”Democratic and Popular Republic of Algeria
MINISTRY OF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH”

Bejaia University

Faculty of Exact Sciences

Computer Science Department

ACADEMIC MASTER

in Computer Science

Option: *AI*

Theme

Kinship Verification Based on Various Models (PATCH, STATIC, SVM, CNN, LPQ, BSIF,LBP)

Realized By:

FODIL Ayoub

BOULAININE Kenza

Defended on July 4, 2024, before the committee composed of:

President	Dr M.BOUCHEBBAH	M.C.B - Bejaia University
Examiner	Dr M.MOKETFI	M.C.B - Bejaia University
supervisor	Dr M.KHAMMARI	M.C.A - Bejaia University
Co-supervisor	M BELABBACI	Doctorant - Bejaia University

Promotion : 2023/2024

Gratitude

We thank **Allah** Almighty, who helped us complete this work and scientific research, giving us strength, health, and determination.

We also extend our sincere thanks and appreciation to the supervising professor, ***KHAMMARI Mohammed***, and assistant supervisor, ***BELEABACI El Ouanas***, for all the advice, guidance, and valuable information they provided, which contributed to facilitating the path to this respectable work.

Our gratitude also extends to the members of the correcting committee, Professor ***Meketfi*** and ***Bouchebah***, for their valuable feedback and suggestions.

Dedication

With this beautiful moment, I dedicate this work especially to the soul of the deceased " **Younis Hamidi** " , may God make him one of the birds of Paradise.

Mom you made me a Men ,

- **Dad** you keep my life.

My brother ***Fodil isamil***. My sister and second mother ***Fodil zineb***.

I dedicate this work for all who are mentioned there

Fodil Ayoub.

Dedication

To my parents, for their love and unwavering support.

To my husband, for your invaluable support and patience throughout this journey. To my friends,

for their presence and encouragement. May

this work reflect their trust and encouragement.

Kenza Boulainine.

Contents

Gratitude **1**

Table of Contents **i**

Table of figures **iv**

List of tables **vii**

list of abbreviations **viii**

General Introduction **1**

1 Kinship Verification **3**

1.1 Introduction 3

1.2 The Kinship 4

1.2.1 What is it ? 4

1.2.2 Importance 5

1.3 Kinship in Computer Vision 5

1.4 Kinship, its applications, and the motivations 5

1.5 Automatic recognition of kinship 8

1.6 Kinship verification and face verification 9

1.6.1 Face Verification 9

1.6.2 Kinship verification 9

1.7 Kinship verification system structure 9

1.7.1 Preprocessing 11

1.7.2 The features extraction phase 11
1.7.3 Similarity measurement 12
1.7.4 Verification Phase 12
1.8 Problems and challenges of kinship verification 13
1.8.1 Pose variations 13
1.8.2 Presence or absence of structuring elements and occlusions 14
1.8.3 Facial expression changes 14
1.8.4 Ageing of the face: 15
1.8.5 Varying illumination conditions 15
1.8.6 Image resolution and modality 16
1.9 Conclusion 17

2 Kinship Verification Methods 18
2.1 Introduction 18
2.2 Face detection 19
2.2.1 YOLO 19
2.2.2 MTCNN 20
2.3 Features Extraction: 23
2.3.1 Shallow Features: 23
2.3.2 Feature Extraction Using Deep Learning 26
2.4 Classification Algorithms 29
2.4.1 The k-NN algorithm 29
2.4.2 A decision tree 31
2.4.3 Support Vector Machine (SVM) 32
2.4.4 pretrained CNN models for the Kinship System 34
2.5 Decision: 35
2.6 Related work 37
2.7 Conclusion 39

