PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

DR. YAHIA FARES UNIVERSITY OF MEDEA



Faculty of **T**echnology

Department of **E**lectrical **E**ngineering

Master's End of Study Project

Branch: Telecommunications.

Option: Telecommunication Systems.

Deep Convolutional Features to Improve Facial Kinship Verification

---

Presented By:

LAOUADI Oussama.
ZOULIKHA Walid.

Proposed and Supervised By:

Dr. CHOUCHANE Ammar.

Class 2019/2020

Dedication

To all my Family.

To all my friends with their help and support.

To everyone who wants to make a positive change in this world.

Oussama.

I dedicate this work with my warm thanks and my deep gratitude:

To all my dear family.

To all my friends with their help and support, especially to Abd Elkader Filali.

To all the victims of the covid-19 pandemic.

Walid.

Acknowledgments

First, we thank Allah for giving us this opportunity to accomplish this work.

We'd like to thank our supervisor ***Dr. Ammar Chouchane*** *for his efforts in this work, is expertise, experience and patience were invaluable throughout the duration of this work especially within the unfortunate circumstances of the current pandemic.*

Finally, many thanks go to the teachers and staff of Electrical Engineering department of Medea University.

Abstract

Kinship verification from face images is a new and challenging field of study in both biometrics and computer vision, involving many researches and studies due to its importance in many potential applications, such as forensics and finding missing children. It consists of automatically determining whether two persons share a biological kin relation by examining their facial attributes, i.e., features. These features are extracted using many methods, such as descriptors. In this work we develop an efficient method named Hist-Gabor based on the histogram features extracted from basic Gabor wavelet. Indeed, we examine the use of deep features collected from a convolutional neural network model called VGG-face and shallow features by our new Gabor wavelet invoking a powerful dimensionality reduction method called TXQDA. Empirically, the extensive experiments demonstrate that the proposed approach outperforms the previous state-of-the-art in the challenging datasets Cornell and TSKinFace.

Keywords**:** *facial kinship verification, deep learning, machine learning, Hist-Gabor descriptor, computer vision, TXQDA.*

Résumé

La vérification de la parenté à partir des images faciales est un domaine d'étude nouveau et difficile en biométrie et en vision par ordinateur impliquant de nombreuses recherches et études en raison de son importance dans de nombreuses applications telles que la criminalistique et la recherche d'enfants disparus. Il consiste à déterminer automatiquement si deux personnes partagent une relation biologique de parenté en examinant leurs attributs faciaux, appelés traits ou caractéristiques. Ces caractéristiques sont extraites à l'aide de nombreuses méthodes, telles que les descripteurs. Dans ce travail, nous développons une méthode efficace appelée Hist-Gabor basée sur les caractéristiques de l'histogramme extraites depuis l'ondelette de base de Gabor. En effet, nous examinons l'utilisation des caractéristiques profondes collectées à partir d'un modèle de réseau de neurones convolutif appelé VGG-face et des caractéristiques superficiels par notre nouvelle ondelette de Gabor en utilisant une méthode puissante de réduction des dimensions appelée TXQDA. Empiriquement, les nombreuses expériences démontrent que notre approche proposée surpasse les approches de l'état de l'art antérieur dans les bases de données difficiles Cornell et TSKinFace.

Mots-Clés **:** *vérification faciale de la parenté, apprentissage profond, apprentissage machine, Hist-Gabor, vision d'ordinateur, TXQDA.*

ملخّص

التحقّق من القرابة الوجهية هو ميدان دراسة جديدة في علم الإحصاء الحيوي والرؤية الحاسوبية، يضمّ العديد من الأبحاث والدّراسات لأهمّيته في عدة تطبيقات مُحتملة مثل الأدلّة الجنائية والعثور على الأطفال المفقودين. يقوم على التّحديد التلقائي للقرابة البيولوجية للأشخاص اعتمادًا على سمات الوجه. هاته السّمات يتمّ جمُعها باستعمال عدة طرق، تكون غالبًا عبارة عن محدّدات تدعى واصفات الوجه. نحاول في هذا العمل تطوير منهجيّة فعالة تدعى Hist-Gabor اعتمادًا خصائص مخطط التوزيع الإحصائي المُستخلصة من مُوجيات غابور الأساسية. نقوم بدراسة استعمال الخصائص العميقة المُستخلصة عن طريق الشّبكات العصبونية الالتفافية VGG-face والخصائص السّطحية المُستخلصة باستعمال منهجيّتنا المقترحة التي تعتمد على مويجات غابور، باستعمال تقنية فعّالة لتقليص حجم الأبعاد تدعى TXQDA. اعتمادًا على النّتائج التي قدّمتها عديد التجارب التي قمنا بها، منهجيّتُنا المقترحة تتفوّق على نتائج الطّرق الحديثة في هذا المجال في قواعد البيانات الخاصة بـ Cornell وTSK-KinFace.

كلمات مفتاحية : التحقق من القرابة الوجهية، التعلم العميق، التعلم الآلي، Hist-Gabor ، الرؤية الحاسوبية، TXQDA.

Table of Contents

Dedication I

Acknowledgments II

Abstract III

Résumé III

ملخّص IV

List of Figures VIII

List of Tables X

List of Abbreviations XI

General Introduction 1

Deep Learning & Artificial Intelligence: An Overview 3

1.1 Introduction 3

1.2 Artificial Intelligence 3

1.3 Machine Learning 4

1.3.1 Types of Machine Learning 6

1.3.1.1 Supervised Learning 6

1.3.1.2 Unsupervised Learning 7

1.3.1.3 Semi-Supervised Learning 8

1.3.1.4 Reinforcement Learning 8

1.3.2 Data for machine learning 9

1.3.3 Artificial Neural Networks 10

1.3.3.1 Multilayer Perceptron (MLP) 12

1.3.3.2 Activation Functions: 13

1.3.3.3 Learning in a neural network 14

1.4 Deep Learning 18

1.4.1 Deep Neural Networks 18

1.4.1.1 Recurrent Neural Networks (RNN) 19

1.4.1.2 Autoencoders 19

1.4.1.3 Convolutional Neural Networks (CNNs) 20
1.4.1.4 CNN Case Study: VGG16 24
1.4.1.5 Deep Learning Frameworks 25
1.4.2 Applications of Deep Learning 26
1.4.2.1 Computer Vision 26
1.4.2.2 Speech and Language Processing 29
1.4.2.3 Biometrics 29
1.5 Conclusion 30
Biometrics and Facial Kinship Verification 31
2.1 Introduction 31
2.2 General description of biometric systems 31
2.3 Modalities of biometric systems 33
2.4 Applications of biometric systems 35
2.5 Kinship Verification 35
2.6 Challenges of face kinship verification 36
2.6.1 Unique Challenges 36
2.6.2 Common challenges 37
2.6.3 Kinship modes 39
2.6.4 Kinship verification system structure. 40
2.7 Data bases for kinship verification 41
2.8 Conclusion 43
Kinship Verification Based VGG16 & Gabor Wavelet Features 44
3.1 Introduction 44
3.2 Environment used and work outline 44
3.3 Preprocessing stage 45
3.4 Features Extraction 46
3.4.1 Features extraction with VGG16 and VGGface model 46
3.4.2. Feature Extraction using Gabor Wavelet 47
3.4.2.1. Theoretical Aspects of Gabor filters 47
3.4.3. Tensor Design 50
3.4.4. Dimensionality reduction stage 50

3.4.5. Train and test Phase 52
3.4.6. Scoring and Fusion phase 53
3.5 Implementation and Results 54
3.5.2. Cornell KinFace Dataset 54
3.5.3. TSK KinFace Dataset 56
3.6 Discussion 57
3.6.1. Comparison with state-of-the-art methods 58
3.7 Conclusion 59
General Conclusion 61
Appendices 63
References 66

List of Figures

Fig.1.1 Supervised Learning.....6

Fig.1.2 Unsupervised Learning.....8

Fig.1.3 Semi-supervised learning.....8

Fig.1.4 Biological structure of a neuron.....10

Fig.1.5 Mathematical model of an artificial Neuron.....11

Fig.1.6 Simple feed forward network.....12

Fig.1.7 Commonly used activation functions.....14

Fig.1.8 Illustration of the backpropagation algorithm on a one-node, one hidden layer network..16

Fig.1.9 Gradient Descent visualized.....18

Fig.1.10 The basic architecture of an RNN.....19

Fig.1.11 The basic architecture of an AE.....20

Fig.1.12 Diagrammatic representation of a Convolutional Neural Network showing the main two operations of convolution and pooling.....21

Fig.1.13 Convolution operation.....22

Fig.1.14 Producing different feature maps with different kernels.....23

Fig.1.15 Left: Maxpooling, Right: Average Pooling.....23

Fig.1.16 Fully Connected Layer.....24

Fig.1.17 VGG16 Architecture Diagram (configuration-D).....25

Fig.1.18 Example of image classification with localization from two scenes, Left : a movie scene, Right : a camera shot.....28

Fig.1.19 Some examples of Object Segmentation on the COCO dataset.....28

Fig.1.20 Examples of image inpainting.....28

Fig.2.1 Biometric System Description.....32

Fig.2.2 Biometric modalities: (left) from top to bottom, Face, Fingerprint, Iris. (Right) from top to bottom: Palmprint, Ear, Gait. 34

Fig.2.3 Illustration of: (first row) Pose Variations. (second row) absence presence of structuring elements, and partial occlusion. (third row) Changes in facial expression. 38

Fig.2.4 Illustration of: (first row) illumination variations. (second row) variations of the image scale and resolution. (third row) Aging. 39

Fig.2.5 A global system framework for kinship verification. 41

Fig.2.6 Sample images from six different face kinship databases: Cornell KinFace, UvA-NEMO Smile, TSKinFace, KinFaceW, Family 101 and UB KinFace. 42

Fig.3.1 The proposed kinship verification system. 45

Fig.3.2 Feature extraction using VGG-Face. 47

Fig.3.3 Gabor kernel visualized for different settings. 49

Fig.3.4 Feature extraction using Hist-Gabor wavelet. 50

Fig.3.5 K-fold Cross Validation, with K = 5. 52

Fig.3.6 Cosine Similarity. 53

Fig.3.7 Fusion at the score level of VGG-Face and Gabor wavelet features. 53

Fig.3.8 ROC curves of best performing methods on Cornell KinFace database. 55

Fig.3.9 ROC curves of best performing methods TSKinFace database obtained on (a) F-S set, (b)F-D set, (c) M-S set and (d) M-D set, respectively. 57

List of Tables

Table 2.1 Summary of kinship databases in the literature. 43

Table 3.1 The mean accuracy (%) of kinship verification for TXQDA using our Hist-Gabor. on Cornell Kin Face database defining 6 scales, (table down): defining 20 blocs. 54

Table 3.2 the mean accuracy (%) of kinship verification for TXQDA using Hist-Gabor. on Cornell Kin Face database defining 20 blocs. 55

Table 3.3 Fusion of best performing features for our Hist-Gabor with 6 Scales/20blocks/200 features and VGG16 with 150 features on TSK kin Face database. 55

Table 3.4 The mean accuracy (%) of kinship verification for TXQDA using our Hist-Gabor with 6 scales on TSK kin Face database. 56

Table 3.5 The mean accuracy (%) of kinship verification for TXQDA using our Hist-Gabor with 28 blocks and different scales. on TSK KinFace database. 56

Table 3.6 Fusion of best performing features for our Hist-Gabor with 6 Scales/28blocks/200 features and VGG16 with 100 features on TSK KinFace database. 56

Table 3.7 Comparison of realized performance vs existing methods in TSKinFace. 59

Table 3.8 Comparison of realized performance vs existing methods in Cornell KinFace. 59

Table 3.9 Increase in mean accuracy of proposed approach against state-of-the art methods. . 59

List of Abbreviations

AE	Auto Encoder
AFR	Automatic Face Recognition
AI	Artificial Intelligence
API Application	Programming Interface
BSIF	Binarized Statistical Image Features
CIFAR	Canadian Institute for Advanced Research
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DIEDA	Discriminative Information based on Exponential Discriminant Analysis
DL	Deep Learning
DoG	Difference of Gaussian
FC	Fully Connected
GPU	Graphic Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
KNN	K-Nearest Neighbor
LBP	Local Binary Patterns
LDA	Linear Discriminant Analysis
LPQ	Local Phase Quantization
ML	Machine Learning
MLP	Multi-Layer Perceptron
MPCA	Multilinear Principal Component Analysis
MS COCO	Microsoft Common Objects in Context
MSE	Mean Squared Error
MSIDA	Multilinear Side Information Discriminant Analysis
NLP	Natural Language Processing
NN	Neural Network
PCA	Principal Component Analysis
ReLu	Rectified Linear Unit
RGB	Red Green Blue
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristics
SAE	Stacked Auto Encoder
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TXQDA	Tensor Cross-View Discriminant Analysis

General Introduction

The traditional way to verify kinship relations among humans is a chemical examination using DNA sequences. It is reliable and accurate but it is not always feasible or easy to implement due to various reasons. Kinship Verification through facial images is a growing new field in computer vision, it pulls the kinship verification process out of biochemistry labs to computer vision due to the fact that facial images can be useful to verify kin relations.

Facial images convey many important human characteristics, such as identity, gender, expression, age, and ethnicity. Over the past two decades, a large number of face analysis problems have been investigated in the computer vision and pattern recognition research community giving the rise to many other tasks like face recognition, facial expression recognition, facial age estimation, gender classification and ethnicity recognition. Facial kinship verification is a relatively new research topic in face analysis and only few attempts have been made over the recent years. However, this new research topic has several potential applications such as family album organization, image annotation, social media analysis, and missing children/parents search [1].

The advances in machine learning and deep learning capabilities, and deep convolutional neural networks (CNN) in particular, have been applied successfully in many fields of image processing and computer vision, leading to great achievement in those fields in recent years, and kinship verification is no exception.

The kinship verification system generally includes four main stages, the preprocessing stage, where facial images are processed in certain ways to correct illumination and pose and other variations. The features extraction stage, where these facial images are subject to many methods to extract useful information known as features, those features are fed to the similarity measurement stage to measure similarity between face samples then the verification stage outputs a binary decision for the input samples, kin or non kin. In this work we develop a new feature extraction method based on Gabor wavelet, called His-Gabor, along with using features extracted by a convolutional neural network called CNN, then we use fusion at the score level to take

advantage of the two methods. The key contributions of our work in order to improve the performance of kinship verification system is summed up as follows:

1. Proposing a new method to extract a solid discriminative handcrafted feature based on histograms of basic Gabor wavelet named His-Gabor, in which, the Gabor wavelets images are decomposed into several blocks to encode the facial structure. The histogram features based on Gabor wavelet give more microstructure information such as plane area, spots, orientations and edges. The histograms of all blocks are extracted and concatenated creating an enhanced feature vector.
2. Using a new Multilinear subspace approach for kinship verification called Tensor Cross-View Discriminant Analysis, TXQDA for short based on multidimensional representation of data, to provide a better discrimination between samples in the dataset.
3. To make a better use of multiple feature descriptors we propose a new fusion scheme based on multiple features, Hist-Gabor and VGG-Face, using logistic regression LR fusion at the score level.
4. The proposed approach TXQDA based fusion at score level of Hist-Gabor and VGG-face features is evaluated and compared against the state of-the-art methods using two challenging kinship databases called Cornell KinFace and TSKinFace. We proved that our method could achieve a high verification performance.

The remainder of our work is organized as follows: An Overview of Deep Learning and Artificial Intelligence and their applications are presented in chapter 1. Biometrics and Facial Kinship Verification systems are conducted in chapter 2. Finally, the implementation part which contains Kinship Verification Based VGG16 and Hist-Gabor Wavelet Features are presented in chapter 4.

Chapter 1

Deep Learning & Artificial Intelligence: An Overview

1.1 Introduction

Many interdisciplinary researches in several fields of physics, information technology, computer science, signal and image processing, electrical and electronic technologies have led to the development of Intelligent Systems.

These Intelligent Systems (or Intelligent Agents) represent the still more advanced and innovative frontier of research within the electronics and computer field, able to directly influence the quality of life, competitiveness, and production methods of companies, to watch and evaluate the environmental impact, to make public service and management activities more efficient, and to ensure people's safety. These intelligent systems are governed by the literature and the advances of a field called Artificial Intelligence [2]. Which is by itself governs a lot of sub-disciplines like Machine Learning, Neural Networks, Deep Learning, Fuzzy Logic…

There are many important applications of AI in the real world using a biometric system based on a biological characteristic of humans, naming automatic face recognition and verification, speech processing and recognition. Our work is a biometric system based on facial images to determinate a kinship relation between two humans.

In this chapter we review the basic concepts in the Machine Learning literature, its types and uses and its different algorithms. We also review the most common deep neural networks with focusing on convolutional neural networks.

1.2 Artificial Intelligence

Artificial intelligence (AI for short) is the field of embedding human thinking into machines, in other words, creating an artificial brain that mimics the functions of the biological brain. Whatever the human can do intelligently is now moved into machines. First-generation AI systems focused on problems that can be formally described by humans. steps for doing something intelligent are described in a form of instructions that machines

follow. Machines follow the human without changes. These features are characteristic of the first era of AI.

Humans can fully describe only simple problems such as the game of Tic-Tac-Toe or chess and cannot describe the more other complicated problems. In chess, the problem can be simply represented by a board matrix of size 8×8, describing each piece and its moves and describing the goals. Machines will be restricted to this task described by humans. By coding these instructions, machines can play chess intelligently. Machine intelligence is now said to be artificial but the machine itself is not intelligent, the humans have transferred his intelligence to it in a several static lines of code. By static, it is meant that the behavior does not change in all cases.

Embedding intelligent behavior inside lines of code can't handle all intelligent behaviors that humans perform. Some trivial tasks, such as sorting numbers or playing some games, can be described by humans and then handled by the machine with 100% of human intelligence. However, there are many complicated tasks, such as speech-to-text, image recognition, sentiment analysis, and many others, can't be solved by just static code, in fact these problems could not be described by the human as the case with chess. It is impossible to write code to recognize image objects such as cats or dogs. Recognizing objects is a type of intelligent behavior that can't be simply solved using a static code because there is no rule for classifying objects. Even if a rule is successfully created to recognize a certain object in one environment, it will definitely fail when applied in another [3]. Here comes what is known as AI, the goal here is not to hardcode instructions into machines but to teach machines to learn by themselves and find patterns in the data then give appropriate decisions. AI application extends to many subfields [4]:

- Machine Learning (ML).
- Data Mining.
- Information Retrieval and Semantic Web.
- Speech Recognition and Natural Language Processing.
- Image Processing/Recognition and Computer Vision.
- Robotics.
- Knowledge Representation and Knowledge Database.
- Logic Reasoning and Probabilistic Reasoning.
- Expert Systems.

1.3 Machine Learning

Definition 1. Machine learning is a subfield of computer science that is concerned with building algorithms which, to be useful, rely on a collection of examples of some phenomenon. These examples can come from nature, be handcrafted by humans or generated by another algorithm [5].

Definition 2. Machine learning can also be defined as the process of solving a practical problem by 1) gathering a dataset, and 2) algorithmically building a statistical model based on that dataset. That statistical model is assumed to be used somehow to solve the practical problem [5].

We are surrounded by a machine learning based technology: search engines learn how to bring us the best results (while placing profitable ads), anti-spam software learns to filter our email messages, and credit card transactions are secured by a software that learns how to detect frauds. Digital cameras learn to detect faces and intelligent personal assistance applications on smart-phones learn to recognize voice commands. Cars are equipped with accident prevention systems that are built using machine learning algorithms. Machine learning is also widely used in scientific applications such as bioinformatics, medicine, and astronomy [6].

Why do we need Machine Learning?

Why do we need machine learning and not explicitly program computers to carry out the task at hand? Two aspects of a given problem may need the use of programs that learn and improve on the basis of their "experience": the problem's complexity and the need for adaptivity.

Tasks that are way too complex to be programmed. There are many tasks that human beings perform routinely, yet the introspection concerning how those tasks are done is not sufficiently elaborate to extract a well-defined program. Examples of such tasks include driving, speech recognition, and vision (image understanding). In all of these tasks, state of the art machine learning programs, programs that "learn from their experience," achieve quite satisfactory results, once exposed to sufficiently many training examples [6].

Tasks beyond Human Capabilities. Another wide family of tasks that benefit from machine learning techniques are related to the analysis of very large and complex data sets: astronomical data, turning medical archives into medical knowledge, weather prediction, analysis of genomic data, Web search engines, and electronic commerce. With more and more available digitally recorded data, it becomes obvious that there are treasures of meaningful information buried in data archives that are way too large and too complex for humans to make sense of. Learning to detect meaningful patterns in large and complex data sets is a promising domain in which the combination of programs that learn with the almost unlimited memory capacity and ever-increasing processing speed of computers opens up new horizons.

