW-I and the same photo in KinFaceW-II in most cases. Moreover, the number of data available for each database is also different. In the KinFaceW-I dataset, there are 156, 134, 116, and 127 pairs of kinship images for the four relations: (F-S), (F-D), (M-S) and (M-D) respectively. For the KinFaceW-II dataset, each relation contains 250 pairs of kinship images. Figure 17 shows several relation images of both databases.

Figure 17: Aligned and cropped examples of KinFaceW-I (left) and KinFaceW-II (right) datasets. From top to bottom are the F-S, F-D, M-S and M-D kinship relations, and the neighboring two images in each row are with the kinship relation, respectively. Figure from [11].

These datasets are available online [11].

7.2 Search of best configuration

This section is about the most relevant experiments performed to find the best results achieved by this master thesis. As baseline, the accuracy of each descriptor is measured using only a SVM per descriptor (see table 1). In principle, the pipeline should always improve the best results shown in that table.

Table 1: Individual descriptor performance. Baseline results for each image descriptor classified by Gaussian SVM on KinFace-I and KinFace-II databases. Accuracies are shown as a percentage. Best results are shown in bold.

Database	Descriptor	Accuracy F-D	Accuracy F-S	Accuracy M-D	Accuracy M-S	Accuracy mean
KinFace-I	VGG-Face	57.09	61.22	62.99	59.05	62.44
	VGG-F	61.19	69.55	**75.98**	**67.67**	**66.85**
	LBP	60.82	58.33	58.27	59.05	59.12
	HOG	**62.31**	**70.51**	63.39	60.78	64.24
KinFace-II	VGG-Face	58.60	56.60	61.40	57.60	58.55
	VGG-F	**74.60**	**76.20**	**80.60**	**78.00**	**77.35**
	LBP	64.00	61.60	61.40	61.00	62.00
	HOG	66.20	70.00	64.40	69.80	67.60

Table 1 shows that VGG-F is the best image descriptor individually and HOG performed better than VGG-Face but worse than VGG-F.

In order to set the best configuration of LBP descriptor for this problem, we have based out configuration on Huerta *et al.* [41]; where authors found the best settings per each image size for facial age estimation. In our case, images are of size 64×64 pixels, so we followed the paper parameters that represent the nearest dimensions: 50×50 pixels. The resulting parameters have been number of sampling points 16; radius 4 and uniform mode for the two datasets.

In order to set the best configuration of HOG descriptor for this problem, we have again followed the strategy of Huerta *et al.* [41]; where authors found the best settings per each image size for facial age estimation. The images of this problem are of size 64×64, so we have chosen the parameters that represent the nearest dimensions: 50×50. The resulting parameters have been cell size of 15×15; number of bins 13 for the two datasets. The methodology followed in the search of the best configuration for the pipeline has been the following: every component is set as it is and only one of its components is added/modified at a time. Then, the results for the new setting are computed and compared with the best previous results. Experimentation continues from the setting that obtained the best results. The majority of the experiments were performed in the database KinFace-I because most of the proposals perform worse on this than in KinFace-II. Moreover, using the same dataset allows comparison among the results of different configurations. Table 2 shows different settings of the pipeline and the best configuration found in bold.

Table 2: Different architectures tested to find the best configuration. All the cells that contain an X in a row are assembled together to complete the pipeline evaluated in that row. VGG-Face and VGG-F deep descriptors are fed as input data for all the experiments of the table. Accuracies are shown as a percentage. Best results are displayed in bold.

Fisher feature selection	PCA	MNRML	LDE	SVMs blending	Database	Accuracy F-D	Accuracy F-S	Accuracy M-D	Accuracy M-S	Accuracy mean
X	X	X	X	X	KinFace-I	78.36	79.49	78.74	84.05	80.16
					KinFace-II	80.40	83.20	85.20	84.40	83.30
	X	X	X	X	KinFace-I	65.67	75.64	77.56	72.84	72.93
					KinFace-II	77.20	80.20	82.40	81.20	80.25
X		X	X	X	KinFace-I	62.69	72.76	72.44	64.22	68.03
					KinFace-II	51.40	64.60	63.00	50.20	57.30
X	X		X	X	KinFace-I	71.64	78.53	79.13	79.13	77.15
					KinFace-II	77.60	82.20	85.80	81.20	81.70
X	**X**	**X**		**X**	**KinFace-I**	**79.85**	**83.97**	**86.61**	**87.07**	**84.38**
					KinFace-II	**82.20**	**86.80**	**89.00**	**87.60**	**86.40**
	X	X		X	KinFace-I	65.30	77.88	82.28	78.08	75.87
					KinFace-II	77.60	81.00	83.20	81.60	80.85
X		X		X	KinFace-I	66.42	74.68	80.71	79.31	75.28
					KinFace-II	77.00	82.00	84.20	82.00	81.30
X	X			X	KinFace-I	73.13	81.41	80.71	82.33	79.40
					KinFace-II	80.40	84.40	87.40	82.80	83.75