3 Proposed Methods 40
3.1 Introduction 40
3.2 Proposed Method 40
3.2.1 Face detcetion 42

3.2.2 Feature Extraction 42
3.3 Conclusion 50

4 Experimentation, results and discussion 51
4.1 Introduction 51
4.2 Development Environment 52
4.2.1 Hardware Environement 52
4.2.2 Devlopement Tools Presentation 52
4.3 Databases for kinship verification 58
4.3.1 Database Used 61
4.4 Evaluation 62
4.4.1 Methodology for Evaluation 62
4.5 Experiment 64
4.5.1 Experiment Setting: 66
4.6 Test and Result 67
4.7 Conclusion 69

general conclusion 70

Bibliographie 72

List of Figures

1.1 Illustration of photo organization 6

1.2 Illustration of missing children search 7

1.3 Illustration of surveillance 7

1.4 Illustration of genealogical research 8

1.5 General Kinship verification system structure 10

1.6 Features Extraction Methode[1] 12

1.7 Illustration of pose variations 13

1.8 Illustration of Presence/absence of structuring elements/occlusions 14

1.9 Illustration of Facial Expression Changes 15

1.10 Illustration of pose variations 15

1.11 Illustration of illumination variation 16

1.12 Illustration of variations of the image scale and resolution[2]. 16

2.1 YOLO architecture)[3] 20

2.2 image pyramid exemple[1] 21

2.3 P-Net [4] 22

2.4 R-Net[4] 22

2.5 O-Net [4] 23

2.6 texture extraction using Local Binary Pattern (LBP)[?, 5] 24

2.7 Sample images of Binarized statistical image features (BSIF) descriptor.With: a) Input Image, b-c-d-e)result of BSIF with filterof 3x3, 7x7 and 11x11 dimension respectively. [5] 25

2.8 Standard test image (rice.png) and the LOOP output[6]. 26

2.9 VGG Neural Network Architecture 28

2.10 Residual Network Architecture [7] 29

2.11 An example of k-NN classification.. 30

2.12 An example of a decision tree predicting ”yes” or ”no” for playing outside based on the weather[8]. 32

2.13 SVM[8]. 34

2.14 Common Pretrained CNN Models for Kinship Verification 35

2.15 The first facial kinship verification study[9] 37

3.1 Proposed kinship verification system. 41

3.2 Proposed kinship verification system. 42

3.3 image patched 43

3.4 vector of static features for the image 44

3.5 The process of feature extraction from two images and distance measurement 45

3.6 KIN And NO_KIN List 47

3.7 subtract two color images (RGB). 48

4.1 The figure shows the F-S relationship database based on KinFaceW-II. . . 52

4.2 Python Logo 53

4.3 google collab logo 54

4.4 jupyter logo 55

4.5 OpenCV 56

4.6 NumPy logo 56

4.7 Matplotlib logo 57

4.8 TensorFlow ogo 57

4.9 Keras Logo 57

4.10 Google Drivelogo 58

4.11 Sample images from six different data sources: Cornell KinFace, TSKinFace, KinFaceW, Family 101, and UBKinFace. 61

4.12 Different types of relationships. 62

4.13 Holdout method 63

4.14 K-fold cross validation[2] 64

4.15 Learning curve for 04 relationship for SVM 68
4.16 Learning curve for FS relationship for SVM 69

List of Tables

1.1 Comparison between Kinship Verification and Face Verification [3] 10

4.1 Summary of kinship datasets in the literature [4]. 60

4.2 Comparison of different approaches on the KinFaceW-I and KinFaceW-II databases 67

lists of abbreviations

1. **CNN**: Convolutional Neural Networks.
2. **SVM**: Support Vector Machine.
3. **ML**: Machine Learning.
4. **TL**: Transfer Learning.
5. **AI**: Artificial Intelligence.
6. **F**: Father.
7. **S**: Son.
8. **KV**: Kinship Verification.
9. **DL**: Deep Learning.
10. **KIN**: Kinship.
11. **FS**: Father Son.
12. **FD**: Father Daughter.
13. **MS**: Mother Son.
14. **MD**: Mother Daughter.
15. **LSTM**: Long Short-Term Memory.
16. **ResNet**: Residual Networks.