Adaptivity. One limiting feature of programmed tools is their rigidity – once the program has been written down and installed, it stays unchanged. However, many tasks change over time or from one user to another. Machine learning tools – programs whose behavior adapts to their input data – offer a solution to such issues; they are, by nature,

adaptive to changes in the environment they interact with. Typical successful applications of machine learning to such problems include programs that decode handwritten text, where a fixed program can adapt to variations between the handwriting of different users; spam detection programs, adapting automatically to changes in the nature of spam e-mails; and speech recognition programs [6].

1.3.1 Types of Machine Learning

There are four types of machine learning [5], each one depends on the task at hand, they are classified as follows: Supervised, Unsupervised, Semi-Supervised and Reinforcement learning. This section explains these different types and the purpose of each one of them along with some examples.

1.3.1.1 Supervised Learning

In supervised learning, the dataset is the collection of labeled examples $\{(x_i, y_i)\}_{i=1}^{N}$. Each element x_i among N is called a feature vector. A feature vector is a vector in which each dimension $j = 1, \ldots, D$ contains a value that describes the example in a certain way. That value is called a feature and is denoted as $x^{(j)}$. The label y_i can be either an element belonging to a finite set of classes $\{1,2, \ldots, C\}$, or a real number, or a more complex structure, like a vector, a matrix, a tree, or a graph. For instance, if the examples are email messages and the problem is spam detection, then there are two classes {spam, not spam}[5].

The objective of a supervised learning algorithm is to use the dataset to produce a model that takes a feature vector x as input and outputs information that allows deducing the label for this feature vector. For instance, the model created using the dataset of people could take as input a feature vector describing a person and output a probability that the person has cancer [5]. There are many algorithms that fall in the supervised learning category, Linear and Logistic regression, support vector machine, decision trees, naïve Bayes… etc. Figure 1.1 illustrate the supervised learning algorithm, where three sets of data samples are given along with their labels, i.e., colors and shapes, after training the supervised learning model it should separate the data samples into three distinct categories depending on their given labels.

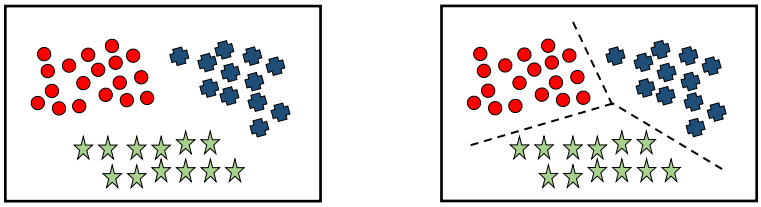


Fig.1.1 Supervised Learning

(i) Linear and logistic regression

Regression algorithms are a type of supervised algorithm that uses features of the input data to predict a value, such as the cost of a house, given certain features, such as size, age, number of bathrooms, number of floors, and location. Regression analysis tries to find the value of the parameters for the function that best fits an input dataset. In a linear-regression algorithm, the goal is to minimize a cost function by finding appropriate parameters for this function, over the input data that best approximates the target values. A cost function is a function of the error, that is, how far we are from getting a correct result. A popular cost function is the mean square error (MSE), where we take the square of the difference between the expected value and the predicted result. The sum overall the input examples gives us the error of the algorithm and represents the cost function [7].

(ii) Support Vector Machine

A support vector machine (SVM) is a supervised machine learning algorithm that is mainly used for classification. It is the most popular member of the kernel methods. The support vector machine algorithm tries to find a hyperplane, which separates the samples in the dataset. A hyperplane is a plane in a high-dimensional space. For example, a hyperplane in a one-dimensional space is a point, and in a two-dimensional space, it would just be a line. We can think of classification as a process of trying to find a hyperplane that will separate different groups of data points. Once we have defined our features, every sample in the dataset can be thought of as a point in the multi-dimensional space of features. One dimension of that space represents all the possible values of one feature. The coordinates of a point (a sample) are the specific values of each feature for that sample. The machine learning algorithm task will be able to draw a hyperplane to separate points with different classes. [7]

1.3.1.2 Unsupervised Learning

In unsupervised learning, the dataset is a collection of unlabeled examples $\{x_i\}_{i=1}^{N}$. Again, x is a feature vector, and the goal of an unsupervised learning algorithm is to create a model that takes a feature vector x as input and either transforms it into another vector or into a value that can be used to solve a practical problem. For example, in clustering, the model returns the identity (i.e. class) of the cluster for each feature vector in the dataset. In dimensionality reduction, the output of the model is a feature vector that has fewer features than the input x; in outlier detection, the output is a real number that indicates how x is different from a typical example in the dataset [5]. One example of this type is K-Means algorithm. Figure 1.2 illustrates an unsupervised learning algorithm, where data samples

are presented but without labeling, the objective of the algorithm is to find distinctive patterns to separate the data samples into many categories according the patterns found.

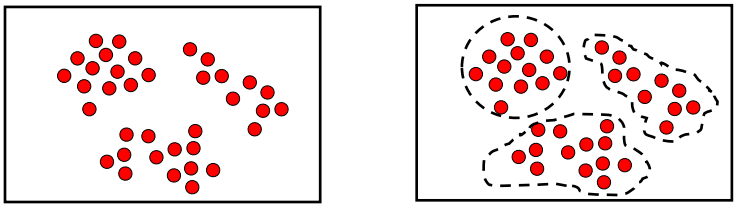


Fig.1.2 Unsupervised Learning.

1.3.1.3 Semi-Supervised Learning

In semi-supervised learning, the dataset contains both labeled and unlabeled examples. Usually, the quantity of unlabeled examples is much higher than the number of labeled examples. The goal of a semi-supervised learning algorithm is the same as the goal of the supervised learning algorithm. The hope here is that using many unlabeled examples can help the learning algorithm to find (we might say "produce" or "compute") a better model [5]. Figure 1.3 illustrates a semi-supervised algorithm, where data samples are given, the data contains labeled and unlabeled samples, so the objective the algorithm is to find similarities between labeled and unlabeled data and produce separation according to the pattern.

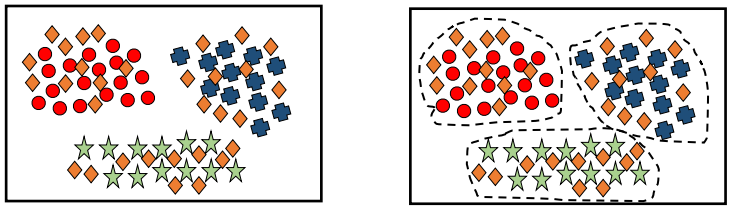


Fig.1.3 Semi-supervised learning

1.3.1.4 Reinforcement Learning

Reinforcement learning is another subfield of machine learning where the machine "lives" so to speak in an environment and is capable of perceiving the state of that environment as a vector of features. The machine can execute actions in every state. Different actions bring different rewards (responses) and could also move the machine to another state of the environment. The goal of a reinforcement learning algorithm is to learn a policy. A policy is a function f (similar to the model in supervised learning) that takes the feature vector of a state as input and outputs an optimal action to execute in that state. The action is optimal if it maximizes the expected average response [5]. One example of reinforcement learning is Q-Learning. Reinforcement learning solves a particular kind of problems where decision making is sequential, and the goal is long-term, such as gameplaying, robotics, resource management, or logistic [5].

1.3.2 Data for machine learning

Typically, a problem that is solvable with classic machine learning techniques may require a thorough understanding and processing of the training data before deployment. The steps to solve an ML problem are as follows [7]:

Data collection: This implies the gathering of as much data as possible. In the case of supervised learning, this also includes correct labeling.

Data processing: This implies cleaning the data, such as removing redundant or highly correlated features, or even filling missing data, and understanding the features that define the training data.

Usually, the data can be divided into three sets:

Training set: this set is used to train the ML algorithm.

Validation set: this set is used to evaluate the accuracy of the algorithm with unknown data during training. The algorithm is trained for some time on the training set and then the validation set is used to check its performance. If the results are not satisfying, (i.e. change) the hyper parameters of the algorithm will be tuned (i.e. changed) and repeat the process all over again. The validation set can also help us to determine when to stop the training process.

Test set: When tuning the algorithm is done with the training or validation cycle, the test set is now used only once for a final evaluation. The test set is similar to the validation set in the sense that the algorithm hasn't used it during training. However, when striving to improve the algorithm on the validation data, a bias might be inadvertently introduced, which can skew the results in favor of the validation set and not reflect the actual performance. Because the test is used only once, this will provide a more objective measurement of the algorithm at hand [7].

Each type of machine learning involves maximizing or minimizing a function during the training phase. This function is called the loss function, known also as cost or error function, and it evaluates how the machine learning model fits the data. Below, we review some widely used loss functions. N stands for the size of the training set, y_i and $\hat{y}_i$ are respectively the label and the prediction (i.e. the output of the model and the desired output) for the i^{th}example, and I is the indicator function:

- **01-Loss:** called accuracy as well, used widely in tasks that involves classification. Given by the formula:

$$\frac{1}{n}\sum_{i=1}^{n} I(\hat{y}_i - y_i) \qquad 1.1$$

- **Mean squared error:** used mostly for tasks involving regression, its formula is given by:

$$\frac{1}{n}\sum_{i=1}^{n} I(\hat{y}_i - y_i)^2 \quad 1.1$$

- **Binary cross-entropy loss:** also used in classification tasks, it is given by:

$$-\frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \quad 1.2$$

1.3.3 Artificial Neural Networks

Artificial neural networks are popular machine learning techniques that simulate the mechanism of learning in biological organisms. The human nervous system contains cells, which are referred to as neurons. The neurons are connected to one another with the use of axons and dendrites, and the connecting regions between axons and dendrites are referred to as synapses.

These connections are illustrated in Fig.1.4 The strengths of synaptic connections often change in response to external stimuli. This change is how learning takes place in living organisms. This biological mechanism is simulated in artificial neural networks, which contain computation units that are referred to as *neurons*. Throughout this work, we will use the term "neural networks" to refer to artificial neural networks rather than biological ones. The computational units are connected to one another through *weights*, which serve the same role as the strengths of synaptic connections in biological organisms. Each input to a neuron is scaled with a weight, which affects the function computed at that unit. Figure 1.4 represents the biological structure of a neuron, in the brain, each neuron constitutes an elementary processing unit and figure 1.5 illustrates the mathematical model created after the biological structure, it's a simulation of the behavior of this structure.

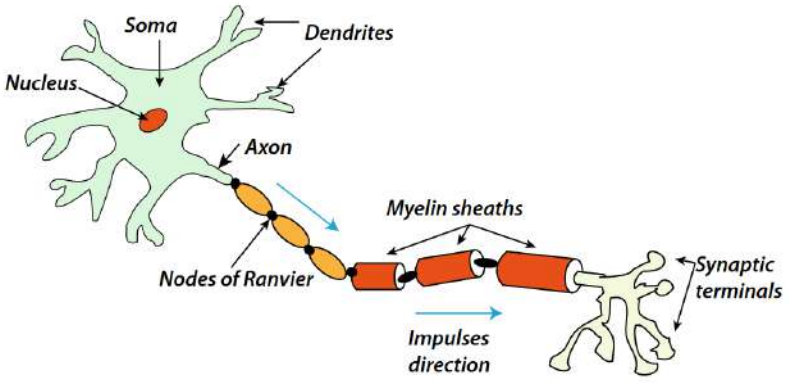


Fig.1.4 Biological structure of a neuron [8].

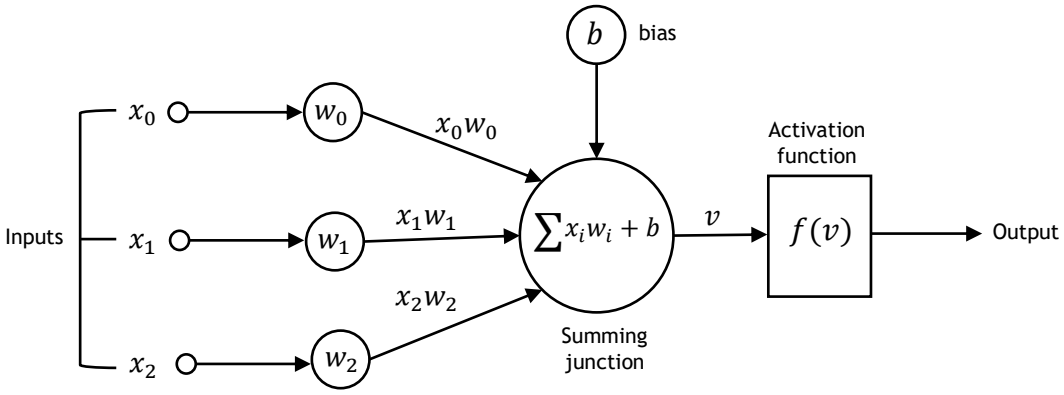


Fig.1.5 Mathematical model of an artificial Neuron [9].

A neuron can represent any significant entity (for example, a pixel of an image, an element of the features vector) that must be processed with a neural model. The essential components of the neuron.

An elementary unit of neural network processing, are:

- the set of synaptic inputs.
- the adder of input signals, and
- the activation function that generates the output value of the neuron

This model predicts a neuron represented by N input values $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ that model the signals coming from dendrites. These signals are associated with synaptic weights $\boldsymbol{w} = (w_1, w_2, \ldots, w_3)$ which measure their permeability. According to the neuro-biological motivation some of these synaptic weights are negative to express their inhibitory characteristic [10]. The weighted sum of the input values x_i represents the excitation level of a neuron.

$$\boldsymbol{\xi} = \sum_{i=1}^{N} w_i x_i \tag{1.3}$$

The excitation level $\boldsymbol{\xi}$ compared to a suitable $\boldsymbol{\theta}$ threshold associated with the neuron determines its final state, i.e., it produces the output y of the neuron that models the electric-chemical signal generated by the axon. The nonlinear growth of the output value y after the excitation value has reached the $\boldsymbol{\theta}$ threshold level is determined by the activation function (i.e., the transfer function) σ for which, we have:

$$y = \sigma(\boldsymbol{\xi}) = \sigma\left(\sum_{i=1}^{N} w_i x_i\right) = \begin{cases} 1 \; if \; \boldsymbol{\xi} \geq \boldsymbol{\theta} \\ 0 \; if \; \boldsymbol{\xi} < \boldsymbol{\theta} \end{cases} \tag{1.4}$$

1.3.3.1 Multilayer Perceptron (MLP)

Multilayer perceptron or Multilayer neural networks contain more than one computational layer. The perceptron contains an input and output layer, of which the output layer is the only computation-performing layer. The input layer transmits the data to the output layer, and all computations are completely visible to the user. Multilayer neural networks contain multiple computational layers; the additional intermediate layers (between input and output) are referred to as hidden layers because the computations performed are not visible to the user [11]. In simplest terms, the network can be treated as a black box, which operates on a set of inputs and generates some outputs [12]. The data transmission from the input layer to the output layer called a feed forward operation or forward propagation, the opposite operation called backpropagation. Figure 1.6 illustrates the architecture of a simple feed forward neural network, it consists of an input layer, three hidden layers and an output layer. Each circle represents an artificial neuron or a node, the neurons are connected with each other by connections, each connection is a weight.

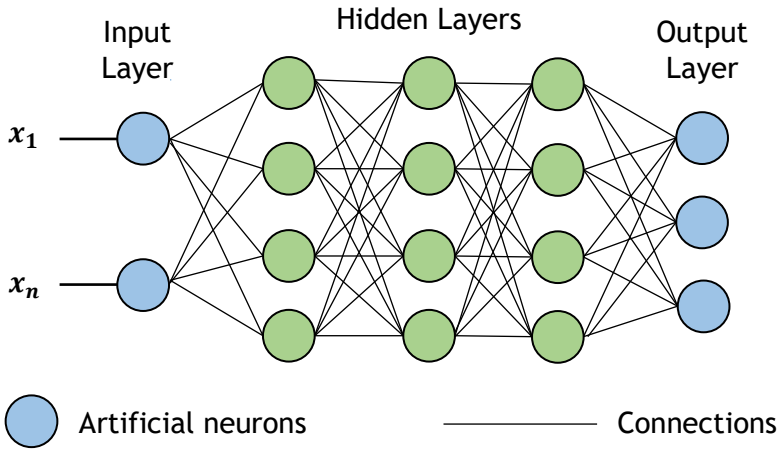


Fig.1.6 Simple feed forward network.

Layered Architecture: Neural networks comprise a hierarchy of processing levels. Each level is called a "network layer" and consists of a number of processing "nodes" (also called "neurons" or "units"). Typically, the input is fed through an input layer and the final layer is the output layer which makes predictions. The intermediate layers perform the processing and are referred to as the hidden layers. Due to this layered architecture, this neural network is called an MLP.

Nodes: The individual processing units in each layer are called the nodes in a neural net-work architecture. The nodes basically implement an "activation function" which given an input, decides whether the node will fire or not.

Dense Connections: The nodes in a neural network are interconnected and can communicate with each other. Each connection has a *weight* which specifies the strength of the connection between two nodes. For the simple case of feed-forward neural networks, the information is transferred sequentially in one direction from the input to the output layers. Therefore, each node in a layer is directly connected to all nodes in the immediate previous layer.[13]

1.3.3.2 Activation Functions:

Activation functions live inside neural network layers and modify the data they receive before passing it to the next layer. Activation functions give neural networks their power—allowing them to model complex non-linear relationships. By modifying inputs with non-linear functions, neural networks can model highly complex relationships between features. Popular activation functions include ReLu and Sigmoid. Activation functions typically have the following properties:

Non-linear - In linear regression we're limited to a prediction equation that looks like a straight line. This is nice for simple datasets with a one-to-one relationship between inputs and outputs, but what if the patterns in our dataset were non-linear? (e.g. x2, sin, log). To model these relationships, we need a non-linear prediction equation. Activation functions provide this non-linearity.

Continuously differentiable - To improve our model with gradient descent, we need our output to have a nice slope so we can compute error derivatives with respect to weights. If our neuron instead outputted 0 or 1 (perceptron), we wouldn't know in which direction to update our weights to reduce our error.

Fixed Range - Activation functions typically squash the input data into a narrow range that makes training the model more stable and efficient [14].

The following Figure 1.7, shows some commonly used activation functions in the literature of neural networks. Each activation function has its pros and cons, for example the sigmoid function is historically popular due to its biological interpretation, but suffers from certain issues, because when neurons are saturated the gradients will vanish, the outputs are not zero centered and the $exp(.)$ function is computationally expensive. On the other hand, the ReLu function is computationally efficient, converges much faster and it's somehow biologically plausible then the Sigmoid, but its outputs are not centered around zero, hence the Leaky ReLu and many other activation functions were introduced to exceed the limitations of previous ones.

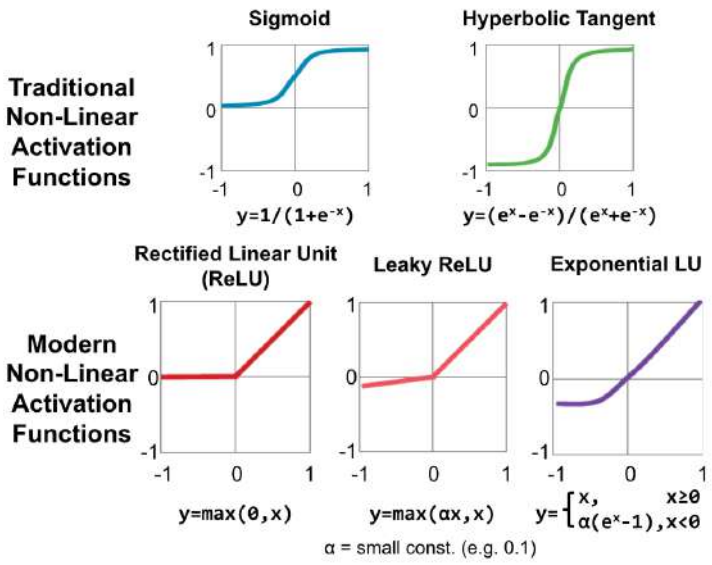


Fig.1.7 Commonly used activation functions.

1.3.3.3 Learning in a neural network

As we described previously, the weights of a neural network define the connections between neurons. These weights need to be set appropriately so that a desired output can be obtained from the neural network. The weights encode the "model" generated from the training data that is used to allow the network to perform a designated task (e.g., object detection, recognition, and/or classification). In practical settings, the number of weights is huge which requires an automatic procedure to update their values appropriately for a given task. The process of automatically tuning the network parameters is called *Learning* which is accomplished during the training stage (in contrast to the test stage where inference/prediction is made on "unseen data," i.e., data that the network has not "seen" during training). This process involves showing examples of the desired task to the network so that it can learn to identify the right set of relationships between the inputs and the required outputs.

For example, in the paradigm of supervised learning, the inputs can be media (speech, images) and the outputs are the desired set of "labels" (e.g., identity of a person) which are used to tune the neural network parameters [15].