Table 3: Deep vs deep+traditional descriptors with the best known architecture to see which performs best in databases KinFace-I and KinFace-II. Accuracies are shown as a percentage. Best results are shown in bold.

Database	Descriptors	Accuracy F-D	Accuracy F-S	Accuracy M-D	Accuracy M-S	Accuracy mean
KinFace-I	VGG-Face + VGG-F	**79.48**	**84.62**	**86.61**	**87.07**	**84.44**
	VGG-Face + VGG-F + LBP + HOG	72.39	79.17	80.71	81.47	78.43
KinFace-II	VGG-Face + VGG-F	**82.40**	**86.80**	**89.00**	**87.40**	**86.40**
	VGG-Face + VGG-F + LBP + HOG	77.20	84.40	85.80	84.00	82.85

In table 2 only deep descriptors were used because previous experiments had shown that deep descriptors alone obtained better results than deep+traditional descriptors combined (see section 11 for details). In order to confirm that statement, the best configuration found was tried with both descriptor combinations (see table3).

Table 3 shows that the fusion of VGG-Face and VGG-F descriptors obtains better results than any descriptor individually. It also shows that only deep descriptors obtain better results than using deep and traditional descriptors.

8 Results

The best configuration obtained in the previous section is compared against the state-of-the-art results for KinFace-I and KinFace-II databases (see table 4).

Table 4: Results of the proposed architecture compared against state-of-the-art approaches on databases KinFace-I and KinFace-II. The symbol * represents that the approach takes outside data for training. Accuracy is shown as a percentage. Best results without using external data are shown in bold.

Year	Authors	Algorithm	Database	Accuracy
2014	Lu et al. [8]	Multiview neighborhood repulsed metric learning	KinFace-I	69.90
			KinFace-II	76.50
	Yan et al. [15]	Discriminative multimetric learning	KinFace-I	72.00
			KinFace-II	78.00

Table 4: Results of the proposed architecture compared against state-of-the-art approaches on databases KinFace-I and KinFace-II. The symbol * represents that the approach takes outside data for training. Accuracy is shown as a percentage. Best results without using external data are shown in bold.

Year	Authors	Algorithm	Database	Accuracy
	Dehghan et al. [28]	Discrimination via gated autoencoders	KinFace-I	74.50
			KinFace-II	82.20
2015	Yan et al. [25]	Prototype discriminative feature learning	KinFace-I	70.10
			KinFace-II	77.00
	Liu et al. [42]	Inheritable Fisher Vector Feature based kinship	KinFace-I	73.45
			KinFace-II	81.60
	Alirezazadeh et al. [43]	Genetic Algorithm for feature selection for kinship	KinFace-I	81.30
			KinFace-II	86.15
2016	Zhou et al. [17]	Ensenble similarity learning	KinFace-I	78.60
			KinFace-II	75.70
	Qin et al. [44]	Kinship verification via multi-view multi-task learning	KinFace-I	73.70
			KinFace-II	77.20
	Liu et al. [45]	Inheritable Color Space with Application to Kinship Verification	KinFace-I	77.85
			KinFace-II	81.40
	Puthenputhussery et al. [46]	SIFT flow based genetic fisher vector feature kinship verification	KinFace-I	76.09
			KinFace-II	85.65
	Li et al. [47]	Similarity Metric Based CNN for kinship verification	KinFace-I	72.72
			KinFace-II	79.25
	Fang et al. [48]	Sparse Similarity Metric Learning for Kinship Verification	KinFace-I	79.55
			KinFace-II	80.15
	Zhou et al. [49]	Kinship verification from by scalable similarity fusion	KinFace-I	78.00
			KinFace-II	75.90
	Lan et al. [50]	Quaternion-Michelson descriptor for color image classification	KinFace-I	71.00
			KinFace-II	75.30
2017	Lan et al. [51]	Quaternionic Weber local descriptor of color images	KinFace-I	73.60
			KinFace-II	76.60
	Yan et al. [52]	Neighborhood repulsed correlation metric learning for Kinship Ver.	KinFace-I	65.50
			KinFace-II	77.90
	Kohli et al. [23]	Hierarchical Representation Learning for Kinship Verification	KinFace-I	96.10*
			KinFace-II	96.20*
	Kumar et al. [53]	Harmonic Rule for Measuring the Facial Similarities among Relatives	KinFace-I	80.60