17. **VGG16**: Visual Geometry Group 16.
18. **MTCNN**: Multi-task Cascaded Convolutional Networks.
19. **YOLO**: You Only Look Once.
20. **LBP**: Local Binary Patterns.
21. **BSIF**: Binary Spatial Independent Features.
22. **LOOP**: Local Phase Quantization.
23. **KNN**: k-Nearest Neighbors.
24. **HOG**: Histogram of Oriented Gradients.
25. **NRML**: Neighborhood Repulsed Metric Learning.
26. **NRCML**: Neighborhood Constrained Metric Learning.
27. **DKV**: Direct Kinship Verification.
28. **ESL**: Ethnicity Similarity Learning.
29. **PML**: Pairwise Metric Learning.

General Introduction

Technological progress, driven by advancements in information processing techniques and the ongoing revolution in microprocessor systems, has led to the creation of artificial solutions aimed at addressing various security challenges. These challenges include embezzlement, forgery, unauthorized access, and the alteration or loss of confidential information. Among the most critical innovations are biometric verification and identification systems, which play a vital role in mitigating these risks[5].

Authentication and identification are two key processes within modern security systems. Authentication verifies the identity of a person or machine, granting access to controlled environments or information systems. It ensures that only authorized individuals or devices can enter secure areas or access sensitive data. Identification, on the other hand, determines an individual's identity efficiently and accurately, allowing for proper recognition and tracking. Both processes are crucial for maintaining security and preventing unauthorized access in areas such as banking, government systems, and digital platforms[6][5].

Biometric systems utilize unique biological characteristics (such as fingerprints, facial features, or voice patterns) to provide fast and reliable identification or verification. Compared to traditional methods like passwords, biometric systems offer enhanced security since they rely on traits that are difficult to forge or steal. The European Union's AI Act also emphasizes the importance of regulating biometric systems, especially in areas like emotion recognition and categorization.

Currently, biometric authentication is widely used in sectors requiring secure access, such as banking, government offices, and electronic commerce. These systems are also critical in forensic investigations and immigration services, where they provide reliable methods of identifying individuals and controlling access. The integration of biometric systems into these fields significantly enhances security measures and ensures the reliability of both access and identification processes.

In recent years, kinship verification has emerged as a vital area within biometric identification. This field focuses on determining familial relationships based on facial features, with applications ranging from family reunification and social media to forensic investigations. In this study, we aim to implement kinship verification systems using advanced machine learning (ML) and deep learning (DL) techniques. Specifically, we employ Convolutional Neural Networks (CNNs) for feature extraction and Support Vector Machines (SVM) for classification. Additionally, we explore other methods and fusion of algorithms to improve feature extraction and overall accuracy. By integrating multiple approaches, this research seeks to develop more robust and accurate kinship verification systems for real-world applications.

Chapter 1

Kinship Verification

1.1 Introduction

This chapter delves into the procedural aspects of facial kinship verification, beginning with foundational theories, definitions, and essential terminology necessary to elucidate this field. It provides an overview of automatic kinship verification methods, detailing the technological advancements and methodologies involved. Additionally, the chapter explores the design of facial kinship verification systems, highlighting their structure, components, and operational frameworks.

Kinship verification (KV) from facial images is an emerging and challenging technique with many potential applications in computer vision. It involves the automatic process of verifying whether two or more individuals are blood relatives (kin) by analyzing their facial images. KV is a significant research field with applications such as finding missing persons, organizing family albums, and enhancing online image searches. Despite substantial progress in kinship verification over the past decade, there remain challenges such as intrinsic issues (e.g., differences in facial appearance) and extrinsic issues (e.g., varying imaging conditions). Additionally, there is a continuing need for more diverse datasets to improve the robustness and accuracy of KV systems.

In this work, we propose a new approach to kinship verification utilizing convolutional neural network (CNN) models. We employ the KIN Face II dataset, which includes four relationships: father-son, mother-son, mother-daughter, and father-daughter. Our approach leverages pre-trained deep CNN architectures to extract features from parent-child image pairs.