The simplest example of backpropagation and how the gradient descent is to be performed is the following: the input-to-output relationship for a single node, one hidden layer network, is given by

$$y = g(z, w_2) = g(f(x, w_1), w_2) \quad 1.5$$

Thus, given a function $f(\cdot)$ and $g(\cdot)$ with weighting constants $w_1$and w_2, the output error produced by the network can be computed against the ground truth as

$$E = \frac{1}{2}(y_0 - y)^2 \quad 1.6$$

where y_0 is the correct output and y is the neural network's approximation to the output. The goal is to find a and b to minimize the error. The minimization requires

$$\frac{\partial E}{\partial w_1} = -(y_0 - y)\frac{\partial y}{\partial z}\frac{\partial z}{\partial w_1} = 0 \quad 1.8$$

A critical observation is that the compositional nature of the network along with the chain rule forces the optimization to backpropagate error through network. In particular, the terms ${\partial y}/{\partial z}\,{\partial z}/{\partial w_1}$ show how this backprop occurs. Given functions $f(\cdot)$ and $g(\cdot)$, the chain rule can be explicitly computed. Backprop results in an iterative, gradient descent update rule

$${w_1}^{k+1} = {w_1}^{k}\delta\frac{\partial E}{\partial {w_1}^{k}} \quad 1.7$$

$${w_2}^{k+1} = w_2\delta\frac{\partial E}{\partial {w_2}^{k}} \quad 1.8$$

Where δ is the so-called learning rate and ${\partial E}/{\partial w_1}$ along with ${\partial E}/{\partial w_2}$ can be explicitly computed using (1.18). The iteration algorithm is executed to convergence. As with all iterative optimization, a good initial guess is critical to achieve a good solution in a reasonable amount of computational time.

The idea behind gradient descent, or steepest descent, is to use the derivative information as the basis of an iterative algorithm that progressively converges to a local minimum point of $f(x)$, as shown in the Fig.1.8.

Backpropagation algorithm proceeds as follows:

(i) A neural network is specified along with a labeled training set.

(ii) The initial weights of the network are set to random values. Importantly, one must not initialize the weights to zero, similar to what may be done in other machine learning algorithms. If weights are initialized to zero, after each update, the outgoing weights of each neuron will be identical, because the gradients will be identical. Moreover, neural nets often get stuck at local optima where the gradient is zero but that are not global minima, so random weight initialization allows one to have a chance of circumventing this by starting at many different random values.

(iii) The training data is run through the network to produce an output y, whose ideal ground-truth output is y_0. The derivatives with respect to each network weight is then computed using backpropagation formulas (1.8).

(iv) For a given learning rate δ, the network weights w_1 and w_2 are updated as in (1.9) and (1.10).

(v) We return to step (iii) and continue iterating until a maximum number of iterations is reached or convergence is achieved.

As a simple example, consider the linear activation function

$$f(\varepsilon, \alpha) = g(\varepsilon, \alpha) = \alpha\varepsilon \tag{1.9}$$

In this case we have in fig 1.8

$$\begin{aligned} z &= w_1 x \\ y &= w_2 z \end{aligned} \tag{1.12}$$

We can now explicitly compute the gradients such as (1.8). This gives

$$\frac{\partial E}{\partial w_1} = -(y_0 - y)\frac{\partial y}{\partial z}\frac{\partial z}{\partial w_1} = -(y_0 - y) \cdot w_2 \cdot x \tag{1.10}$$

$$\frac{\partial E}{\partial w_2} = -(y_0 - y)\frac{\partial y}{\partial w_2} = -(y_0 - y)z = -(y_0 - y) \cdot w_1 \cdot x \tag{1.11}$$

Thus, with the current valued of w_1 and w_2 along with the input output pair x and y and target truth y_0 each derivative can be evaluated. This provides the required information to perform the updates (1.9) and (1.10).

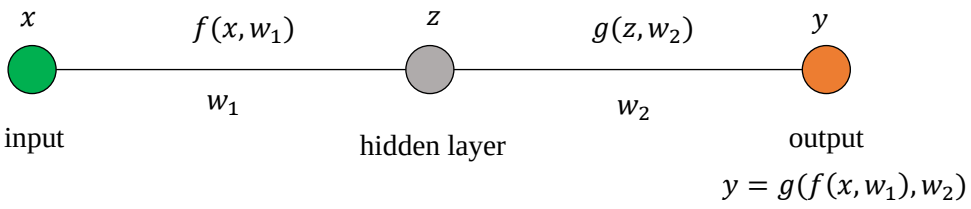


Fig.1.8 Illustration of the backpropagation algorithm on a one-node, one hidden layer network.

The backprop for a deeper net follows in a similar fashion. Consider a network with M hidden layers labeled z_1 to $z_m$with the first connection weight $w_1$between x and z_1. The generalization of Fig.I.8 and equation (1.8) is given by

$$\frac{\partial E}{\partial w_1} = -(y_0 - y)\frac{\partial y}{\partial z_m}\frac{\partial z_m}{\partial z_{m-1}}\cdots\frac{\partial_{z_2}}{\partial z_1}\frac{\partial z_1}{\partial z_{w_1}} \quad 1.12$$

The cascade of derivates induced by the composition and chain rule highlights the backpropagation of errors that occurs when minimizing the classification error. A full generalization of backprop involves multiple layers as well multiple nodes per layer. The objective is to determine the matrix elements of each matrix A_j. Thus, a significant number of network parameters need to be updated in gradient descent. Indeed, training a network can often be computationally infeasible even though the update rules for individual weights is not difficult. Neural Nets can thus suffer from the curse of dimensionality as each matrix from one layer to another requires updating n^2 coefficients for a n-dimensional input, assuming the two connected layers are both n-dimensional. Denoting all the weights to be updated by the vector w, where w contains all the elements of the matrices A_j, then:

$$w_{k+1} = w_k + \delta\nabla E \quad 1.13$$

where the gradient of the error ∇E, through the composition and chain rule, produces the backpropagation algorithm for updating the weights and reducing the error. Expressed in a component-by-component way

$$w_{k+1}^j = w_k^j + \delta\frac{\partial E}{\partial w_k^j} \quad 1.14$$

where this equation holds for the jth component of the vector w. The term $\partial E/\partial w^j$ produces the backpropagation through the chain rule, i.e. it produces the sequential set of functions to evaluate as in (1.15). Methods for solving this optimization more quickly, or even simply enabling the computation to be tractable, remain of active research interest.

Figure 1.9 shows the Gradient descent algorithm applied to the function $f(x, y) = x^2 + 3y^2$. In the top panel, the contours are plotted for each successive value (x, y) in the iteration algorithm given the initial guess $(x, y) = (3,2)$. The bottom panel demonstrates the rapid convergence and error (E) to the minimum (optimal) solution.

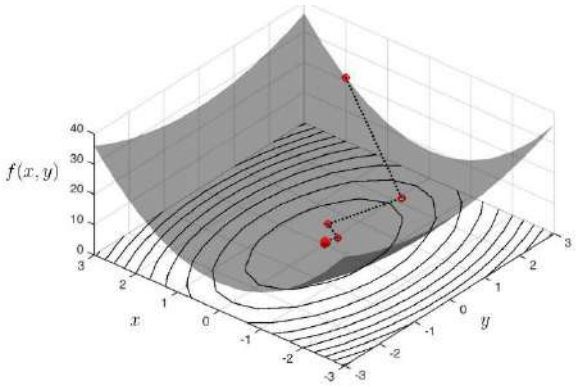


Fig.1.9 Gradient Descent visualized.

Although Gradient Descent is a simple optimization algorithm but it has quite a lot of problems that could come up in practice. First the Gradient Descent might get stuck in a local mimima when evaluating the loss function, this problem occurs much more in high dimensions. to solve this issues researchers proposed many variants of Gradient Descent, like Stochastic Gradient Descent SGD, SGD with momentum [16], AdaGrad [17], AdaMax, RMSProp [18], Adam Optimizer [19]… etc.

1.4 Deep Learning

In recent years, starting in 2000, with the advancement of Machine Learning research, the neural network sector has vigorously recovered through further availability of low-cost multiprocessor computing systems (computer clusters, Central Processing Units (CPU), Graphical Processing Units (GPU)... etc.), the need to elaborate large amounts of data (big data) in various applications (industries, public administration, social media, communications... etc.), and above all through the development of new algorithms for automatic learning applied for computer vision, speech recognition, textual and language [10]. This advancement of research on neural networks has led to the development of new algorithms for machine learning-based also on the architecture of traditional neural networks developed since the 1980s. The strategic progress was with the best results achieved with the new neural networks developed, called Deep Learning (DL) [10]. Traditional machine learning algorithms typically define a set of rules or features that you want to extract from the data. Hand engineered features are time consuming and not scalable in practice. The key insight of Deep Learning is to learn features directly from raw data [20].

1.4.1 Deep Neural Networks

When a simple neural network (discussed in previous sections) has more than one hidden layer, it is known as a Deep Neural Network (DNN). We will discuss four major architectures of deep neural networks: Recurrent Neural Networks (RNNs), Autoencoders, Deep Belief Networks (DBNs), Convolutional Neural Networks (CNNs). These

architectures are fundamental ones and can be extended or combined to generate new frameworks for specific tasks.

1.4.1.1 Recurrent Neural Networks (RNN)

RNN is a type of neural network, which can process sequential data with variable length. Examples of such data include the words of a sentence or the price of a stock in various moments of time. By using the word sequential, we imply that the elements of the sequence are related to each other and their order matters. For example, if we take a book and shuffle randomly all the words in it, the text will lose its meaning, even though we'll still know the individual words. RNNs get their name because they apply the same function over a sequence recurrently. We can define an RNN as a recurrence relation:

$$s_t = f(s_{t-1}, x_t) \tag{1.15}$$

Here, f is a differentiable function, s_t is a vector of values called internal network state (at step t), and x_t is the network input at step t. Unlike regular networks, where the state only depends on the current input (and network weights), here s_t is a function of both the current input, as well as the previous state s_{t-1}. You can think of s_{t-1} as the network's summary of all previous inputs. The recurrence relation defines how the state evolves step by step over the sequence via a feedback loop over previous states, as illustrated in the following diagram:

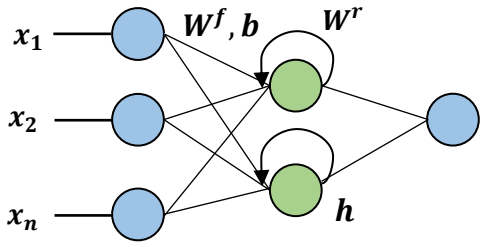


Fig.1.10 The basic architecture of an RNN.

1.4.1.2 Autoencoders

Autoencoders (AEs) are simple networks directly cascading many layers, which have a similar forward structure with multiple-layer perceptron (MLPs). However, the Auto Encoders learn the parameters by reconstructing the input signals precisely, while MLPs use supervised information to train models[21]. For the sake of analysis, AEs, reconstructing the input $\boldsymbol{x}$ of size n on the output $\tilde{\boldsymbol{x}}$ with only one hidden layer $\boldsymbol{h}$ of size k, are shown in Fig.1.11. The reconstruction is computed as follows:

$$\begin{aligned} \boldsymbol{h} &= f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \\ \tilde{\boldsymbol{x}} &= \tilde{f}(\widetilde{\boldsymbol{W}}\boldsymbol{h} + \widetilde{\boldsymbol{b}}) \end{aligned} \tag{1.19}$$

Where $\boldsymbol{W}$ and $\widetilde{\boldsymbol{W}}$ are weight matrices. $\boldsymbol{b}$ and $\widetilde{\boldsymbol{b}}$ are biases. If the hidden layer h has the same size as the input vector and use a linear function, the model will simply learn the identity function. If the dimensionality k of his smaller than n of x, the latent representations can be learned according to the equation. Using AEs, the original data can be projected into a lower-dimensional subspace, representing latent structures in the input. So, AEs are usually used for unsupervised learning of efficient coding. Moreover, various techniques have been adopted to prevent AEs from learning the identity function and obtain many variants. For instance, the regularization or sparsity constraints can further be employed to discover relevant structure more precisely and improve their ability to capture important information.

In order to prevent the model from learning a trivial solution, the denoising AEs were proposed by *Vincent et al.* [58]. In the denoising method, the model is trained to reconstruct the input from a noise-corrupted version (typically salt-and-pepper noise). Figure 1.11 illustrates the architecture of an AE.

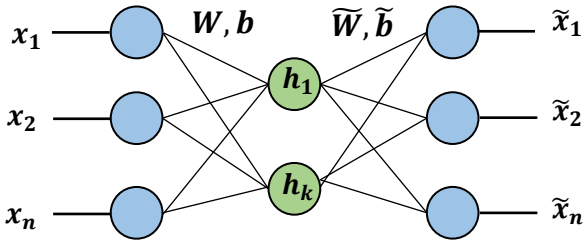


Fig.1.11 The basic architecture of an AE.

1.4.1.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs or ConvNets) are one of the most popular categories of neural networks, especially for high-dimensional data (e.g., images and videos) and they are currently the state of the art in visual recognition. CNNs operate in a way that is very similar to standard neural networks. A key difference, however, is that each unit in a CNN layer is a two - (or high-) dimensional filter which is convolved with the input of that layer. This is essential for cases where we want to learn patterns from high-dimensional input media, e.g., images or videos [13].

CNN filters incorporate spatial context by having a similar (but smaller) spatial shape as the input media, and use parameter sharing to significantly reduce the number of learn-able variables.

At its most basic, convolutional neural networks can be thought of as a kind of neural network that uses many identical copies of the same neuron. This allows the network to have lots of neurons and express computationally large models while keeping the number of actual parameters – the values describing how neurons behave – that need to be learned fairly small.

This trick of having multiple copies of the same neuron is roughly analogous to the abstraction of functions in mathematics and computer science. When programming, we write a function once and use it in many places – not writing the same code a hundred times in different places makes it faster to program, and results in fewer bugs. Similarly, a convolutional neural network can learn a neuron once and use it in many places, making it easier to learn the model and reducing error [22].

Architecture of CNNs

A CNN comprises three main types of neural layers, namely, (i) Convolutional layers, (ii) Pooling layers, and (iii) Fully connected layers. Each type of layer plays a different role. Every layer of a CNN transforms the input volume to an output volume of neuron activation, eventually leading to the final fully connected layers, resulting in a mapping of the input data to a 1-D feature vector which later – generally - will be used in classification tasks in the output layer. Figure 1.12 illustrates the architecture of a CNN with description of layers.

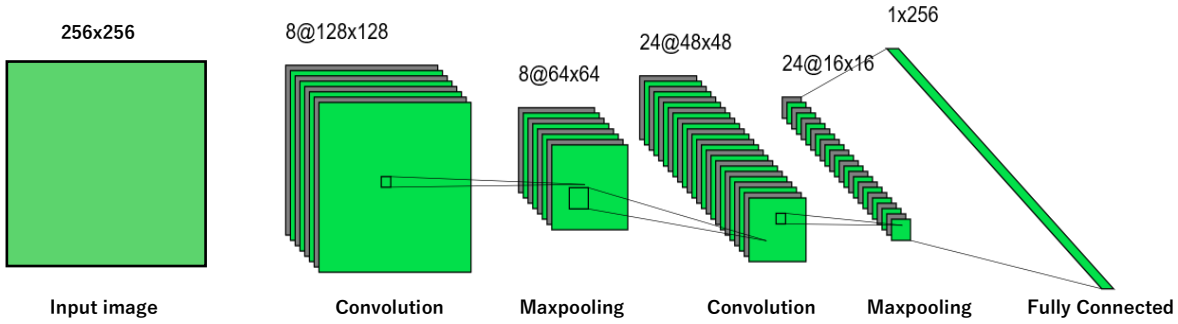


Fig.1.12 Diagrammatic representation of a Convolutional Neural Network showing the main two operations of convolution and pooling.

Convolutional Layers: A convolutional layer (Conv.) is the most important component of a CNN. It comprises a set of filters (also called kernels) which are convolved with a given input to generate various outputs called *feature maps* in a process called convolution.

What is a filter / kernel?

An image filter is a small matrix $k \times k$ used to apply effects like the ones you might find in Photoshop or any other image editing software, such as blurring, sharpening, outlining or embossing. In CNN, kernels are used for ‘’ *feature extraction* ‘’, a technique for determining the most important features of an image. The process of using a kernel is called a ‘’ *convolution* ‘’. The weights of each filter (the numbers in the grid) are learned during the training of CNN. This learning procedure involves a random initialization of the filter weights at the start of the training.

What is convolution?

The core of neural networks is affine transformations: a vector is received as input and is multiplied with a matrix to produce an output (to which a bias vector is usually added before passing the result through a non-linearity). This is applicable to any type of input, be it an image, a sound clip or an unordered collection of features: whatever their dimensionality, their representation can always be flattened into a vector before the transformation. Images, sound clips and many other similar kinds of data have an intrinsic structure. More formally, they share these important properties:

They are stored as multi-dimensional arrays. They feature one or more axes for which ordering matters (e.g., width and height axes for an image, time axis for a sound clip).

One axis, called the channel axis, is used to access different views of the data. (e.g., the red, green and blue channels of a color image, or the left and right channels of a stereo audio track). These properties are not exploited when an affine transformation is applied; in fact, all the axes are treated in the same way and the topological information is not taken into account. Still, taking advantage of the implicit structure of the data may prove very handy in solving some tasks, like computer vision and speech recognition, and in these cases, it would be best to preserve it. This is where discrete convolutions come into play. A discrete convolution is a linear transformation that preserves this notion of ordering (Spatial information in images). It is sparse (only a few input units contribute to a given output unit) and reuses parameters (the same weights are applied to multiple locations in the input) [23].

The mathematical aspect of 2D convolution is shown in equation below:

$$\boldsymbol{y(i,j) = x(i,j) * h(i,j) = \sum_{k=1}^{n}\sum_{l=1}^{m} x(k,l) \cdot h(i-k, j-l)} \qquad 1.20$$

As the figure 1.13 below shows, the 8×8 input image on the left is convolved with the 3×3 filter in the middle to yield the image on the right. The light blue pixels indicate the source neighborhood for the light green destination pixel [24].

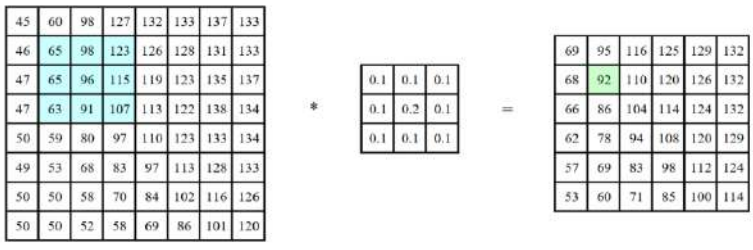


Fig.1.13 Convolution operation.

Figure 1.14 shows some effects of changing filter's weights on the input image to create a feature, those weights (numbers on the grid) are learned automatically during the training of the network.



Fig.1.14 Producing different feature maps with different kernels [20].

Pooling Layers: Pooling layers are in charge of reducing the spatial dimensions (width × height) of the input volume for the next convolutional layer. The pooling layer does not affect the depth dimension of the volume. The operation performed by this layer is also called subsampling or down-sampling, as the reduction of size leads to a simultaneous loss of information. However, such a loss is beneficial for the network because the decrease in size leads to less computational over-head for the upcoming layers of the network, and also it works against overfitting. Figure 1.15 shows two different types of pooling, Maxpooling where the max value is taken, and Average pooling where the average vale between neighboring pixels is taken,

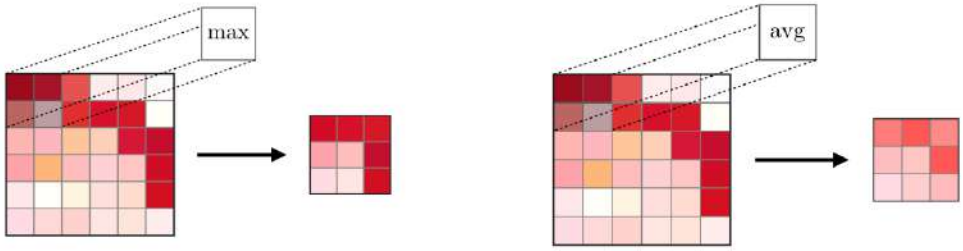


Fig.1.15 Left: Maxpooling, Right: Average Pooling.

Fully Connected Layers: Following several convolutional and pooling layers, the high-level reasoning in the neural network is performed via fully connected layers. Neurons in a fully connected layer have full connections to all activation in the previous layer, as their name implies. Their activation can hence be computed with a matrix multiplication followed by a bias offset. Fully connected layers eventually convert the 2D feature maps into a 1D feature vector. The derived vector either could be fed forward into a certain number of categories for classification [31] or could be considered as a feature vector for further processing [32]. Figure 1.16 illustrates the fully connected layer.

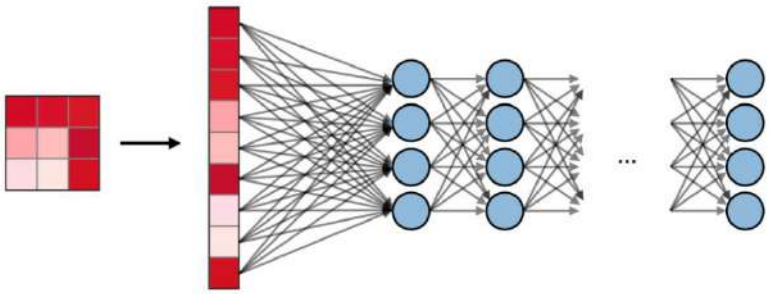


Fig.1.16 Fully Connected Layer.