Table 4: Results of the proposed architecture compared against state-of-the-art approaches on databases KinFace-I and KinFace-II. The symbol * represents that the approach takes outside data for training. Accuracy is shown as a percentage. Best results without using external data are shown in bold.

Year	Authors	Algorithm	Database	Accuracy
			KinFace-II	84.40
	Serradilla et al.	**Proposed approach**	**KinFace-I**	**84.38**
			KinFace-II	**86.40**

The pipeline configuration that obtains the best results is now analyzed. First, the confusion matrix has been processed for all the pairs of the two databases (see table 5).

Finally, the elapsed time for the best configuration is measured. The experiments have been run in a 16GB RAM intel core i3-6100 2cores-4threads 3.7GHz CPU computer. The train and test times are recorded in table 6.

Figures 18 and 19 show pictures of testing examples in which the proposed approach predicts the class correctly and examples in which it does not.

Table 5: Confusion matrices for the best pipeline configuration for all kinship relationships in KinFace-I and KinFace-II databases.

Database	**Pair**	**TP**	**FP**
		FN	**TN**
KinFace-I	F-D	108	26
		28	106
	F-S	129	27
		23	133
	M-D	109	18
		16	111
	M-S	102	14
		16	100

Database	**Pair**	**TP**	**FP**
		FN	**TN**
KinFace-II	F-D	204	46
		43	207
	F-S	224	26
		40	210
	M-D	223	27
		28	222
	M-S	220	30
		32	218

Table 6: Test and train elapsed time for the best configuration. The train time is calculated for 214 parent-child pair images while the test time is calculated only for one pair image on average. Numbers' measure is milliseconds (ms).

Task	Train	Test
Deep features extraction	73209.4	342.1
Fisher feature selection	4	< 0.1
PCA projection	0.3	< 0.1
MNRML projection	20	< 0.1
Pair merging	< 0.1	< 0.1
Classification (SVMs + blending)	< 0.1	< 0.1
Subtotal no deep descriptors extraction	24.3	≈ 0.2
Total	73233.7	342.3