Specifically, we consider all different relationships proposed in the dataset, establishing kinship relations between them using various distance metrics. We then concatenate these distances into a single measure, calculated using four types of distance measurements. This method integrates similarity computation and feature extraction into a cohesive process, enhancing the efficiency and accuracy of kinship verification.

Our approach aims to address the intrinsic and extrinsic challenges by utilizing advanced deep learning techniques and a comprehensive feature extraction strategy. By leveraging the capabilities of CNNs and a robust dataset, we strive to push the boundaries of kinship verification, making it more reliable and applicable in real-world scenarios.

1.2 The Kinship

1.2.1 What is it ?

Kinship (KIN), is the most universal and fundamental of all human relationships, based on ties of blood, marriage, or adoption.

There are two primary types of kinship ties:

- Those based on blood, tracing descent.
- Those based on marriage, adoption, or other connections.

Some sociologists and anthropologists argue that kinship extends beyond familial ties to include social bonds,According to Encyclopaedia Britannica, kinship is a ”system of social organization based on real or putative family ties.” However, in sociology, kinship encompasses more than family ties. The Sociology Group describes kinship as one of the most important organizing components of society[7], stating:

KIN is one of the most important organizing components of society ,This social institution ties individuals and groups together and establishes a relationship among them,this can involve relationships between individuals unrelated by lineage or marriage, David Murray Schneider, a renowned professor of anthropology at the University of Chicago, emphasized this broader perspective in his studies of kinship,at its core, kinship refers to ”the bond of marriage and reproduction,” according to the Sociology Group. However, it can also encompass a wide range of groups or individuals based on their social relationships.

1.2.2 Importance

Kinship is important to a person and a community's well-being. Because different societies define kinship differently, they also set the rules governing kinship, which are sometimes legally defined and sometimes implied. At its most basic levels, according to the Sociology Group

1.3 Kinship in Computer Vision

The objective of kinship verification [8] is to determine whether a pair of given images of faces belong to individuals in a related relationship. Recent studies in psychology have shown that facial appearance plays a crucial role in identifying familial relationships. This is because biologically related individuals generally exhibit greater facial similarity compared to unrelated individuals.

Unfortunately, even for humans, predicting relatedness based solely on facial features can be challenging. However, advancements in machine learning and computer vision have made it possible for machines to tackle this problem effectively. Parental classification from facial images is a relatively new challenge in shape recognition and computer vision, with numerous potential applications in various real-world scenarios.

1.4 Kinship, its applications, and the motivations

Automated kinship verification using facial images has numerous applications. It can help locate relatives in public databases, determine the kinship of victims or suspects for law enforcement, validate asylum claims where family ties must be confirmed, and organize photo albums by identifying family members. This verification has significant security implications, such as identifying relatives of individuals deemed security threats.

Moreover, automated kinship determination can enhance facial recognition systems by using kinship traits as soft biometrics. The field of automated kinship verification in videos remains relatively unexplored but holds potential for security, surveillance, and immigration control. For example, during the investigation of the Boston Marathon bombing, video surveillance identified two male suspects who were later found to be brothers. An automated kinship verification system could have expedited their identification.

Another application is border control, where surveillance videos can verify the relationship between adults and children, preventing illegal child trafficking. Additionally, video-based kinship verification can validate or refute refugee and asylum seekers' familial claims. Currently, the U.S. Department of State uses DNA testing for family reunification; however, an automated kinship verification algorithm could provide real-time, cost-effective results.

Kinship information can also manage multimedia on social media platforms like Facebook and YouTube. Families often have different YouTube channels for uploading daily videos. Kinship data can automatically tag these videos and identify family members, aiding in the automatic indexing and organization of videos for easy searching.

There are several applications for kinship verification, including:

- **Photo Organization**: Organizing and resolving identities in photo albums by accurately identifying and tagging family members.