There are various architectures of CNN models available which have been a key in building algorithms which power and shall power the field of computer vision and AI as a whole in the foreseeable future: Lenet, AlexNet, GoogleNet, ResNet, ZFNet, VGGNet…

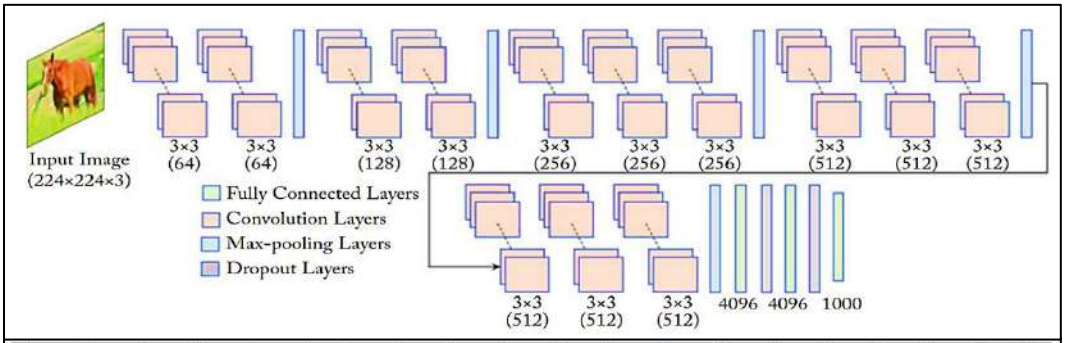
1.4.1.4 CNN Case Study: VGG16

VGG is a convolutional neural network model proposed by *K. Simonyan* and *A. Zisserman* from the University of Oxford in their paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". [25] VGG stands for Oxford's Visual Geometry Group. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. The VGGNet architecture strictly uses 3x3 convolution kernels with intermediate max-pooling layers for feature extraction and a set of three fully connected layers toward the end for classification. Each convolution layer is followed by a ReLU layer in the VGGNet architecture. The design choice of using smaller kernels leads to a relatively reduced number of parameters, and therefore an efficient training and testing. Moreover, by stacking a series of 3x3 sized kernels, the effective receptive field can be increased to larger values (e.g., 5x5 with two layers, 7x7 with three layers, and so on). Most importantly, with smaller filters, one can stack more layers resulting in deeper networks, which leads to a better performance on vision tasks. This essentially conveys the central idea of this architecture, which supports the usage of deeper networks for an improved feature learning.

Architecture

The input to VGG ConvNet is a fixed-size 224×224 RGB image. The only pre-processing done is subtracting the mean RGB value, computed on the training set, from each pixel. The image is passed through a stack of convolutional (conv.) layers, where filters with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center) are used. In one of the configurations utilize 1×1 convolution filters are used, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution,

i.e. the padding is 1 pixel for 3 × 3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2 × 2 pixel window, with stride of 2. A stack of convolutional layers (which has a different depth in different architectures) is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks. All hidden layers are equipped with the rectification (ReLU) non-linearity. VGG networks (except for one) do not contain Local Response Normalization (LRN) normalization because such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time. Where applicable, the parameters for the LRN layer are those of AlexNet [25]. The upper part of the following figure 1.17 illustrates a diagram representation of the VGG16 Architecture with specifications of each layer, the bottom part represents a variant of VGG-16 called VGG-Face also with details of each layer: layer type, filter dimensions, number of filters, stride, padding. We'll use VGG-Face in our implementation part in chapter 3.



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

Fig.1.17 VGG16 Architecture Diagram (configuration-D)[26] [27].

1.4.1.5 Deep Learning Frameworks

There have been a lot of interest from academics (e.g., The University of California Berkeley, New York University, The University of Toronto, The University of Montreal) and industry groups (e.g., Google, Facebook, Microsoft) to develop deep learning frameworks. It is mainly due to their popularity in many applications domains over the last few years. The key motivation for developing these libraries is to provide an efficient and

friendly development environment for researchers to design and implement deep neural networks. Some of the widely used deep learning frameworks are: Caffe[1], TensorFlow[2], MatConvNet[3] (MATLAB), Torch7[4], Theano[5], Keras[6], Lasange, Marvin, Chainer, DeepLearning4J, and MXNet [13]. We'll discuss both TensorFlow and MATLAB MatConvNet Toolbox since we are going to use one of them in our implementation.

TensorFlow: TensorFlow was originally developed by the Google Brain team, written with a Python API over a C/C++ engine for numerical computation using data flow graphs. Multiple APIs have been provided. The complete programming control is provided with the lowest level APIs, called TensorFlow Core. Machine learning researchers and others who need fine levels of control over their models are recommended to use the TensorFlow Core. The higher-level APIs are built on top of TensorFlow Core and they are easier to learn and use, compared to the TensorFlow Core. TensorFlow offers automatic differentiation capabilities, which simplify the process of defining new operations in the network. It uses data flow graphs to perform numerical computations. The graph nodes represent mathematical operations and the edges represent tensors. TensorFlow supports multiple backends, CPU or GPU on desktop, server or mobile platforms. It has well-supported bindings to Python and C++. TensorFlow also has tools to support reinforcement learning. For more details, we suggest the readers to visit the TensorFlow website [13].

MatConvNet MATLAB Toolbox: MatConvNet is an open source implementation of Convolutional Neural Networks (CNNs) with a deep integration in the MATLAB environment. The toolbox is designed with an emphasis on simplicity and flexibility. It exposes the building blocks of CNNs as easy-to-use MATLAB functions, providing routines for computing convolutions with filter banks, feature pooling, normalization, and much more. MatConvNet can be easily extended, often using only MATLAB code, allowing fast prototyping of new CNN architectures. At the same time, it supports efficient computation on CPU and GPU, allowing to train complex models on large datasets such as ImageNet ILSVRC containing millions of training examples [28].

1.4.2 Applications of Deep Learning

1.4.2.1 Computer Vision

Computer vision has traditionally been among one of the most active research disciplines in deep learning applications, because vision is a task that is effortless for

[1] caffe.berkeleyvision.org
[2] tensorflow.org
[3] vlfeat.org / matlab.com
[4] torch.ch
[5] deeplearning.net/software/theano/
[6] keras.io

humans and many animals but quite challenging for computers. Many of the most popular standard benchmark tasks for deep learning algorithms are variants of object recognition or optical character recognition. Computer vision is a very broad field encapsulating a wide variety of methods of processing images, and diversity of applications. Applications of computer vision range from reproducing human visual abilities, such as recognizing faces, to creating entirely new categories of visual abilities.

Image classification (Object Detection): Image classification involves assigning a label to an entire image or photograph. This task is also referred to as "object classification" and more generally as "image recognition," although this latter task may apply to a much broader set of tasks related to classifying the content of images. Some examples of image classification include:

- Labeling an x-ray as cancer or not (binary classification).
- Classifying a handwritten digit (multiclass classification).
- Assigning a name to a photograph of a face (multiclass classification).

There are many image classification tasks that involve photographs of objects. Two popular examples include the CIFAR-10 and CIFAR-100 datasets that have photographs to be classified into 10 and 100 classes respectively. The Large Scale Visual Recognition Challenge (ILSVRC) is an annual competition in which teams compete for the best performance on a range of computer vision tasks on data drawn from the ImageNet database. Many important advancements in image classification have come from papers published on or about tasks from this challenge, most notably early papers on the image classification task are : [29][25][26][30].

Beside image classification, there is a similar task but with localization that involves assigning a class label to an image and showing the location of the object in the image by a bounding box (drawing a box around the object). Few examples include: Labeling an x-ray as cancer or not and drawing a box around the cancerous region. Classifying photographs of animals and drawing a box around the animal in each scene.

A classical dataset for image classification with localization is the PASCAL Visual Object Classes datasets, or PASCAL VOC for short (e.g. VOC 2012) and Microsoft's Common Objects in Context Dataset, often referred to as MS COCO. These are datasets used in computer vision challenges over many years. The task may involve adding bounding boxes around multiple examples of the same object in the image. As such, this task may sometimes be referred to as "object detection". Object detection is a very important task in many fields like autonomous cars, robotics, wireless sensors…etc. Some examples of papers on image classification with localization are: [31][32]. Figure 1.18 shows two examples of image classification from a casual photo and a movie scene.



Fig.1.18 Example of image classification with localization from two scenes, Left : a movie scene, Right : a camera shot . [31]

Object Segmentation: Object segmentation, or semantic segmentation, is the task of object detection where a line is drawn around each object detected in the image. Unlike object detection that involves using a bounding box to identify objects, object segmentation identifies the specific pixels in the image that belong to the object. Again, the VOC 2012 and MS COCO datasets can be used for object segmentation. Some example papers are: [33][34][35][36].

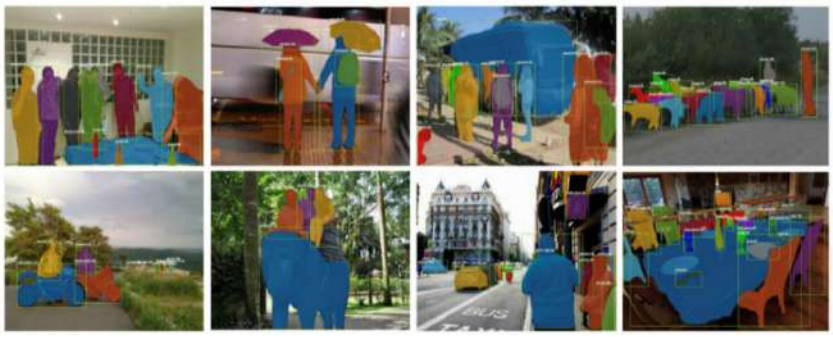


Fig.1.19 Some examples of Object Segmentation on the COCO dataset.[33]

Image reconstruction: Image reconstruction and image inpainting is the task of filling in missing or corrupt parts of an image. Examples include reconstructing old, damaged black and white photographs and movies. As shown in the figure 1.20, some applications of reconstructionm where the original image is reconstructed from a damaged one.



Fig.1.20 Examples of image inpainting.

There are other important and interesting problems that we did not cover because they are not purely computer vision tasks. Notable other examples with some related papers:

- **Image Colorization:** converting a grayscale image to a full color image.
- **Image Super-Resolution:** generating a new version of an image with a higher resolution and detail than the original image.
- **Image Synthesis:** generating targeted modifications of existing images or entirely new images.
- **Image Captioning:** Generating a textual description of an image.
- **Text to Image:** Synthesizing an image based on a textual description

1.4.2.2 Speech and Language Processing

Since Natural Language Processing is shifting from statistical methods to deep learning and neural networks methods, Both the convolutional neural network (CNN) and the recurrent neural network (RNN) have been successfully applied in this field. There are still many challenging problems to solve in natural language. Nevertheless, deep learning methods are achieving state-of-the-art results on some specific language problems.

- **Text Classification:** Given an example of text, predict a predefined class label. Potential applications include Spam filtering, Language identification, Genre classification.
- **Language Modeling:** A language model learns the probabilistic relationship between words such that new sequences of words can be generated that are statistically consistent with the source text. Potential applications include: generating new article headlines, generating new sentences, paragraphs, texts, generating suggested continuation of a sentence.
- **Speech recognition:** This task is about understanding the spoken words from a given speech. It is applied in transcription speech, creating text captions for movies of TV shows, and issuing commands to the radio while driving.

1.4.2.3 Biometrics

The applications of deep learning techniques in biometrics depends on each modality (type), such as fingerprint, face, iris, voice… etc. below we review some of these applications.

Face recognition: Face is perhaps one of the most popular biometrics (and the most researched one during the last few years). It has a wide range of applications, from security cameras in airports and government offices, to daily usage for cellphone authentication

(such as in FaceID in iPhones). There are countless number of works using deep learning for face recognition, such as DeepFace[37], DeepID [38], DeepID2 [39] and 3, FaceNet [40]., VGG-Face [27]…etc.

Fingerprint recognition: Fingerprint is arguably the most commonly used physiological biometric feature. There are several works using deep learning techniques for fingerprint recognition, namely MENet [41], FingerNet [42].

Iris recognition: Iris recognition has gained a lot of attention in recent years in different security-related fields. The deep learning works on iris recognition resulted in many state-of-the-art datasets, such as CASIA-Iris-1000, UBIRIS Dataset, IT Delhi Iris Database, ND Datasets, and MICHE database, which are used as benchmarks for DL research in iris recognition [43].

Voice recognition: Voice Recognition (also known as speaker recognition) is the task of determining a person's ID using the characteristics of one's voice. In a way, speaker recognition includes both behavioral and physiological features, such as accent and pitch respectively [43]. Some of the early attempts to apply deep learning to this task was by using a DNN-based acoustic model [44].

1.5 Conclusion

In this chapter we reviewed the concepts, methods of machine learning and deep learning and their multiple and extended applications in variety of domains along with the different techniques used. The ongoing research of ML and DL revolutionized a lot of fields of study, such as image processing, computer vision, biometrics, speech and language processing… that resulted in many crucial inventions and applications like facial recognition, autonomous cars, information retrieval … etc., all of this achievement made deep learning a dominant technology in artificial intelligence community.

Chapter 2

Biometrics and Facial Kinship Verification

2.1 Introduction

The use of biometrics for identity verification is becoming popular in many application areas such as border/port entry/exit, access control, civil identification and network security. It is well-known that no device capturing a single bio-metric trait works optimally in every application domain. The biometric modality that we are investigating throughout this work is *Kinship Verification*.

It is a common and an easy practice for humans to identify their relatives from faces. Relatives usually wonder which facial attributes does a new born baby inherit from which family member. Automatic kinship verification from faces aims at determining whether two persons have a biological kin relation or not by comparing their facial attributes [1].

In this chapter we examine the different modalities used in biometrics, with focusing on the kinship verification, its modes and challenges and datasets used in literature.

2.2 General description of biometric systems

A biometric system is a system that allows the recognition of a certain characteristic of an individual using mathematical algorithms and biometric data. There are several uses of biometric systems. There are systems that require enrollment upstream of users. Other identification systems do not require this phase [45].

Enrollment mode is a learning phase that aims to collect biometric information about who to identify. Several data acquisition campaigns can be carried out to ensure a certain robustness of the recognition system to temporal variations of the data. During this phase, the biometric characteristics of individuals are captured by a biometric sensor, and then represented in digital form (signatures), and finally stored in the database. The processing related to the enrollment has no time constraint, since it is performed “off-line.”

The verification or authentication mode is a “one-to-one” comparison, in which the system validates the identity of a person by comparing the biometric data entered with the biometric template of that person stored in the system’s database. In such a mode, the system must then answer the question related to the identity of the user. Currently the verification is carried out via a personal identification number, a user name, or a smart card.

The identification mode is a “one-to-N” comparison, in which the system recognizes an individual by matching it with one of the models in the database. The person may not be in the database. This mode consists of associating an identity with a person.

Fig.2.1 presents the general architecture of a biometric system, which consists of the following elements [45]:

- **Capture module** that represents the entry point of the biometric system and consists in acquiring the biometric data in order to extract a digital representation. This representation is used later in the following phases.
- **Signal processing module** makes it possible to optimize the processing time and the digital representation acquired in the enrollment phase in order to optimize the processing time of the verification phase and the identification.
- **Storage module** that contains the biometric templates of the system enrollees.
- **Matching module** that compares the data extracted by the extraction module with the data of the registered models and determines the degree of similarity between the two biometric data.
- **Decision module** that determines whether the similarity index returns through the matching module is sufficient to make a decision about the identity of an individual.

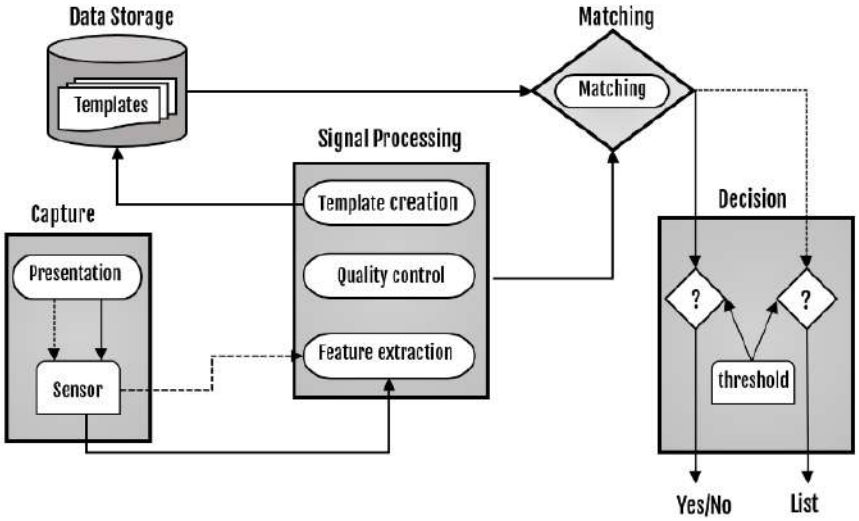


Fig.2.1 Biometric System Description.

2.3 Modalities of biometric systems

In the time being, many competitive biometric modalities and technologies are proposed. The choice of one modality with known technology is dependent on the targeted application and the desired performance. Some modalities as well as technologies, such as iris and retina, are preferred for their high reliability; others are chosen for their user-acceptability and applicability such as fingerprint and face [46]. Bellow we discuss different biometric modalities along with illustration of each modality in figure 2.2.

Fingerprint: Fingerprint is arguably the most commonly used physiological biometric feature, it is a texture pattern determined by the interleaved ridges and valleys on a fingertip characterized by positions and directions of Minutiae, which are local discontinuities caused by sudden broken ridges or merged ridges. It has Relatively high matching accuracy with a high matching speed at low cost. But still needs a dedicated sensor that needs to be touched and maintained, also these sensors can be fooled by tricked fingers and the accuracy depends on the sensor and the interoperability of the user [43].

Face: Face is perhaps one of the most popular biometrics (and the most researched one during the last few years). It has a wide range of applications, from security cameras in airports and government offices, to daily usage for cellphone authentication (such as in FaceID in iPhones). It consists of defining the location and shape of facial attributes, and detecting Eigenfaces which are a weighted combination of a number of canonical faces. It can operate on simple 2D images or 3D in static or movies images, and has a high user's acceptability and a reasonable accuracy but this latter depends on the controlled acquisition like background and light and it is sensitive to simple changes like glasses or facial appearance (challenges discussed in details section 2.6) [43].

Iris: Iris is the texture pattern in the colored part of the eye, it contains a rich set of features embedded in their texture and patterns which do not change overtime, such as rings, corona, ciliary processes, freckles, and the striated trabecular meshwork of chromatophore and fibroblast cells, which is the most prevailing under visible light. Iris recognition has gained a lot of attention in recent years in different security-related fields [43]. it has a high accuracy, difficult to be tricked even by using lenses or dead iris and less sensitive to outside influence, on the other hand the cost of implementing iris verification tend to be high [46].

Ear: Ear recognition is a more recent problem that scientists are exploring, and the volume of biometric recognition works involving ears is expected to increase in the coming years. It consists of defining the individual earmarks by capturing the appearance, structure, and morphology of the human ear [43]. It is useful because it's minimally affected by changes in the facial expressions, but it suffers from outside influences like occlusion by hair or modification by ear piercing and age [46].

Voice: Voice Recognition (also known as speaker recognition) is the task of determining a person's ID using the characteristics of one's voice. In a way, speaker recognition includes both behavioral and physiological features, such as accent and pitch respectively. Using automatic ways to perform speaker recognition dates back to 1960s when Bell Laboratories were approached by law enforcement agencies about the possibility of identifying callers who had made verbal bomb threats over the telephone. It is easy to use at low cost but suffers from low accuracy and possible replay attacks [43].

Palmprint: Palmprint is another biometric which is gaining more attention recently. In addition to minutiae features, palmprints also consist of geometry-based features, delta points, principal lines, and wrinkles. Each part of a palmprint has different features, including texture, ridges, lines and creases. An advantage of palmprints is that the creases in palmprint virtually do not change over time and are easy to extract [43]. However, sampling palmprints requires special devices with larger areas, making their collection not as easy as other biometrics such as fingerprint, iris and face [46].

Signature: Signature is considered a behavioral biometric. It is widely used in traditional and digital formats to verify the user's identity for the purposes of security, agreements in governmental legal, and commercial transactions. Therefore, being able to distinguish an authentic signature from a forged one is of utmost importance. Signature forgery can be performed as either a random forgery, where no attempt is made to make an authentic signature (e.g., merely writing the name), or a skilled forgery, where the signature is made to look like the original and is performed with the genuine signature in mind [43].