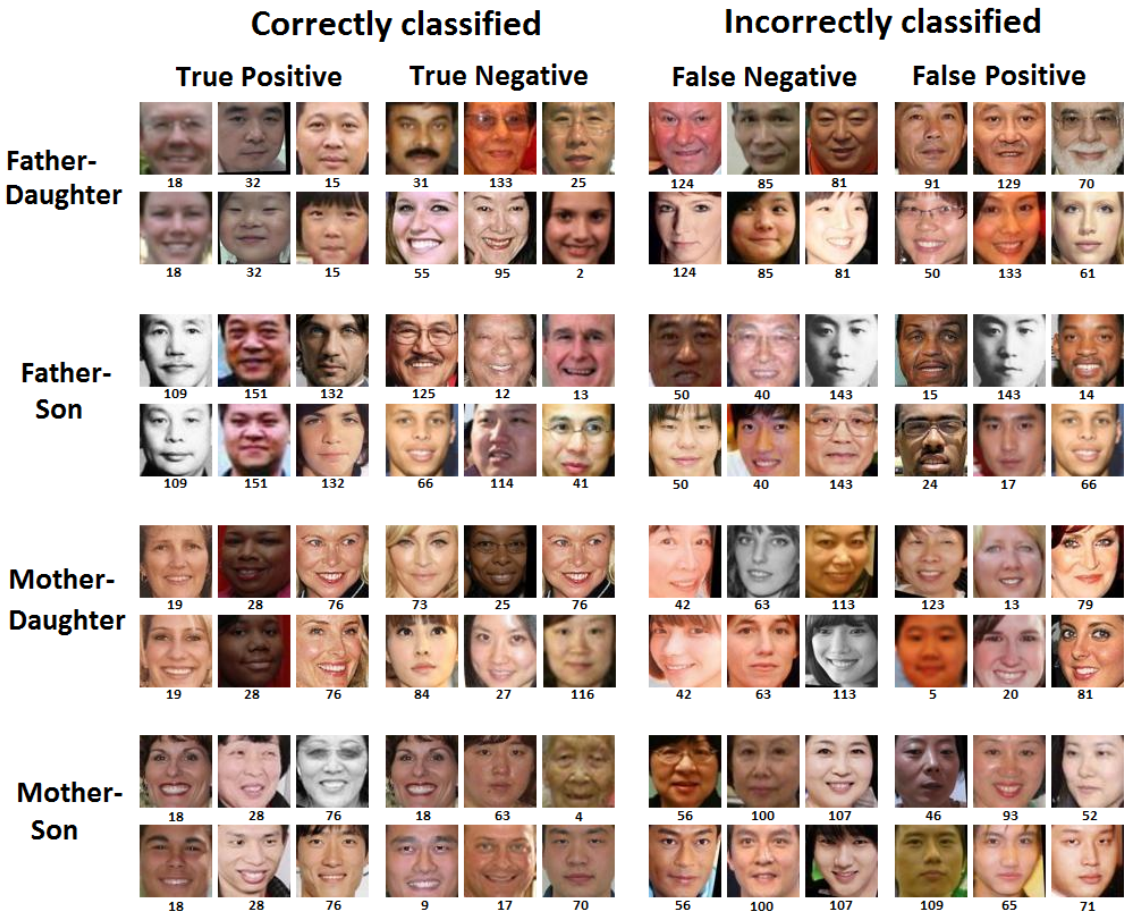

Figure 18: KinFace-I pipeline's classification results. Here we show correctly and incorrectly classified pairs, ordered by kinship relation. The number below each image indicates the ID of the pair in the database.