Gait: Gait recognition is a popular pattern recognition problem and attracts a lot of researchers from different communities such as computer vision, machine learning, biomedical, forensic studying and robotics. This problem has also great potential in industries such as visual surveillance, since gait can be observed from a distance without the need for the subject's cooperation. Similar to other behavioral biometrics, it is difficult, however possible, to try to imitate someone else's gait and it can be affected by footwear, nature of clothing, affliction of the legs, walking surface [46].

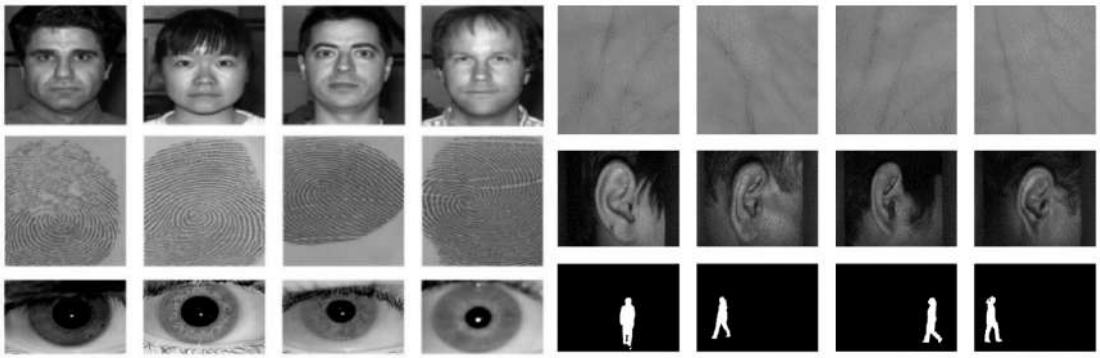


Fig.2.2 Biometric modalities: (left) from top to bottom, Face, Fingerprint, Iris. (Right) from top to bottom: Palmprint, Ear, Gait.

2.4 Applications of biometric systems

Biometrics has successfully convinced wide range of applications to be adopted not only as a fundamental component in their security architecture, but as an economical tool that can lead directly or indirectly to saving costs and reducing financial risks [47]. It had advanced quickly and significantly over the two decades. Recent researches confirmed that the market of biometrics had registered an annual growth of 19,8% between 2014 and 2019 [48]. Fingerprint modality will still dominate the. This acceleration is justified by the proliferation of the electronic services that necessitate identification, along with the rise of fraud acts and identity theft that must be fought. In addition, the adoption of the electronic documents, especially biometric passports and national ID cards, by major governments will participate to largely increase its use [46].

Biometrics is now adopted in, but not limited to:

- **Governmental applications:** biometric national identity card, biometric passport, border control, social security, e-voting ...
- **Access control:** it can be physical such as time-attendance systems and door security, or logical such as accessing remote computer resources and information systems.
- **Mobile applications:** recent mobiles are equipped with biometrics technologies that allow identifying the owner, to unlock the device, to make commercial transactions, etc. For instance, many modern smart phones and laptops are delivered with a built-in fingerprint reader along with smart software that are attended to recognition purposes.
- **Commercial applications:** most products integrate biometrics to enhance the user's experience. Access to computers, internet applications, e-commerce, banking transactions, etc.
- **Forensics applications:** forensic laboratories usually use biometrics in their criminal investigations and parenthood determination as well as to identify corpses. New researches have confirmed the possibility to determine a person's ancestor origin based on his fingerprint.
- **Military applications:** these include identification systems for use in the field, access control and monitoring applications to sensible areas, as well as large database deployments.

2.5 Kinship Verification

Any parent knows that people are always interested in whether their own child bears some resemblance to any of them, mother or father. Human interest in family ties and

connections, in common ancestry and lineage, is as natural to life as life itself. Automatic kinship verification from face images aims at verifying whether two persons have a kin relation or not by only comparing their visual facial similarities [49].

Recent advances in psychology and cognitive sciences have revealed that human face is an important cue for kin similarity measure as children usually look or behave like one of both of their parents, or both, more than any other adults, because children and their parents are biologically related and have overlapped genetics. Inspired by this finding, there have been some attempts on kinship verification via human faces, and computer vision researchers have developed several advanced computational models to verify human kinship relations via facial image analysis. But verifying similarities between face images for different people can be very challenging due to multiple factors that come from the nature of the kinship verification itself and the challenges inherited from automatic facial recognition.

2.6 Challenges of face kinship verification

Kinship verification through facial images is challenging due to the high degree of appearance variability of influences such as genetic difference, age gap and gender difference [50]. In summary, the following two factors have significant influence on problem-solving: Unique challenges which appear only in the kinship verifications systems, and common challenges which already exist in the automatic face recognition systems, both challenges are discussed in details below.

2.6.1 Unique Challenges

Humans inherit traits from their ancestors. We often hear phrases such as “Mohamed has his father’s nose” or “Ahmed has his mother’s eyes’’. A “familial trait” is defined as a trait shared by a pair of family members, e.g., father and son. Given that information, Kinship verification is different from face recognition or verification. In face recognition, the problem is to recognize the same individual by matching the probe face image with gallery images. The measure is about the “facial identity.” In kinship verification, we are dealing with different individuals, i.e., to extract similar features but from different persons that share some traits in common. We identify three major challenges in kinship verification which are considered to be unique (only for kinship verification tasks). This will help to understand the problem deeply and provide a general guide to develop a computational system [51].

(i) Trait Similarities Are Measured Across Age and gender

Familial traits (features) are measured for people at different ages (e.g., father-son), with different gender (e.g., brother–sister), or who are different in both (e.g., father – daughter) [51].

(ii) Familial Traits Are "Special" for Each Family Pair

Familial traits have "special" properties for each specific pair of family members. In a family, sons or daughters (with the same sex) may inherit traits from their parents differently [51].

(iii) Kinship Has a Stochastic Combination of Traits

As stated above, familial traits are "special" for each pair of faces. The shared traits between fathers and daughters (or other relations) can be stochastic. As a result, the kinship has to be measured by a "stochastic" (rather than a fixed) combination of familial traits. In sum, kinship verification has its own challenges. Familial traits are the building blocks for kinship measure. While some recent approaches, e.g.,[52], have processed kinship similar to an automatic face verification problem [51].

2.6.2 Common challenges

Since kinship verification is considered a subset of automatic face verification, As the challenges that exist in the latter, the appearance of kin faces is sensitive to the change of various factors, such as facial expression variations, occlusion and pose. Moreover, some common influence factors in real scene, such as lighting, blurring or low resolution, may also change the appearance of kin faces in different ways.

Pose variations

Head's movements, which can be described by the egocentric rotation angles, i.e. pitch, roll and yaw, or camera changing point of views could lead to substantial changes in face appearance and/or shape and generate intra-subject face's variations as illustrated in Fig.2.4, making automated face recognition across pose a difficult task. Since AFR is highly sensitive to pose variations, pose correction is essential and could be achieved by means of efficient techniques aiming to rotate the face and/or to align it to the image's axis as detailed in reference [53] [45].

Presence/absence of structuring elements/occlusions

The diversity in the intra-subject face's images could also be due to the absence of structuring elements or the presence of components such as beard and moustache, cap, sunglasses, etc. or occlusions of the face by background or foreground objects. Thus, face's images taken in an unconstrained environment often require effective recognition of faces with disguise or faces altered by accessories and/or by occlusions, as dealt by appropriate approaches such as texture-based algorithms [45]. This issue is illustrated in the second row of figure 2.3 bellow, where elements like hats or glasses or beard can represent a factor for occlusion.

Facial expression changes

Some more variability in face appearance could be caused by changes of facial expressions induced by varying person's emotional states which are displayed in Figure 2.3. Hence, efficiently and automatically recognizing the different facial expressions is important for both the evaluation of emotional states and the automated face recognition. In particular, human expressions are composed of macro-expressions, which could express, e.g., anger, disgust, fear, happiness, sadness or surprise, and other involuntary, rapid facial patterns, i.e. micro-expressions; all these expressions generating non-rigid motion of the face [45].



Fig.2.3 Illustration of: (**first row**) Pose Variations. (**second row**) absence presence of structuring elements, and partial occlusion. (third row) Changes in facial expression [45].

Ageing of the face

Another reason of face appearance's changes could be engendered by the ageing of the human face, and could impact on the entire face recognition process if the time between each image capture is significant [54], as illustrated in Fig.2.4. To overcome face ageing issue in AFR, methods need to take properly into account facial ageing patterns [45].

Varying illumination conditions

Large variations of illuminations could degrade the performance of AFR systems. For low levels of lighting of the background or foreground, face detection and recognition are much harder to perform, since shadows could appear on the face and/or facial patterns could be (partially) indiscernible. On the other hand, too high levels of lights could lead to over-exposure of the face and (partially) indiscernible facial patterns. Many image-processing techniques are set to place to deal with these variations, such as illumination normalization, e.g. through histogram equalization; or machine-learning methods involving the actual image global image intensity average value [45].

Image resolution and modality

Other usual factors influencing AFR performance are related to the quality and resolution of the face image and/or to the set-up and modalities of the digital equipment capturing the face. real-world situations of face image acquisition imply the use of different photographic hardware, in which could include, e.g. wide-field sensors, photometric stereo, etc. Cameras could work in the range of the visible light or use infra-red sensors, leading to multiple modalities for AFR. Hence, faces acquired in real-world conditions lead to further AFR challenges [45].



Fig.2.4 Illustration of: (first row) illumination variations. (second row) variations of the image scale and resolution. (third row) Aging.

2.6.3 Kinship modes

The most basic bonds in human relationships are those based on marriage and reproduction. Kinship refers to these basic bonds and all other relationships resulting from them. Thus, the kinship is defined as a set of relationships and relatives formed by either marriage or reproduction. It can be based on blood (consanguineal), or marriage (affinal) [55].

- **Consanguineal Kinship:** It refers to the relationships based on blood, i.e., the relationship between parents and children, and between siblings are the most basic and universal kin relations.
- **Affinal Kinship:** It refers to the relationships formed on the basis of marriage. The most basic relationship that results from marriage is that between husband and wife, son-in-law, daughter-in-law, father-in-law, mother-in-law…etc.

Any relationship between two individuals is based on the degree of closeness or distance of that relationship. This closeness or distance of any relationship depends upon how individuals are related to each other, so we find three degrees of kin relationships:

Primary Kinship: Primary kinship refers to direct relations. People who are directly related to each other are known as primary kin. There are basically eight primary kins - wife father son, father daughter mother son, wife; father son, father daughter, mother son, mother daughter; brother sister; and younger brother/sister older brother/sister.

Secondary Kinship: Secondary kinship refers to the primary kin's of primary kin. In other words, those who are directly related to primary kin (primary kin's primary kin) become one's secondary kin. There are 33 secondary kin.

Tertiary Kinship: Tertiary kinship refers to the primary kin of primary kin's primary kin or secondary kin of primary kin primary kin of secondary kin. Roughly 151 tertiary kin have been identified.

2.6.4 Kinship verification system structure.

Because of the difficulty and complexity of kinship verification, a framework must be set in place to deal with different approaches. The framework is divided into four components, Preprocessing, Features extraction, Similarity measurements and Verification

Preprocessing stage: The goal of preprocessing stage is to make it easy to measure kinship similarity by extracting kin faces from photos and reducing the influences of various variations.

Features extraction: This stage aims to extract certain caracterstics from face image to do the training with. These features can be either handcrafted-based features or learning-based features. Most existing kinship verification methods have opted for handcrafted-based features, where each face image is first divided into several blocks and then a certain characteristic is extracted from these blocks, most common descriptors are Local Binary Patterns (LBP), Local Phase Quantization (LPQ), Binarized Statistical Image Features (BSIF), Gabor wavelets… etc. The features can be typed as low-level, high, mild-level, high-level features.

Similarity measurement: The goal of this stage is to examine the similarity between features vectors that were obtained from the previous stage. There are many methods to measure similarity based on distance calculation or statistical model learning.

Verification: the final stage aims to finally verify the kin relations between data samples using various classifiers like Support Vector Machine (SVM), or K-Nearest Neighbor (KNN).

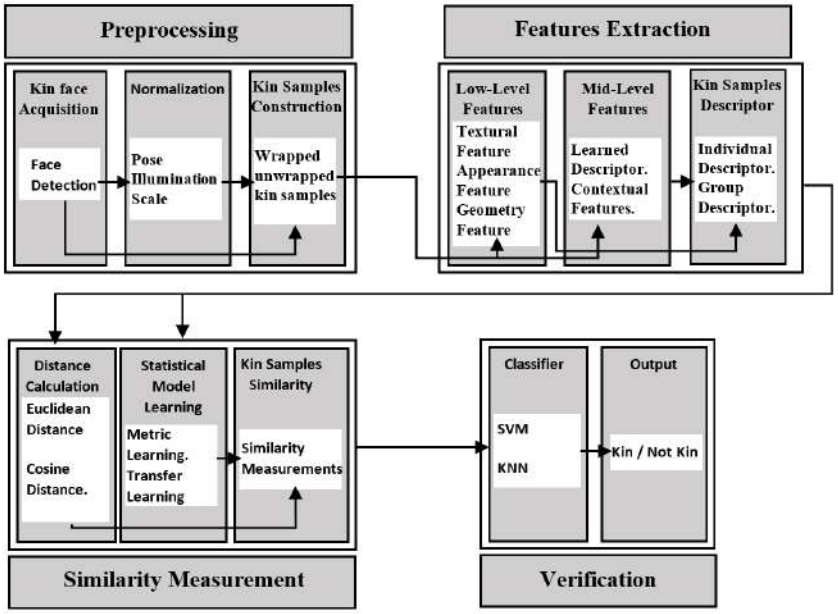


Fig.2.5 A global system framework for kinship verification.

2.7 Data bases for kinship verification

The recent trend in the study of kinship verification has brought about the availability of several databases associated by various kin relationships. All samples in these databases are mostly harvested from the Internet by means of keyword-based search engines. Fig. 2.6 illustrates some samples from several kinship databases.

(i) Cornell Kinship [52]**:** It consists of 286 images pertaining to 143 subject pairs. The facial images in this database are frontal pose and have a neutral expression.

(ii) KinFaceW-I [56]**:** This database consists of 1066 images corresponding to 533 kin pairs. It has 156 Father-Son, 134 Father-Daughter, 116 Mother-Son, and127 Mother-Daughter kin pair images.

(iii) KinFaceW-II [56]**:** This database has been created such that images belonging to the kin pair subjects are acquired from the same photograph. It consists of 1000 kin pair images with an equal number of images belonging to the four kinship relationships: Father-Son, Father-Daughter, Mother-Son, and Mother-Daughter.

(iv) UB KinFace [57]**:** This database consists of 200 groups consisting of 600 images. Each group has one image of the child and one image belonging to the corresponding parent

when they were young and when they were old. The database has 91 Father-Son, 79 Father-Daughter, 15 Mother-Son, and 21 Mother-Daughter kin pair images.

(v) WVU Kinship [58]**:** All the above kinship datasets contain only one image per kin-pair and hence are not suitable for face recognition experiments. The WVU kinship database [3] was developed as part of this research. It consists of 113pairs of individuals from Caucasian and Indian ethnicity. The dataset has four images per person, which introduces intra-class variations for a specific kin-pair. These multiple images per kin-pair include variations in pose, illumination, and occlusion.

(vi) Family 101 [59]**:** The Family 101 dataset contains 101 different families with distinct family names, including 206 nuclear families, 607 individuals, with 14,816images. The database is particularly useful for family classification where the problem is identifying which family a particular image belongs to.

(vii) TSKinFace [60]**:** The Tri-subject kinship face database consist of images be-longing to the child, mother, and father. The database consists of 513 images of Father, Mother and Son group and 502 images of Father, Mother and Daughter group.

(viii) FIW [61]**:** The families in the wild dataset consist of 10,700 images belonging to 1000 families.

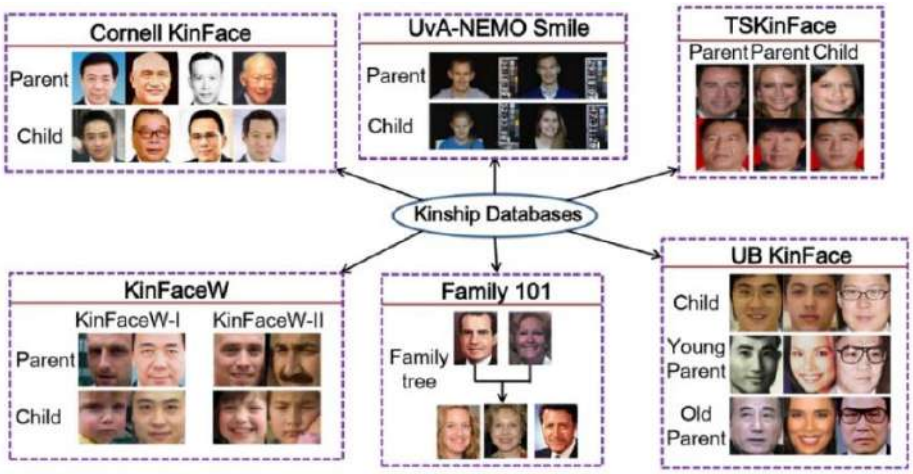


Fig.2.6 Sample images from six different face kinship databases: Cornell KinFace, UvA-NEMO Smile, TSKinFace, KinFaceW, Family 101 and UB KinFace.

Table 2.1 Summary of kinship databases in the literature.

	Dataset	No. of Subjects	No. of Samples	Relations
Image	CornellKin	300	300	4
	UB KinFace	400	600	4
	KFW-I	1,066	1,066	4
	KFW-II	2,000	2,000	4
	WVUKin	226	904	7
	Family101	607	14,816	4
	TSKinFace	2,589	2,000	4
	FIW	10,700	31,000	11
Video	UvA-Nemo	152	514	7
	KIVI	503	503	7

2.8 Conclusion

In this chapter, we have discussed the field of biometrics along with its different modalities and their applications. We also investigated the kinship verification problem, its challenges and the benchmark databased used in its literature. The kinship verification is a new field of study involves both biometrics and computer vision, but it achieves quite satisfactory performance. Many ongoing researches are constantly developing new and efficient methods to tack the kinship verification challenges and improve the verification accuracy by implementing powerful methods for features extraction, dimensionality reduction, fusion and classification.

Chapter 3

Kinship Verification Based VGG16 & Gabor Wavelet Features

3.1 Introduction

Computer vision researchers around the world are always trying to optimize the performance of kinship verification systems with trying and developing different approaches and methods. Most of the methods used in the kinship verification literature are based on metric learning and hand engineered face descriptors, for example, LBP [62], LPQ [63], BSIF [64], HoG [65]... etc. Few of them are using deep features constructed by deep learning methods.

In this chapter we are going to investigate and compare the performance of kinship verification system using two different types of features: Deep features using VGG16 CNN model, and Shallow features using Gabor Wavelet or Filters. We are going to run a set of experiments with different settings and parameters on two kinship datasets to estimate the best performing features for kinship verification system using a new dimensionality reduction method called Tensor Cross-View Quadratic Discriminant Analysis [66].

3.2 Environment used and work outline

The experiments were carried out using MATLAB R2019a on a PC with an Intel (R) i5-7200U CPU at 2.5 GHz with 8 GB of RAM. We used MatConvNet [28] environment for CNNs. The figure 3.1 represents our architecture for kinship matching system. It consists of three major parts: Preprocessing, Feature Extraction, Dimensionality Reduction and comparison. Each part is discussed in details in the next sections.

The preprocessing stage consists of setting the images to be ready for use in the verification system by performing some editing. The feature extraction stage consists of extracting a certain number of features from face images by certain methods explained later in this chapter. The dimensionality reduction stage is about reducing the dimensions of the collected data (i.e. features) to be used without complications inside the learning algorithm. The training stage is responsible for learning the discriminative features for each pair of

images in the datasets. The testing stage estimates the performance of the training stage by testing face image pairs randomly via k-fold cross validation technique. The Matching stage determines if the pair of face images are kin related or not by comparing their feature vectors using cosine similarity metric. The score fusion Stage takes the scores obtained from each of the features extracted Shallow and Deep and fuse them. Each stage is explained in details in the following sections.

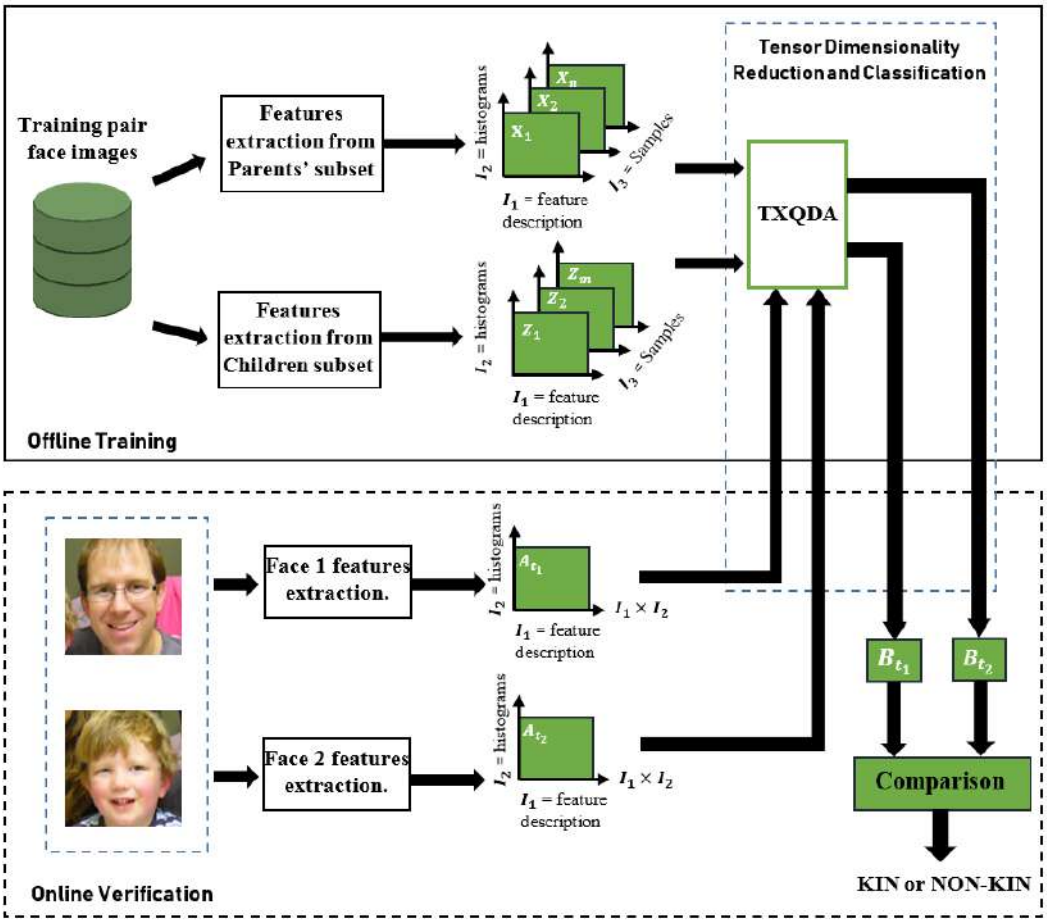


Fig.3.1 The proposed kinship verification system.

3.3 Preprocessing stage

The preprocessing stage is responsible for preparing the facial images to the training stage. The processed images for *CornellKin* Face dataset are resized to 200×200 pixels and cropped to identify the face region between pixel number 55 to 180 along the X axis and from pixel number 43 to 157 along the Y axis. For *TSK KinFace* dataset we leave images as they are 64×64 pixels. Then, the cropped faces are processed with the method used by *Xiaoyang Tan* and *Bill Triggs* paper, the processing incorporates a series of steps chosen to counter the effects of illumination variations, local shadowing and highlights, while still preserving the essential elements of visual appearance for use in recognition, the steps include : Gamma correction, Difference of Gaussian (DoG) Filtering, Equalization

and Masking. Finally, the images are supplied to the local feature representation and extraction stage to extract the main features from the processed faces [20].

3.4 Features Extraction

To describe face images with a strong discrimination power, we extract two kind of features, deep features using VGG16 ConvNet, and shallow features using Gabor Wavelet. The fusion at score level procedure is used to get advantage of the two kind of features.

3.4.1 Features extraction with VGG16 and VGGface model

The VGG-Face model, named later, was described by *Omkar Parkhi* in the 2015 paper titled 'Deep Face Recognition'[27], the contribution of this paper was a description of how to develop a very large training dataset, required to train modern-convolutional-neural-network-based face recognition systems, to compete with the large datasets used to train models at Facebook and Google. This dataset has about 2.6 million images, including 2,622 identities and up to 1,000 images per identity [67]. In the end *Parkhi et al.* trained the VGG16 model on this dataset, and then created the pretrained model known as VGGface, which we'll use in our work, this operation is names Transfer Learning.

We run the VGGface ConvNet model which was trained on VGGface dataset, over each image of our kinship verification datasets, the result of each execution is a set of unique features for each input image that the network has learned during the execution. As a next step we extract four fully connected layers of the network, these layers contain the features (i.e. weights) learned from our kinship dataset's images. Each face image results in a set of unique features that can be considered as a biometric identity for each person in each image.

The layers extracted are as follows:

The **relu6** is the sixth Rectified Linear Unit (**ReLu**) activation function layer which contains 4096D weights. The **fc7** layer is the seventh fully connected layer which contains 4096D weights. The **relu7** is the seventh Rectified Linear Unit (ReLu) activation function layer which contains 4096D weights. The **fc8** layer is the eighth fully connected layer which contains 4096D weights. all weights resulted from the four layers are concatenated in one vector of size $(4096 \times 4) \times 1 = 16232$. The process is fully described in the diagram presented in Fig.3.2. The choice of the four fully connected layers is justified empirically through parametric studies from various kinship verification papers [68].

Note: No preprocessing was done to face images before feeding them to the VGG-Face model.

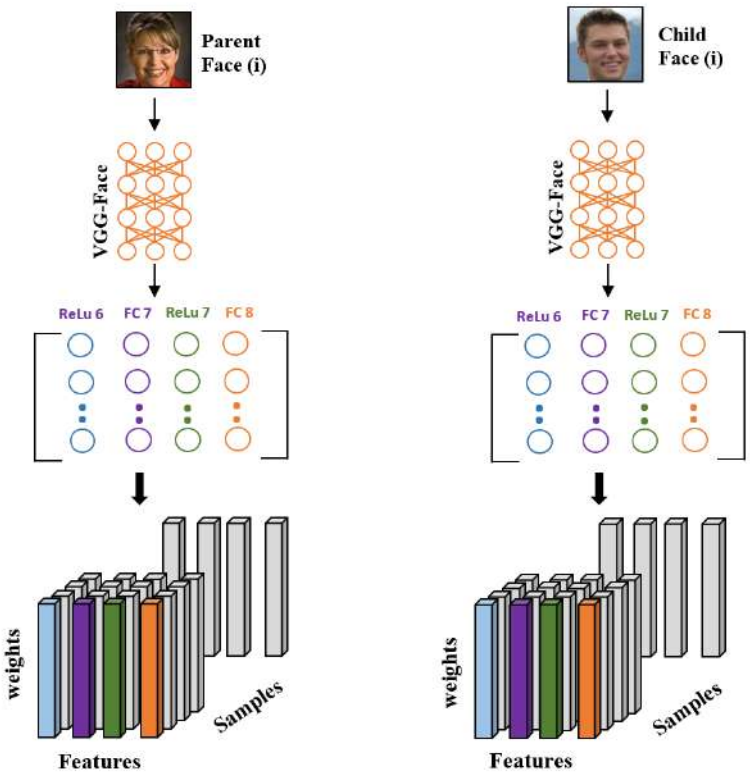


Fig.3.2 Feature extraction using VGG-Face.

3.4.2. Feature Extraction using Gabor Wavelet

3.4.2.1. Theoretical Aspects of Gabor filters

In image processing, a Gabor filters [69], named after Dennis Gabor, are a set of linear filters widely used for the analysis of the texture motivated by their spatial location nature, orientation, selectivity, and the frequency characteristic. Gabor filters are seen as the precursors of wavelets, where each filter captures energy at a particular frequency and for a specific direction. Their diffusion is motivated by mathematical properties and neurophysiological evidence. In 1946 Gabor showed that the specificity of a signal, simultaneously in time and frequency is fundamentally limited by a lower limit given by the product of its band length and duration. Subsequently, Gabor's functions were generalized in two-dimensional space to model the profile of receptive fields of simple cells of the primary visual cortex (also known as striated cortex)[7][2].

[7] *Psychovisual redundancy* studies indicate that the human visual system processes images at different scales. In the early stages of vision, the brain performs a sort of analysis in different spatial frequencies, and consequently, the visual cortex is composed of different cells that correspond to different frequencies and orientations. It has also been observed that the responses of these cells are similar to those of the Gabor functions. This multiscale process, which successfully takes place in the human vision for texture perception, has motivated the development of texture analysis methods that mimic the mechanisms of human vision.

In the 2D spatial domain, the canonical elementary function of Gabor $h(x, y)$ is a complex harmonic function (i.e., composed of the sine and cosine functions) modulated by a Gaussian oriented function $g(x_0, y_0)$. The 2D oriented Gaussian function is given by

$$g(x_0, y_0) = \frac{1}{2\pi\gamma\sigma} \exp\left[-\frac{\left(\frac{x_0}{\gamma}\right)^2 + y_0^2}{2\sigma^2}\right] \quad 3.1$$

Two-dimensional Gabor functions were proposed by *J.G Daugman* in [69] to model the spatial summation properties (of the receptive fields) of simple cells in the visual cortex. They are widely used in image processing, computer vision, neuroscience and psychophysics [2]. The parametrization is as follows:

$$G(x, y; \lambda, \theta, \varphi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi \frac{x'}{\lambda} + \varphi\right)\right)$$
$$x' = x\cos\theta + y\sin\theta \quad 3.2$$
$$y' = -x\sin\theta + y\cos\theta$$

Where parameters are explained and demonstrated as follows:

Wavelength ($\boldsymbol{\lambda}$): This is the wavelength of the cosine factor of the Gabor kernel and herewith the preferred wavelength of this filter. Its value is specified in pixels. Valid values are real numbers equal to or greater than 2.

Orientations ($\boldsymbol{\theta}$): This parameter specifies the orientation of the normal to the parallel stripes of a Gabor function. Its value is specified in degrees. Valid values are real numbers between 0 and 360.

Phase offset(s) ($\boldsymbol{\varphi}$): The phase offset φ in the argument of the cosine factor of the Gabor function is specified in degrees. Valid values are real numbers between -180 and 180. The values 0 and 180 correspond to center-symmetric 'center-on' and 'center-off' functions, respectively, while -90 and 90 correspond to anti-symmetric functions. All other cases correspond to asymmetric functions.

Aspect ratio ($\boldsymbol{\gamma}$)**:** This parameter, called more precisely the spatial aspect ratio, specifies the ellipticity of the support of the Gabor function. For $\gamma = 1$, the support is circular. For $\gamma < 1$ the support is elongated in orientation of the parallel stripes of the function. Default value is $\gamma = 0.5$.

Bandwidth ($\boldsymbol{\beta}$): The half-response spatial frequency bandwidth b (in octaves) of a Gabor filter is related to the ratio σ / λ, where σ and λ are the standard deviation of the Gaussian factor of the Gabor function and the preferred wavelength. The bandwidth value must be specified as a real positive number. Default is 1, σ and λ are connected as follows: $\sigma = 0.56\,\lambda$. the support of the Gabor function and the number of visible parallel excitatory and inhibitory stripe zones.

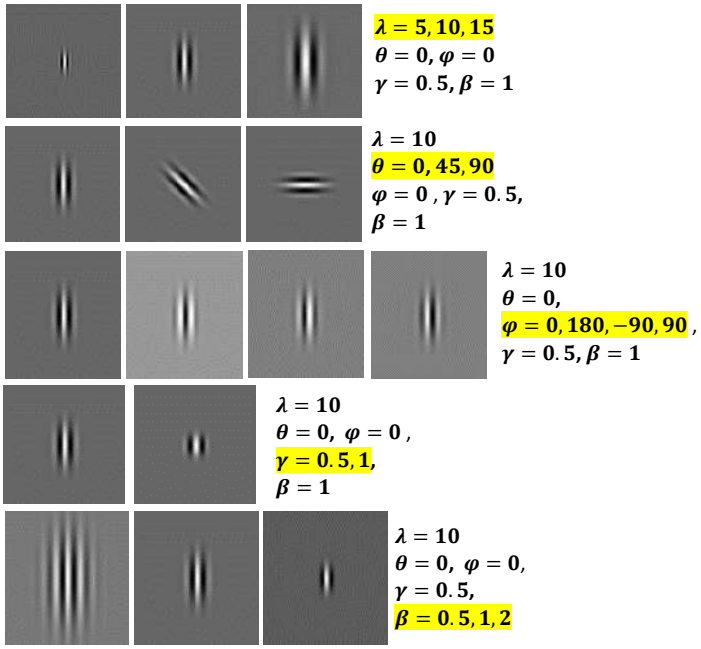


Fig.3.3 Gabor kernel visualized for different settings.

We use Gabor filter with different scales by changing the settings the following settings repeatedly (Empirically chosen as the best performing settings):

Orientations ($\boldsymbol{\theta}$): 45°, 67.5°, 360°, 112.5°.

Wavelength ($\boldsymbol{\lambda}$): 16, 22.63.

Gaussian's variance: we set it equal to wavelength.

Phase offsets ($\boldsymbol{\varphi}$): = 0, 90°.

The process goes as follows:

a. Each face image from our datasets is subdivided into a certain number of blocs $M \times N$, where $M = \frac{Y}{P_2}$ and $N = \frac{X}{P_1}$, P_2 and P_1 denote the number of pixels corresponding to the number of blocks along X and Y respectively.

b. The resulted matrix is arranged into another matrix of size $(P_1 \times P_2) \times (M \times N)$

c. Histograms of 256 bins are applied to aggregate the features from each bloc to get a matrix of size $(256) \times (M \times N)$.

d. The resulted feature matrix is reshaped to a set of vectors where each vector represents one scale of the features, we get $(Scales) \times (256) \times (M \times N))$. Scales are set to: $4, 6, 8, 10$.

e. Finally, these features are concatenated (flattened) into a one-dimensional vector of length $Scales \times 256 \times M \times N$.

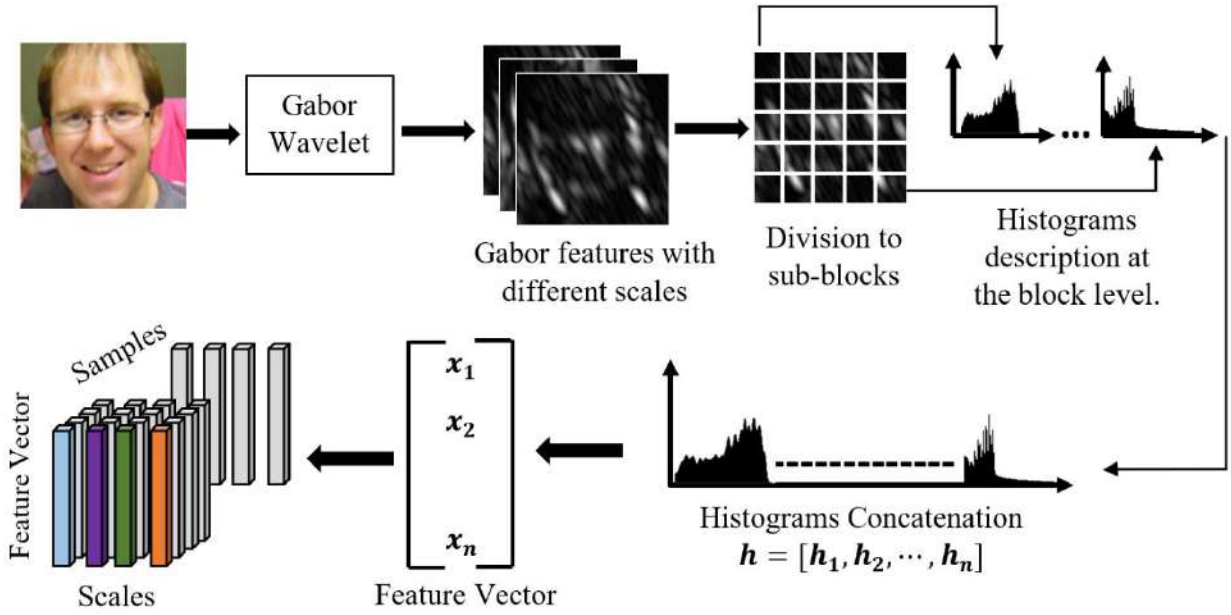


Fig.3.4 Feature extraction using Hist-Gabor wavelet.

3.4.3. Tensor Design

The training 3rd order tensors $\boldsymbol{X}$, $\boldsymbol{Z} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ are constructed using the histograms of different feature descriptors extracted from the training face images. The three modes of the tensors $\boldsymbol{X}$ and $\boldsymbol{Z}$ are defined as follows: I_1 corresponds to the feature descriptors extracted at different scales, I_2 represents the histograms, and I_3 face samples in the database as shown in Fig.3.4 and Fig.3.3.

3.4.4. Dimensionality reduction stage

The feature vector is constructed by concatenating the extracted features from VGG16 and Gabor Wavelet techniques. The main problem of this vector is its high dimensionality. Therefore, it is required to project the feature vector into a lower space that contains only discriminant information. The dimensionality reduction stage stimulates the efficiency of the calculation of the recognition system and avoids technical problems, such as the curse of dimensionality. Dimensionality reduction can circumvent this problem by

reducing the number of features in the data set before the training process. This can also reduce the computation time, and the resulting classifiers take less space to store. Models with small number of variables are often easier for domain experts to interpret. Dimensionality reduction is also invaluable as a visualization tool, where the high dimensional data set is transformed into two or three dimensions for display purposes. This can give the system designer additional insight into the problem at hand. The main drawback of dimensionality reduction is the possibility of information loss. When done poorly, dimensionality reduction can discard useful instead of irrelevant information. No matter what subsequent processing is to be performed, there is no way to recover this information loss.[70]

There are many dimensionality reduction methods used in data science for different types of applications namely Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Side Information Discriminant Analysis (SIDA), TeSNE, Multilinear PCA (MPCA), Multilinear SIDA (MSIDA), Cross-View Quadratic Discriminant Analysis XQDA… etc. In our work we use a new dimensionality reduction technique developed for kinship verification called Tensor Cross-view Quadratic Discriminant Analysis (TXQDA) proposed by *Laiadi et al.* in [66] which was developed over the XQDA method. this new technique had achieved better results than the previous ones in dimensionality reduction for kinship verification datasets. The algorithm for Tensor Cross-view Quadratic Discriminant Analysis method is proceeds as follows.

Algorithm 1 Tensor Cross-view Quadratic Discriminant Analysis (TXQDA)

Input:

- $\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times n}$
- $\boldsymbol{Z} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times m}$
- $Iteration_{max}$
- $I'_1 \times I'_2 \times \ldots \times I'_N$

Output:

- $W_k = W^{iter} \in \mathbb{R}^{I_k \times I'_k}, k = 1, \ldots, N$

Algorithm:

1. **Initialization:** $W_1^0 = I_{I_1}, W_2^0 = I_{I_2}, W_N^0 = I_{I_N}$,
2. **For** $iter: 1\ to\ Iteration_{max}$
   (a) **For** $K = 1\ to\ N$
   - $\boldsymbol{U} = \boldsymbol{X} \times_1 W_1^{iter-1} \cdots \times_{k-1} W_{k-1}^{iter-1} \times_{k+1} W_{k+1}^{iter-1} \cdots \times_N W_N^{iter-1}$

- $U^k \Leftarrow_k \boldsymbol{U}$
- $\boldsymbol{Y} = \boldsymbol{Z} \times_1 W_1^{iter-1} \cdots \times_{k-1} W_{k-1}^{iter-1} \times_{k+1} W_{k+1}^{iter-1} \cdots \times_N W_N^{iter-1}$
- $Y^k \Leftarrow_k Y$
- $n_I \Sigma_I = \sum_{p=1}^{\Pi_{0 \neq k} I_0} n_I \Sigma_I^p, n_I \Sigma_I^p = \widetilde{U}^{k,p} (\widetilde{U}^{k,p})^T + n\tilde{Y}^{k,p} (\tilde{Y}^{k,p})^T - S^{k,p} (R^{k,p})^T - R^{k,p} (S^{k,p})^T$
- $n_E \Sigma_E = \sum_{p=1}^{\Pi_{0 \neq k} I_0} n_E \Sigma_E^p, n_E \Sigma_E^p = mU^{k,p} (U^{k,p})^T + nY^{k,p} (Y^{k,p})^T - \boldsymbol{s}^{k,p} (\boldsymbol{r}^{k,p})^T - \boldsymbol{r}^{k,p} (\boldsymbol{s}^{k,p})^T - n_I \Sigma_I^p$
- **Compute** $\Sigma_E^k W_k^{iter} = \Lambda_k \Sigma_I^k W_k^{iter}$ **obtain** , W_k^{iter}.

(b) **If** $iter > 2$ and $\left\| W_k^{iter} - W_k^{iter-1} \right\| < I_k I_k \epsilon, k = 1, \cdots, N$, break;

3. Sort the $\acute{I_N}$ eigenvectors $W_k^{iter} \in \mathbb{R}^{I_k \times \acute{I_k}}$ according to Λ_k in decreasing order, $k = 1, \cdots, N$.

3.4.5. Train and test Phase

We use Cross-Validation or K-fold Cross Validation method, the model is trained and tested in the same time. Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model. It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split. The process of testing is determined by the chosen dataset's protocol. Figure 3.5 illustrates the cross validation process using 5 folds, hence it's named 5-fold cross validation.