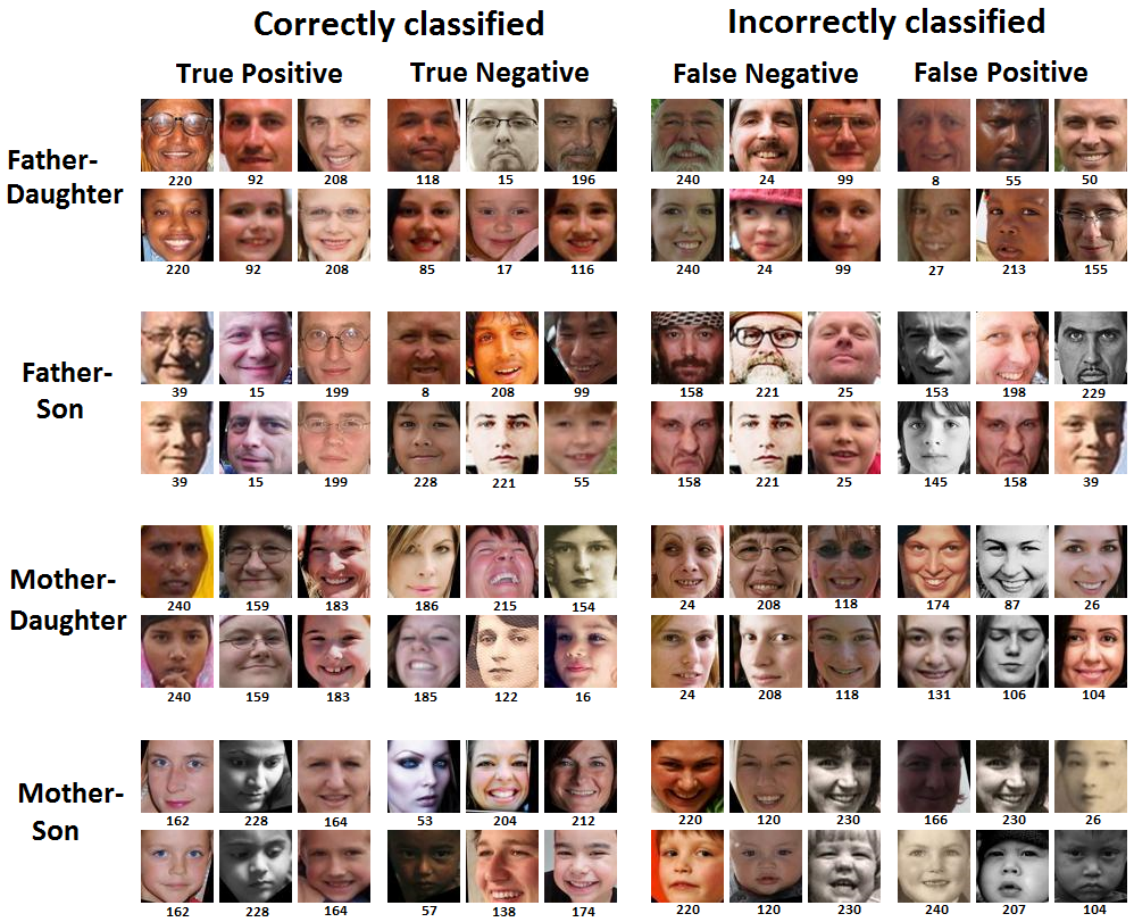

Figure 19: KinFace-II results pictures. We show here correctly and incorrectly classified pairs, ordered by kinship relation. The number below each image indicates the ID of the pair in the database.

9 Conclusions

This report proposes a pipeline for kinship verification, formed by components that individually improve the accuracy and together reach the best results of the state on the art on KinFace-I and KinFace-II databases that do not use additional data.

For SVM classification, the best pair merging method found for these databases has been the absolute value of the subtraction of the individuals. The same value divided by the sum of the individuals metric does not work well because the values of the vector are very small for the classifiers to perform well. Concatenation of the individuals does not work well either.

Fisher feature selection has been the only one found that improves the classification re-

sults. Variance feature selection did not improve them.

PCA dimension has much influence in the final accuracy, that is why almost all the experiments try different PCA dimensions to choose the best one for each setting.

MNRML projection has shown to be a good way for improving the performance of the descriptors that work well individually, improving the results of any individual descriptor.

LDE does not improve the accuracy in this problem probably because its input data is very discriminated by the previous two projections, so this new projection is not able to separate the data even more.

SVM with Gaussian kernel has been the best classifier found and is also very common in the state-of-the-art approaches. This is because it is a classifier that is easy to compute, performs well with big number of descriptors and gets good classification results in general.

The SVMs blending technique using its score (weighing the distance from the hyperplane) has been successful.

Adding traditional descriptors (LBP and HOG) worsen the results obtained by the deep descriptors. Possibly, the MNRML considers them with large contribution so they deteriorate the significance of deep descriptors.

10 Future Lines

Despite having performed more than 50 experiments and achieved state-of-the-art results, we think that there is still room for improvement in the pipeline's accuracy.

The most direct way to improve the pipeline is tweaking the parameters of the current structure. Until now, only exhaustive search has been done for several ranges and values. Using a genetic algorithm for parameters setting could be a good choice to find the best setting in feasible time.

Feature selection can adopt a wrapper technique that use the entire pipeline in order to select the best input features.

We could also try feeding the pipeline with the descriptors used in [23] to see if the results can be improved.

Changing the VGG-F descriptor by VGG-M and VGG-S descriptors could improve the

results of the current pipeline, since they seem to perform better [7]. Moreover, other pre-trained CNNs could also be tested.

Another option would be creating a new ANN similar to VGG-Face and VGG-F but with two input images instead of one, creating a NN designed and trained specifically for kinship verification problem. Its final layers should be fully connected to perform classification and the last layer only containing two neurons as output, so that the network directly outputs the probability of input images having kinship relationship.

Changing the way SVM's blending method. Maybe using a majority voting algorithm could improve the performance of the pipeline (i.e Adaboost).

11 Appendix

The code is available on GitHub in the following link so that anyone can replicate the results of this work: https://github.com/oserradilla/KinVer

As appendix there is another document available that shows the development of the project step-by-step, recording all the secondary results. These have been relevant to discover which components and descriptors are important, guide the thesis and reach the primary results.