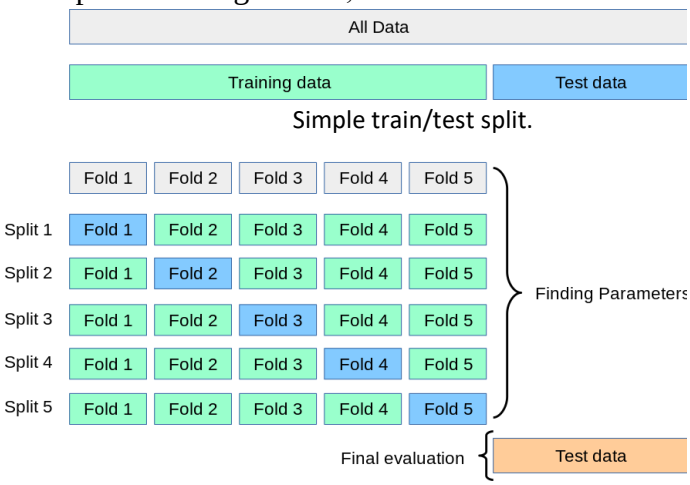


Fig.3.5 K-fold Cross Validation, with K = 5.

3.4.6. Scoring and Fusion phase

To compare between two faces pair, we used the reduced features projected through the TXQDA space which are concatenated to form one feature vector. Then, we applied cosine similarity [71] for each pair test of the two face images, so a match score is done. After discriminant analysis methods, the use of cosine similarity distance has an advantage which comes from its connection to the Bayes decision rule [72]. Cosine similarity between two vectors $\boldsymbol{b}_{t_1}$ and $\boldsymbol{b}_{t_2}$ is defined as the following:

$$\cos(\boldsymbol{b}_{t_1}, \boldsymbol{b}_{t_2}) = \frac{\boldsymbol{b}_{t_1}^T \cdot \boldsymbol{b}_{t_2}}{\|\boldsymbol{b}_{t_1}\| \cdot \|\boldsymbol{b}_{t_2}\|} \quad 3.3$$

Where, $\|\cdot\|$ is the Euclidean norm. A high value of the produced score means a high probability that $\boldsymbol{b}_{t_1}$ and $\boldsymbol{b}_{t_2}$ belong to the same family as shown in figure 3.6.

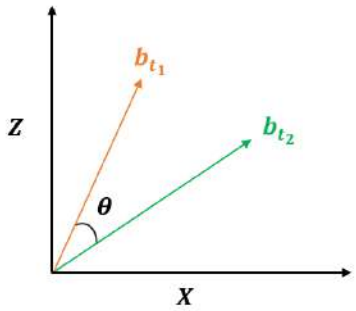


Fig.3.6 Cosine Similarity.

Fusion refers to the combination of two or more scores to obtain a classifier with a higher level of performance, hence it's known by classifier based fusion at score level [73], the classifier used in our work is logistic regression (LR) to fuse the features of the best performing settings from the used methods at the score level. Figure 3.7 illustrates the process.

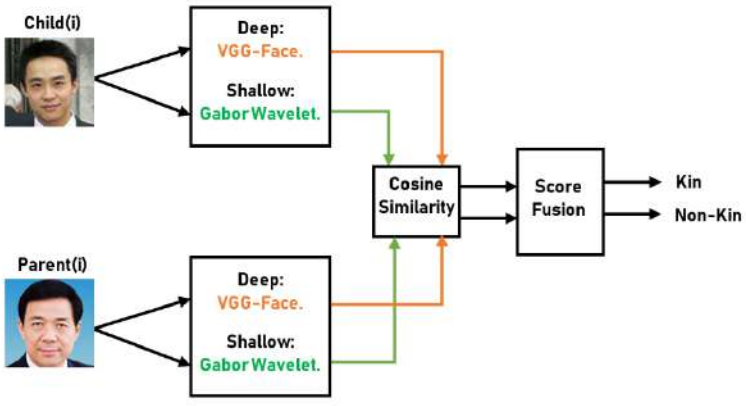


Fig.3.7 Fusion at the score level of VGG-Face and Gabor wavelet features.

3.5 Implementation and Results

We perform a number of experiments to evaluate the two proposed methods.

a. For VGG16 there are only 4 scales of features to be extracted and trained with.

b. For Hist-Gabor:

1. The number of bloc histograms is chosen to: 4,8,12,16,20,24, 28.

2. After experimenting the results with the scale 6, the best performing number of blocs is fixed and the number of scales is changed iteratively to: 4, 6, 8, 10.

3. With best performing number of scales, the number of features projection is changed iteratively from 100 to 200 with the step of 10.

4. The best setting in Gabor wavelet is fixed and fused with the VGG16 score.

3.5.2. Cornell KinFace Dataset

Here we present the results of our experiments on Cornell KinFace dataset, in which Table 3.1 illustrated the mean accuracy of our Hist-Gabor approach using 6 scales with variations of number of blocs and variation of scales with the best performing number of blocs. Table 3.2 illustrates the fusion at score level between the best previous performing results for Hist-Gabor and VGG16. On the other hand, the ROC (Receiver Operating Characteristics) is illustrated in figure 3.8.

Table 3.1 The mean accuracy (%) of kinship verification for TXQDA using our Hist-Gabor. on Cornell Kin Face database defining 6 scales, (table down): defining 20 blocs.

Features	Number of blocs	Number of features	Mean accuracy
Hist-Gabor scales = 6	4	6144	58.21
	8	12288	82.56
	12	18432	90.87
	16	24576	92.37
	20	**30720**	**93.07**
	24	36864	92.40
	28	43008	92.03

Table 3.2 the mean accuracy (%) of kinship verification for TXQDA using Hist-Gabor. on Cornell Kin Face database defining 20 blocs.

Features	Number of scales	Number of features	Mean accuracy (%)
Hist-Gabor Blocs = 20	4	20480	92.74
	6	**30720**	**93.07**
	8	40960	92.40
	10	51200	92.77

Table 3.3 Fusion of best performing features for our Hist-Gabor with 6 Scales/20blocks/200 features and VGG16 with 150 features on TSK kin Face database.

Features	Number of features	Number of features after projection	Mean accuracy (%)
Hist-Gabor (1)	30720	200	93.07
VGG16 (2)	16384	150	91.33
Fusion (1+2)	-	-	**93.76**

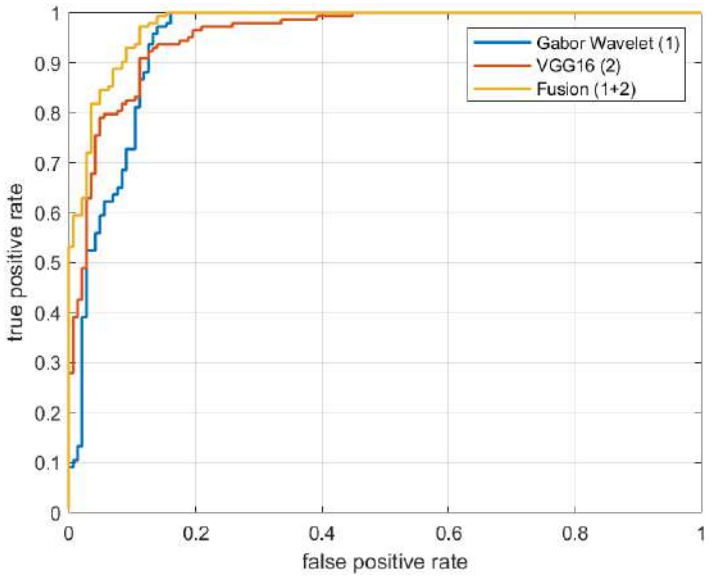


Fig.3.8 ROC curves of best performing methods on Cornell KinFace database.

3.5.3. TSK KinFace Dataset

In this section we present the results of our experiments on TSK KinFace dataset, in which Table 3.4 illustrates the mean accuracy of our Hist-Gabor approach using 6 scales with variations of number of blocs. Table 3.5 illustrates the variation of scales with the best performing number of blocks and table 3.6 illustrates the fusion at score level between the best previous performing results for Hist-Gabor and VGG16. The relations are as follows FS : Father-Son, FD : Father-Daughter, MS : Mother-Son, MD : Mother-Daughter.

Table 3.4 The mean accuracy (%) of kinship verification for TXQDA using our Hist-Gabor with 6 scales on TSK kin Face database.

Features	Number of blocs	Number of features	FS	FD	MS	MD	Mean (%)
Hist-Gabor	4	6144	54.75	53.37	53.37	55.35	54.21
	8	12288	60.79	59.01	60.69	59.60	60.02
	6	9216	51.19	50.89	51.78	51.88	51.43
	12	18432	75.84	76.44	77.62	79.90	77.45
	16	24576	74.95	76.73	75.54	75.05	75.57
	20	30720	84.75	84.65	83.76	86.73	84.97
	24	36864	87.43	89.31	88.42	89.41	88.64
	28	**43008**	**89.01**	**90.20**	**89.11**	**90.59**	**89.73**

Table 3.5 The mean accuracy (%) of kinship verification for TXQDA using our Hist-Gabor with 28 blocks and different scales. on TSK KinFace database.

Features	Number of scales	Number of features	FS	FD	MS	MD	Mean (%)
Hist-Gabor	4	172032	88.12	90.20	89.01	90.59	89.48
	6	**258048**	**89.01**	**90.20**	89.11	**90.59**	**89.73**
	8	344064	88.51	89.90	**89.50**	90.40	89.58
	10	430080	88.02	89.80	88.91	89.90	89.16

Table 3.6 Fusion of best performing features for our Hist-Gabor with 6 Scales/28blocks/200 features and VGG16 with 100 features on TSK KinFace database.

Features	FS	FD	MS	MD	Mean
Hist-Gabor (1)	89.01	90.20	89.11	90.59	89.73
VGG 16 (2)	79.60	80.50	82.97	82.18	81.31
Fusion (1) + (2)	90.79	91.98	91.68	92.18	**91.66**

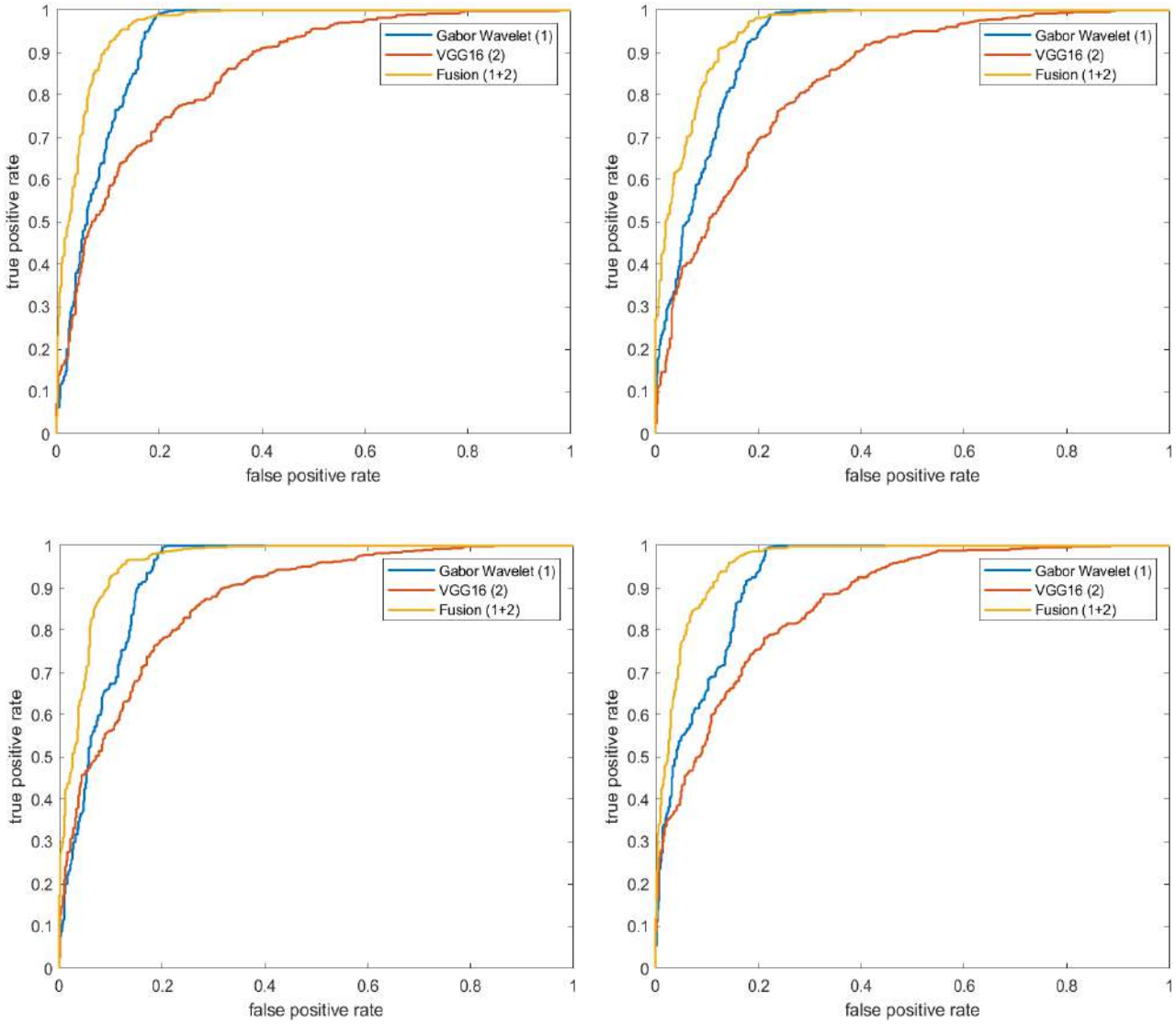


Fig.3.9 ROC curves of best performing methods TSKinFace database obtained on (a) F-S set, (b)F-D set, (c) M-S set and (d) M-D set, respectively.

3.6 Discussion

The experiments that we conducted using the proposed approach TXQDA based LR fusion of two types of features (Hist-Gabor and VGG-Face) on two benchmark datasets Cornell KinFace and TS-KinFace, illustrated the following:

1. The outperformance of VGG-Face CNN model features:

In order to extract deep features from our kinship datasets, we used a pretrained CNN model called VGG-Face, this model was trained for face recognition tasks, on a 2.6 m samples dataset, the experimental results on Cornell KinFace and TSKinFace datasets show a quite satisfactory results, in fact, it outperforms some other approaches in different

learning methods, this is due to the nature of the model, since it was trained for face recognition in a large dataset, the weights of this model are tuned in favor of facial recognition which is useful for kinship verification. But the performance of VGG-Face can still be improved by combining it with conventional metric leaning or other deep metric learning methods.

2. The advantage of using TXQDA for dimensionality reduction:

We apply TXQDA as a dimensionality reduction technique to make use of its advantages for preserving the data structure, the robust discriminative power and reducing the computational cost. Thus, TXQDA exceeds the state-of-the-art dimensionality reduction techniques deployed for kinship verification such as MPCA, MSIDA, DIEDA…etc. and provides better results. Further details about the TXQDA are presented in its original paper here [66].

3. The power of LR fusion at the score level for two types of features:

We apply fusion at the score level on two scores resulting from the two types of features (VGG-face, Hist-Gabor) using LR method. It is remarkable that LR fusion process shows a high performance with 93.76%, 91.66% in with Cornell KinFace and TSKinFace, respectively (see tables 3.3 and 3.6). For better illustration of the performances of different features, the Receiving Operating Characteristic (ROC) Curves of different methods are plotted in Fig.3.8 and Fig.3.9 with Cornell KinFace and TSKinFace respectively. We can observe from these figures that LR fusion can offer a significant increase in accuracy.

3.6.1. Comparison with state-of-the-art methods

The best performance of the proposed method TXQDA based score fusion of Hist-Gabor and VGG-face is compared with recent techniques in Table 3.6 for TSKinFace database, Table 3.7 for Cornell database respectively. The related works are cited according to the type of learning used, metric, multi-metric and multilinear learning approaches. The comparison shows that our proposed method outperforms the recent state of the art on the two databases, Cornell KinFace and TSKinFace.

Comprehensive experimental results on the used kinship datasets confirmed that our approach outperformed (in terms of mean accuracy) the best methods of metric learning, multi-metric learning, and multilinear subspace learning approaches. Table 3.8 shows the increase in average accuracy provided by our method compared to the best methods of each category.

Table 3.7 Comparison of realized performance against existing methods in TSKinFace dataset.

Approach Type	Author	Year	Mean Accuracy (%)
Metric Learning	Lu et al. [74]	2017	84.15
	Liang et al. [75]	2018	90.47
	Laiadi et al. [76]	2018	83.63
Multi Metric Learning	Laiadi et al. [68]	2019	88.59
Multilinear Subspace Learning	Bessaoudi et al. [77]	2019	85.18
	Laiadi et al. [66]	2019	90.32
	Proposed	**2020**	**91.66**

Table 3.8 Comparison of realized performance against existing methods in Cornell KinFace dataset.

Approach Type	Author	Year	Mean Accuracy (%)
Metric Learning	Lu et al. [56]	2014	69.50
	Zhou et al. [78]	2018	81.40
	Laiadi et al. [76]	2018	77.60
Multi Metric Learning	Lu et al. [56]	2014	71.60
	Yan et al. [79]	2014	73.50
	Mahpod et Keller. [80]	2017	76.60
Multilinear Subspace Learning	Bessaoudi et al. [77]	2019	86.87
	Laiadi et al. [66]	2019	93.04
	Proposed	**2020**	**93.76**

Table 3.9 Increase in mean accuracy of proposed approach against state-of-the art methods.

Approach	Database	Best Method	Increase in Mean Accuracy (%)
Metric Learning	Cornell KinFace	Zhou et al. [78]	12.36
	TSKinFace	Liang et al. [75]	1.19
Multi Metric Learning	Cornell KinFace	Mahpod et Keller. [80]	17.16
	TSKinFace	Laiadi et al. [68]	3.07
Multilinear Subspace Learning	Cornell KinFace	Laiadi et al. [66]	0.72
	TSKinFace	Laiadi et al. [66]	1.34

3.7 Conclusion

In this chapter we have presented our new approach for kinship verification, named Hist-Gabor for features extraction based on Gabor wavelet. This approach consists of dividing sample images into a number of blocks, extracting histogram features from each block and then concatenating these features into one feature vector representing each sample image (i.e. person) from the dataset. In order to improve the performance of our

kinship verification systems in terms of accuracy we used the LR fusion at score level of our proposed method Hist-Gabor with the features extracted using a convolutional neural network pretrained model called VGG-Face. The Experiments showed that our proposed approach outperforms the state -of-the-art methods in kinship verification literature.

General Conclusion

Kinship verification is a new and emerging field in computer vision and biometrics. Many potential applications, such as creation of family trees, family album organization, image annotation, finding missing children and forensics, are targeted by kinship verification. But, Learning and extracting the face similarity between family members is challenging due to multiple factors such as stochasticity of traits inherited between family members and the common challenges that occur in automatic face recognition and the fact that there exists people who share face similarities with each other although they are not from the same family, which presents more challenges to the kinship verification systems. Many researches involved in this field trying to develop the best approaches to tackle the kinship verification problem from different sides, features extraction methods, dimensionality reduction methods and fusion at different levels between a variant of features.

Our work aimed for a better approach in terms of accuracy for the kinship verification system, by introducing a new method based on Gabor wavelet, called Hist-Gabor, and by using the advantage of fusion at the score level with the features extracted by a convolutional neural network pretrained model - for face verification - called VGG-Face.

Our contributions can be summarized as bellow:

- A review of the literature of machine learning and deep learning and their applications in both computer vision and biometrics. And the literature of kinship verification along with challenges and related datasets.
- The introduction of new feature extraction method based on Gabor Wavelet called Hist-Gabor, consists of extracting the histogram features for each block of the sample image, and the fusion at the score level of these features with the ones extracted using VGG-Face CNN pretrained model.
- The use of a powerful dimensionality reduction technique called Tensor Cross View Quadratic Discriminant Analysis (TXQDA) that projects the data into new subspace to provide a more robust discrimination power to the features that will be used later in comparison.
- The fusion at score level between two types of features, the proposed Hist-Gabor and VGG-Face to improve the classification accuracy with a significant margin.

The perspectives of this work revolve around three main points:

- Use of other and newer pretrained CNN models for features extraction such as GoogleNet [26], ResNet100 [30], Light CNN [81], VGG-Face2 [67] along with the proposed Hist-Gabor approach.
- The use of combination between three kind of features, Deep features extracted by a pretrained CNN model, and local descriptors like LBP, BSIF or their variants along with our proposed Hist-Gabor.
- CNN models perform poorly on significantly small datasets, thus the need for bigger datasets for kinship verification is crucial to achieve a significant improvement in terms of accuracy using a raw CNN model.