References

[1] Nielsen, M.A.: Neural Networks and Deep Learning. Determination Press (2015)

[2] Andrew Ng, Jiquan Ngiam, C.Y.F.Y.M.C.S.: UFLDF Tutorial. Computer Science Department, Stanford University (2011)

[3] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86** (1998) 2278–2324

[4] Ahonen, T., Hadid, A., Pietikainen, M.: Face description with local binary patterns: Application to face recognition. IEEE transactions on pattern analysis and machine intelligence **28** (2006) 2037–2041

[5] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Volume 1., IEEE (2005) 886–893

[6] Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. In: British Machine Vision Conference. (2015)

[7] Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: Delving deep into convolutional nets. In: British Machine Vision Conference. (2014)

[8] Lu, J., Zhou, X., Tan, Y.P., Shang, Y., Zhou, J.: Neighborhood repulsed metric learning for kinship verification. IEEE transactions on pattern analysis and machine intelligence **36** (2014) 331–345

[9] Dornaika, F., Bosaghzadeh, A.: Exponential local discriminant embedding and its application to face recognition. IEEE transactions on cybernetics **43** (2013) 921–934

[10] Noel Bambrick, A.: (Support vector machines: A simple explanation) [Online; accessed 30-June-2017].

[11] Jiwen Lu, Junlin Hu, X.Z.Y.S.Y.P.T.G.W.: Kinfacew. (2014)

[12] Guo, Y., Dibeklioglu, H., van der Maaten, L.: Graph-based kinship recognition. In: Pattern Recognition (ICPR), 2014 22nd International Conference on, IEEE (2014) 4287–4292

[13] Hu, J., Lu, J., Yuan, J., Tan, Y.P.: Large margin multi-metric learning for face and kinship verification in the wild. In: ACCV (3). (2014) 252–267

[14] Xia, S., Shao, M., Luo, J., Fu, Y.: Understanding kin relationships in a photo. IEEE Transactions on Multimedia **14** (2012) 1046–1056

[15] Yan, H., Lu, J., Deng, W., Zhou, X.: Discriminative multimetric learning for kinship verification. IEEE Transactions on Information forensics and security **9** (2014) 1169–1178

[16] Zhang12, K., Huang, Y., Song, C., Wu, H., Wang, L., Intelligence, S.M.: Kinship verification with deep convolutional neural networks. (2015)

[17] Zhou, X., Shang, Y., Yan, H., Guo, G.: Ensemble similarity learning for kinship verification from facial images in the wild. Information Fusion **32** (2016) 40–48

[18] Hu, J., Lu, J., Tan, Y.P.: Discriminative deep metric learning for face verification in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014) 1875–1882

[19] Fang, R., Tang, K.D., Snavely, N., Chen, T.: Towards computational models of kinship verification. In: Image Processing (ICIP), 2010 17th IEEE International Conference on, IEEE (2010) 1577–1580

[20] Zhou, X., Hu, J., Lu, J., Shang, Y., Guan, Y.: Kinship verification from facial images under uncontrolled conditions. In: Proceedings of the 19th ACM international conference on Multimedia, ACM (2011) 953–956

[21] Guo, G., Wang, X.: Kinship measurement on salient facial features. IEEE Transactions on Instrumentation and Measurement **61** (2012) 2322–2325

[22] Zhou, X., Lu, J., Hu, J., Shang, Y.: Gabor-based gradient orientation pyramid for kinship verification under uncontrolled environments. In: Proceedings of the 20th ACM international conference on Multimedia, ACM (2012) 725–728

[23] Kohli, N., Vatsa, M., Singh, R., Noore, A., Majumdar, A.: Hierarchical representation learning for kinship verification. IEEE Transactions on Image Processing **26** (2017) 289–302

[24] Kohli, N., Singh, R., Vatsa, M.: Self-similarity representation of weber faces for kinship classification. In: Biometrics: Theory, Applications and Systems (BTAS), 2012 IEEE Fifth International Conference on, IEEE (2012) 245–250

[25] Yan, H., Lu, J., Zhou, X.: Prototype-based discriminative feature learning for kinship verification. IEEE Transactions on cybernetics **45** (2015) 2535–2545

[26] Lu, J., Liong, V.E., Zhou, X., Zhou, J.: Learning compact binary face descriptor for face recognition. IEEE transactions on pattern analysis and machine intelligence **37** (2015) 2041–2056