Appendices

1. ROC curve

The ROC curve (ROC stands for "receiver operating characteristic," the term comes from radar engineering) is a commonly used method to assess the performance of classification models. ROC curves use a combination the true positive rate (the proportion of positive examples predicted correctly, defined exactly as **recall**) and false positive rate (the proportion of negative examples predicted incorrectly) to build up a summary picture of the classification performance. The true positive rate and the false positive rate are respectively defined as

$$TPR = \frac{TP}{TP + FN}$$
$$FPR = \frac{FP}{FP + TN}$$

ROC curves can only be used to assess classifiers that return some confidence score (or a probability) of prediction. For example, logistic regression, neural networks, and decision trees (and ensemble models based on decision trees) can be assessed using ROC curves.

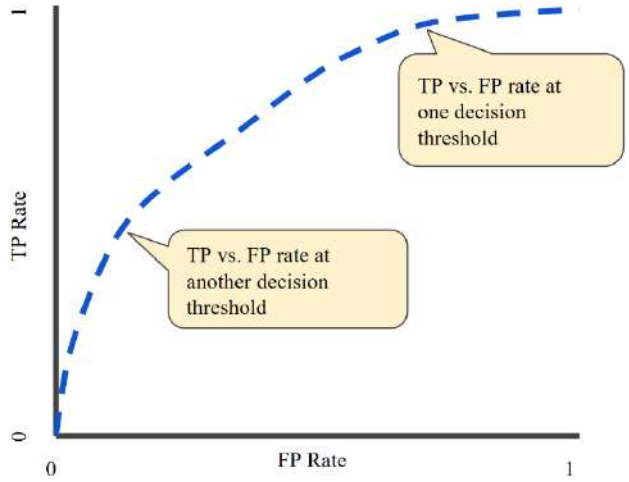


Fig.1 ROC Curve.

2. Fusion in biometrics

Known also as Fusion at the confidence level; Fusion at the measurement level; Match score fusion, in score-level fusion the match scores output by multiple biometric matchers are consolidated in order to render a decision about the identity of an individual.

Typically, this consolidation procedure results in the generation of a single scalar score which is subsequently used by the biometric system. Fusion at this level is the most commonly discussed approach in the biometric literature primarily due to the ease of accessing and processing match scores Fusion methods at this level are classified into three categories: density-based schemes, transformation-based schemes, and classifier-based schemes. Classifier-based fusion schemes assume the availability of a large representative number of genuine and impostor (match and non-match) scores during the training phase of the classifier when its parameters are computed. In the biometric literature, several classifiers have been used to consolidate the match scores like Support Vector Machines SVM, and Logistic Regression LR [73].

3. Logistic Regression and Score Fusion

In order to apply fusion at the score level, first we have to normalize the multiple scores and then fuse them. The similarity scores, for deep features, and the loglikelihood scores, for shallow scores, were normalized to produce probabilities so that the scores will be in the range [0, 1], this process is performed by using logistic regression algorithm by taking y_i as the input and generating probabilities, P_i, as an output:

$$P_i = (1 + \exp\,(ay_i + b))^{-1}$$

where a is a scaling factor and b is a bias. The final combined score is obtained by taking the product of the two scores, $P_{F\ eature1}$ for first feature and $P_{F\ eature2}$ for second feature. Although the sum rule could have also been used. Both rules result in similar generalization performance [82].

Logistic regression algorithms are used to study how a set of predictor variables X is related to a dichotomous response variable Y. The predictors may describe such quantities as treatment assignment, dosage, risk factors, and calendar time. For convenience we define the response to be $Y = 0$ or 1, with $Y = 1$ denoting the occurrence of the event of interest. Letting X denote the vector of predictors $\{X_1, X_2, \ldots, X_k\}$, a first attempt at modeling the response might use the ordinary linear regression model $E\{Y|X\} = X\beta$ since the expectation of a binary variable Y is Prob$\{Y = 1\}$. However, such a model by definition cannot fit the data over the whole range of the predictors since a purely linear model $E\{Y|X\} = Prob\{Y = 1|X\} = X\beta$ can allow $Prob\{Y = 1\}$to exceed 1 or fall below 0. The statistical model that is generally preferred for the analysis of binary responses is instead the binary logistic regression model, stated in terms of the probability that Y=1givenX, the values of the predictors:

$$Prob\{Y = 1|X\} = [1 + \exp(-X\beta)]^{-1}$$

As before, $X\beta$ stands for $B_0 + B_1X_1 + B_2X_2 + \ldots + B_kX_k$. The regression parameters β are estimated by the method of maximum likelihood.

The function

$$P = (1 + \exp(-X))^{-1}$$

is called the logistic function. This function is plotted bellow. For x varying from -4 to $+4$. This function has an unlimited range for x while P is restricted to range from 0 to 1.

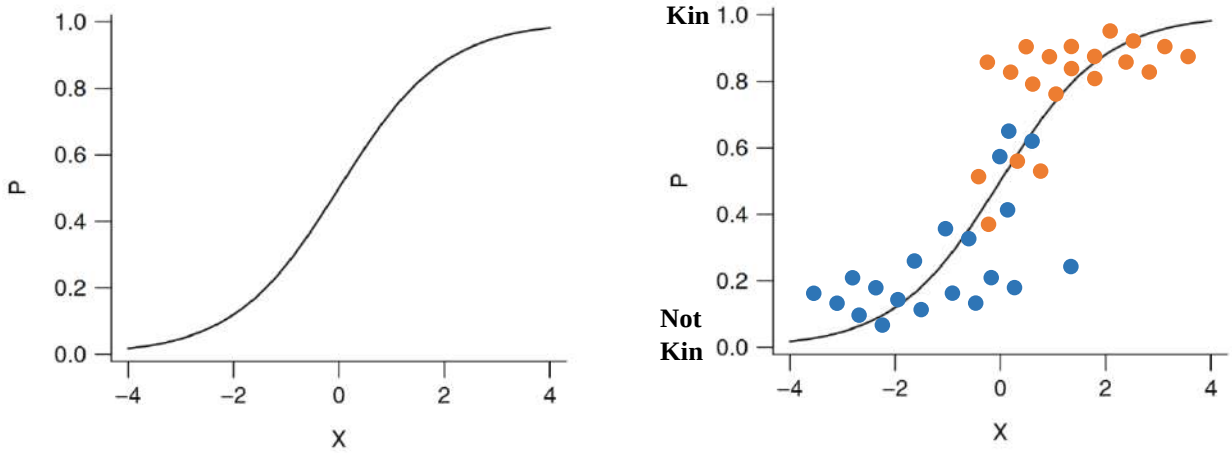


Fig.2 Left: the logistic function, Right: face samples (data) distribution separated by the logistic function.

References

[1] H. Yan and J. Lu, *Facial Kinship Verification. A Machine Learning Approach*. 2017.

[2] A. Distante and C. Distante, *Handbook of Image Processing and Computer Vision*, vol. 2. 2020.

[3] Ahmed Fawzy Gad, *Practical Computer Vision Applications Using Deep Learning with CNNs: With Detailed Examples in Python Using TensorFlow and Kivy*. 2018.

[4] J. VijiPriya, J. Ashok, and S. Suppiah, "A Review on Significance of Sub Fields in Artificial Intelligence," *Int. J. Latest Trends Eng. Technol.*, vol. 6, no. 3, pp. 542–548, 2016.

[5] A. Burkov, "The Hundred-Page Machine Learning Book."

[6] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*, vol. 9781107057. Cambridge University Press, 2013.

[7] I. Vasilev, D. Slater, G. Spacagna, P. Roelants, and V. Zocca, *Python Deep Learning: Exploring Deep Learning Techniques and Neural Network Architectures with PyTorch, Keras and TensorFlow*. 2018.

[8] A. Distante and C. Distante, *Handbook of Image Processing and Computer Vision*, vol. 1. 2020.

[9] M. A. Arbib, *The Handbook of Brain Theory and Neural Networks*, 2nd ed. MIT Press, 2003.

[10] A. Distante and C. Distante, *Handbook of Image Processing and Computer Vision*, vol. 3. 2020.

[11] D. Learning, *Charu C. Aggarwal A Textbook 123 Neural Networks and Deep Learning*. .

[12] H. Habibi Aghdam and E. Jahani Heravi, *Guide to Convolutional Neural Networks*. 2017.

[13] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, *A Guide to Convolutional Neural Networks for Computer Vision*, vol. 8, no. 1. Morgan & Claypool Publishers, 2018.

[14] "Concepts — ML Glossary documentation." [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/nn_concepts.html. [Accessed: 15-Aug-2020].

[15] S. L. Brunton and J. N. Kutz, "Data Driven Science & Engineering - Machine Learning, Dynamical Systems, and Control." p. 572, 2017.

[16] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, no. 2010, pp. 8609–8613, 2013, doi: 10.1109/ICASSP.2013.6639346.

[17] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright, "Adaptive SubgradientMethods for

Online Learning and Stochastic Optimization," *Proc. IEEE Conf. Decis. Control*, vol. 12, pp. 5442–5444, 2012, doi: 10.1109/CDC.2012.6426698.

[18] R. F. Lyon and R. F. Lyon, "Neural Networks for Machine Learning," *Hum. Mach. Hear.*, pp. 419–440, 2017, doi: 10.1017/9781139051699.031.

[19] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

[20] A. S. Alexander Amini, "MIT 6.S191: Introduction to Deep Learning." MIT, 2020.

[21] Y. Pang and J. Cao, *Deep Learning in Object Detection and Recognition*. 2019.

[22] "Conv Nets: A Modular Perspective - colah's blog." [Online]. Available: http://colah.github.io/posts/2014-07-Conv-Nets-Modular/. [Accessed: 18-Feb-2020].

[23] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," pp. 1–31, 2016.

[24] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[26] C. Szegedy *et al.*, "Going deeper with convolutions," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07–12–June, pp. 1–9, 2015, doi: 10.1109/CVPR.2015.7298594.

[27] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," no. Section 3, p. 41.1-41.12, 2015, doi: 10.5244/c.29.41.

[28] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," *MM 2015 - Proc. 2015 ACM Multimed. Conf.*, pp. 689–692, 2015, doi: 10.1145/2733373.2807412.

[29] A. Krizhevsky and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," pp. 1–9.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016–Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[31] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection."

[32] S. Mukhopadhyay and S. Mukhopadhyay, *Deep Learning and Neural Networks*. 2018.

[33] M. R-cnn, P. Doll, and R. Girshick, "Mask R-CNN."

[34] P. Arbel, R. Girshick, and J. Malik, "Hypercolumns for Object Segmentation and Fine-grained Localization."

[35] C. Han, Y. Duan, X. Tao, and J. Lu, "Dense Convolutional Networks for Semantic Segmentation," *IEEE Access*, vol. 7, pp. 43369–43382, 2019, doi: 10.1109/ACCESS.2019.2908685.

[36] C. V Jul, B. Hariharan, and P. Arbel, "Simultaneous Detection and Segmentation," pp. 1–16.

[37] T. Yaniv, Y. Ming, R. Marc'Aurelio, and W. Lior, "Deepface: Closing the gap to human-levelperformance in face verification.," *Proc. theIEEE Conf. Comput. Vis. pattern Recognit.*, pp. 1702–1708, 2014.

[38] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1891–1898, 2014, doi: 10.1109/CVPR.2014.244.

[39] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," *Adv. Neural Inf. Process. Syst.*, vol. 3, no. January, pp. 1988–1996, 2014.

[40] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07–12–June, pp. 815–823, 2015, doi: 10.1109/CVPR.2015.7298682.

[41] L. N. Darlow and B. Rosman, "Fingerprint minutiae extraction using deep learning," *IEEE Int. Jt. Conf. Biometrics, IJCB 2017*, vol. 2018–Janua, pp. 22–30, 2018, doi: 10.1109/BTAS.2017.8272678.

[42] Y. Tang, F. Gao, J. Feng, and Y. Liu, "FingerNet: An unified deep network for fingerprint minutiae extraction," *IEEE Int. Jt. Conf. Biometrics, IJCB 2017*, vol. 2018–Janua, pp. 108–116, 2018, doi: 10.1109/BTAS.2017.8272688.

[43] S. Minaee, A. Abdolrashidi, H. Su, M. Bennamoun, and D. Zhang, "Biometric Recognition Using Deep Learning: A Survey," 2019.

[44] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol., no., pp. 1695–1699, 2014, doi: 10.1109/ICASSP.2014.6853887.

[45] V. I. B. R. A. T. N. A. L. Spectroscopies *et al.*, "Pattern Recognition, Analysis and Applications," in *intechopen*, Dr. Mahali., vol. i, no. tourism, S. Ramakrishnan, Ed. 2016, p. 13.

[46] F. Belhadj, "Biometric system for identification and authentication Foudil Belhadj To cite this version : HAL Id : tel-01456829 Biometric system for identification and authentication," 2017.

[47] E. Mazzon, *Review: Biometrics: Identity Verification in a Networked World*, vol. 45, no. 3. 2003.

[48] House of Commons Science and Technology Commitee, "Current and future uses of

biometric data and technologies," pp. 1–86, 2015.

[49] X. Wu, E. Boutellaa, X. Feng, and A. Hadid, "Kinship Verification From Faces : Methods , Databases and Challenges," 2016.

[50] N. Kohli, "Automatic Kinship Verification in Unconstrained Faces using Deep Learning Automatic Kinship Verification in Unconstrained," 2019.

[51] G. Guodong and W. Xiaolong, "Kinship Measurement on Salient Facial Features Guodong," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 10, pp. 2799–2806, 2012, doi: 10.1109/TIM.2012.2192355.

[52] R. Fang, K. D. Tang, N. Snavely, and T. Chen, "Towards computational models of kinship verification," *Proc. - Int. Conf. Image Process. ICIP*, pp. 1577–1580, 2010, doi: 10.1109/ICIP.2010.5652590.

[53] S. Nash, M. Rhodes, and J. I. Olszewska, "iFR: Interactively pose corrected face recognition," *BIOSIGNALS 2016 - 9th Int. Conf. Bio-Inspired Syst. Signal Process. Proceedings; Part 9th Int. Jt. Conf. Biomed. Eng. Syst. Technol. BIOSTEC 2016*, vol. 4, no. Biostec, pp. 106–112, 2016, doi: 10.5220/0005857801060112.

[54] L. Liu, C. Xiong, H. Zhang, Z. Niu, M. Wang, and S. Yan, "Deep Aging Face Verification With Large Gaps," *IEEE Trans. Multimed.*, vol. 18, no. 1, pp. 64–75, 2016, doi: 10.1109/TMM.2015.2500730.

[55] P. Mondal, "Kinship: Brief Essay on Kinship." [Online]. Available: https://www.yourarticlelibrary.com/sociology/kinship-brief-essay-on-kinship-892-words/6205. [Accessed: 04-Sep-2020].

[56] H. Yan, X. Zhou, and Y. Ge, "Neighborhood repulsed correlation metric learning for kinship verification," *2015 Vis. Commun. Image Process. VCIP 2015*, vol. 36, no. 2, pp. 331–345, 2016, doi: 10.1109/VCIP.2015.7457930.

[57] S. Xia, M. Shao, and Y. Fu, "Kinship verification through transfer learning," *IJCAI Int. Jt. Conf. Artif. Intell.*, pp. 2539–2544, 2011, doi: 10.5591/978-1-57735-516-8/IJCAI11-422.

[58] N. Kohli, M. Vatsa, R. Singh, A. Noore, and A. Majumdar, "Hierarchical representation learning for kinship verification," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 289–302, 2017, doi: 10.1109/TIP.2016.2609811.

[59] R. Fang, A. C. Gallagher, T. Chen, and A. Loui, "Kinship classification by modeling facial feature heredity," *2013 IEEE Int. Conf. Image Process. ICIP 2013 - Proc.*, pp. 2983–2987, 2013, doi: 10.1109/ICIP.2013.6738614.

[60] X. Qin, X. Tan, and S. Chen, "Tri-Subject Kinship Verification: Understanding the Core of A Family," *IEEE Trans. Multimed.*, vol. 17, no. 10, pp. 1855–1867, 2015, doi: 10.1109/TMM.2015.2461462.

[61] J. P. Robinson, M. Shao, Y. Wu, and Y. Fu, "Families in the Wild (FIW): Large-scale kinship image database and benchmarks," *MM 2016 - Proc. 2016 ACM Multimed. Conf.*, pp. 242–246, 2016, doi: 10.1145/2964284.2967219.

[62] T. Ojala, M. Pietikäinen, and D. Harwood, “Performance evaluation of texture measures with classification based on Kullback discrimination of distributions,” *Proc. - Int. Conf. Pattern Recognit.*, vol. 3, pp. 582–585, 1994, doi: 10.1109/ICPR.1994.576366.

[63] V. Ojansivu and J. Heikkilä, “Blur insensitive texture classification using local phase quantization,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5099 LNCS, pp. 236–243, 2008, doi: 10.1007/978-3-540-69905-7_27.

[64] J. Kannala and E. Rahtu, “BSIF: Binarized statistical image features,” *Proc. - Int. Conf. Pattern Recognit.*, pp. 1363–1366, 2012.

[65] N. Dalal, B. Triggs, and D. Europe, “Histograms of Oriented Gradients for Human Detection,” 2005.

[66] O. Laiadi, A. Ouamane, A. Benakcha, A. Taleb-Ahmed, and A. Hadid, “Tensor cross-view quadratic discriminant analysis for kinship verification in the wild,” *Neurocomputing*, vol. 377, pp. 286–300, 2020, doi: 10.1016/j.neucom.2019.10.055.

[67] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, “VGGFace2: A dataset for recognising faces across pose and age,” *Proc. - 13th IEEE Int. Conf. Autom. Face Gesture Recognition, FG 2018*, pp. 67–74, 2018, doi: 10.1109/FG.2018.00020.

[68] O. Laiadi, A. Ouamane, A. Benakcha, A. Taleb-Ahmed, and A. Hadid, “Learning multi-view deep and shallow features through new discriminative subspace for bi-subject and tri-subject kinship verification,” *Appl. Intell.*, vol. 49, no. 11, pp. 3894–3908, 2019, doi: 10.1007/s10489-019-01489-2.

[69] J.G. Daugman, “Uncertainty relations for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters,” *J. Opt. Soc. Am. A*, vol. 2, p. 1160–1169., 1985.

[70] H. C. Law, “Clustering , Dimensionality Reduction , and Side Information,” Michingan State University, 2006.

[71] H. V Nguyen and L. Bai, “Cosine Similarity Metric Learning for Face Verification,” in *Computer Vision -- ACCV 2010*, 2011, pp. 709–720.

[72] C. Liu, “Discriminant analysis and similarity measure,” *Pattern Recognit.*, vol. 47, no. 1, pp. 359–367, 2014, doi: 10.1016/j.patcog.2013.06.023.

[73] S. Z. Li, *Encyclopedia of Biometrics*, Second Edi. Springer Reference, 2015.

[74] J. Lu, J. Hu, and Y. P. Tan, “Discriminative Deep Metric Learning for Face and Kinship Verification,” *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4269–4282, 2017, doi: 10.1109/TIP.2017.2717505.

[75] J. Liang, Q. Hu, C. Dang, and W. Zuo, “Weighted Graph Embedding-Based Metric Learning for Kinship Verification,” *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1149–1162, 2019, doi: 10.1109/TIP.2018.2875346.

[76] O. Laiadi, A. Ouamane, E. Boutellaa, A. Benakcha, A. Taleb-Ahmed, and A. Hadid,

"Kinship verification from face images in discriminative subspaces of color components," *Multimed. Tools Appl.*, vol. 78, no. 12, pp. 16465–16487, 2019, doi: 10.1007/s11042-018-7027-9.

[77] M. Bessaoudi, A. Ouamane, M. Belahcene, A. Chouchane, E. Boutellaa, and S. Bourennane, "Multilinear Side-Information based Discriminant Analysis for face and kinship verification in the wild," *Neurocomputing*, vol. 329, pp. 267–278, 2019, doi: 10.1016/j.neucom.2018.09.051.

[78] X. Zhou, K. Jin, M. Xu, and G. Guo, "Learning deep compact similarity metric for kinship verification from face images," *Inf. Fusion*, vol. 48, pp. 84–94, 2019, doi: 10.1016/j.inffus.2018.07.011.

[79] H. Yan, J. Lu, W. Deng, and X. Zhou, "Discriminative multimetric learning for kinship verification," *IEEE Trans. Inf. Forensics Secur.*, vol. 9, no. 7, pp. 1169–1178, 2014, doi: 10.1109/TIFS.2014.2327757.

[80] S. Mahpod and Y. Keller, "Kinship verification using multiview hybrid distance learning," *Comput. Vis. Image Underst.*, vol. 167, no. September, pp. 28–36, 2018, doi: 10.1016/j.cviu.2017.12.003.

[81] X. Wu, R. He, Z. Sun, and T. Tan, "A light CNN for deep face representation with noisy labels," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 11, pp. 2884–2896, 2018, doi: 10.1109/TIFS.2018.2833032.

[82] Harrell FEJr, *Regression modeling strategies: withapplications to linear models, logistic and ordinal regression, andsurvival analysis.* Springer Berlin Heidelberg, 2015.