[27] Lu, J., Tan, Y.P., Wang, G.: Discriminative multimanifold analysis for face recognition from a single training sample per person. IEEE transactions on pattern analysis and machine intelligence **35** (2013) 39–51

[28] Dehghan, A., Ortiz, E.G., Villegas, R., Shah, M.: Who do i look like? determining parent-offspring resemblance via gated autoencoders. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014) 1757–1764

[29] Dibeklioglu, H., Ali Salah, A., Gevers, T.: Like father, like son: Facial expression dynamics for kinship verification. In: Proceedings of the IEEE International Conference on Computer Vision. (2013) 1497–1504

[30] Xia, S., Shao, M., Fu, Y.: Kinship verification through transfer learning. In: IJCAI. (2011) 2539–2544

[31] Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural networks **4** (1991) 251–257

[32] Rumelhart, D.E., Hinton, G.E., Williams, R.J., et al.: Learning representations by back-propagating errors. Cognitive modeling **5** (1988) 1

[33] Sathyanarayana, S.: A gentle introduction to backpropagation (2014)

[34] Ojala, T., Pietikainen, M., Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on pattern analysis and machine intelligence **24** (2002) 971–987

[35] Gu, Q., Li, Z., Han, J.: Generalized fisher score for feature selection. arXiv preprint arXiv:1202.3725 (2012)

[36] Jolliffe, I.: Principal component analysis. Wiley Online Library (2002)

[37] Chen, H.T., Chang, H.W., Liu, T.L.: Local discriminant embedding and its variants. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Volume 2., IEEE (2005) 846–853

[38] Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20** (1995) 273–297

[39] Roffo, G., Melzi, S.: Ranking to learn: Feature ranking and selection via eigenvector centrality. New Frontiers in Mining Complex Patterns, Fifth International workshop, nfMCP2016 (2017)

[40] Roffo, G., Melzi, S., Cristani, M.: Infinite feature selection. In: 2015 IEEE International Conference on Computer Vision (ICCV). (2015) 4202–4210

[41] Huerta, I., Fernández, C., Segura, C., Hernando, J., Prati, A.: A deep analysis on age estimation. Pattern Recognition Letters **68** (2015) 239–249

[42] Liu, Q., Puthenputhussery, A., Liu, C.: Inheritable fisher vector feature for kinship verification. In: Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on, IEEE (2015) 1–6

[43] Alirezazadeh, P., Fathi, A., Abdali-Mohammadi, F.: A genetic algorithm-based feature selection for kinship verification. IEEE Signal Processing Letters **22** (2015) 2459–2463

[44] Qin, X., Tan, X., Chen, S.: Mixed bi-subject kinship verification via multi-view multi-task learning. Neurocomputing **214** (2016) 350–357

[45] Liu, Q., Puthenputhussery, A., Liu, C.: A novel inheritable color space with application to kinship verification. In: Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on, IEEE (2016) 1–9

[46] Puthenputhussery, A., Liu, Q., Liu, C.: Sift flow based genetic fisher vector feature for kinship verification. In: Image Processing (ICIP), 2016 IEEE International Conference on, IEEE (2016) 2921–2925

[47] Li, L., Feng, X., Wu, X., Xia, Z., Hadid, A.: Kinship verification from faces via similarity metric based convolutional neural network. In: International Conference Image Analysis and Recognition, Springer (2016) 539–548

[48] Fang, Y., Chen, Y.Y.S., Wang, H., Shu, C.: Sparse similarity metric learning for kinship verification. In: Visual Communications and Image Processing (VCIP), 2016, IEEE (2016) 1–4

[49] Zhou, X., Yan, H., Shang, Y.: Kinship verification from facial images by scalable similarity fusion. Neurocomputing **197** (2016) 136–142

[50] Lan, R., Zhou, Y.: Quaternion-michelson descriptor for color image classification. IEEE Transactions on Image Processing **25** (2016) 5281–5292

[51] Lan, R., Zhou, Y., Tang, Y.Y.: Quaternionic weber local descriptor of color images. IEEE Transactions on Circuits and Systems for Video Technology **27** (2017) 261–274

[52] Yan, H.: Kinship verification using neighborhood repulsed correlation metric learning. Image and Vision Computing **60** (2017) 91–97

[53] Kumar, C.R.: Harmonic rule for measuring the facial similarities among relatives. Transactions on Machine Learning and Artificial Intelligence **4** (2017) 29