Figure 1.1: Illustration of photo organization

- **Missing Children Search**: Assisting in the search for missing children by identifying potential family connections.

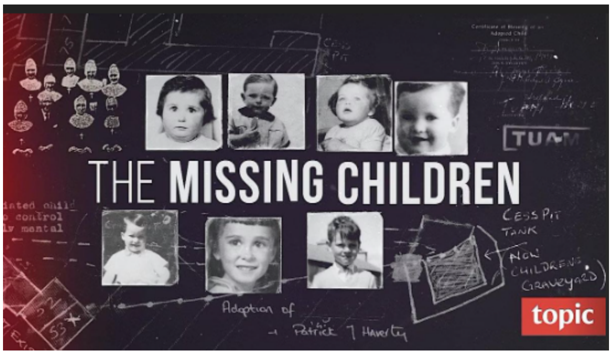


Figure 1.2: Illustration of missing children search

- **Surveillance**: Utilizing automated kinship verification in videos to aid security and surveillance efforts.

Figure 1.3: Illustration of surveillance

- **Genealogical Research**: Creating detailed and accurate family trees.

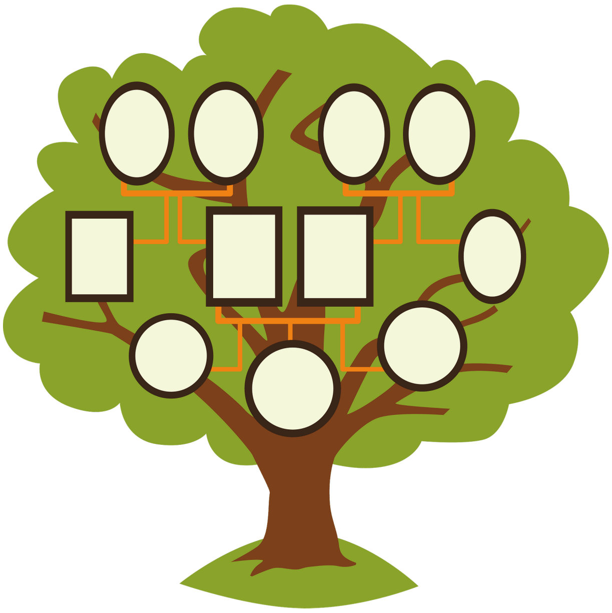


Figure 1.4: Illustration of genealogical research

1.5 Automatic recognition of kinship

Automated kinship recognition from face is a relatively recent problem that is mainly studied by the application of Deep Learning techniques. Despite the impact that an accurate kinship recognition algorithm can reach in a controlled environment, its applicability in smart environments is limited due to the degradation of performances. In this study we investigate the limitations of recent approaches that lead to a difficult applicability in a real case use. We present several tests on Siamese Neural Networks (SNN) based on a VGGFace architecture to solve both the kinship-vs-not-kinship recognition and the kind-of-kinship recognition. To perform our tests we used two popular kinship recognition Datasets that are Faces in the Wild and KinFace-II, respectively. To examine the behavior of the SNNs in a real scenario, we applied them, properly trained on the above mentioned datasets, to a popular TV show in which the aim is to discover kinship in a set of people. The weaknesses demonstrated in those tests have confirmed that the recent literature and algorithm to solve the kinship recognition problem are still far to achieve the high performances required in a smart environment [9].

1.6 Kinship verification and face verification

They are two distinct problems, but they share similarities in terms of image processing and facial feature analysis. Here's a brief overview of each

1.6.1 Face Verification

Face verification is a critical step in facial recognition systems, including kinship verification, as it relies on accurate face detection as a preliminary process. The verification process begins with capturing an image of a scene that contains a face. Subsequently, the face is extracted using a detection method, which isolates the region containing facial components such as the eyes, nose, and mouth. This extracted region is then refined through preprocessing procedures to enhance its quality. Accurate face verification is vital, as it directly impacts the reliability and performance of subsequent identification tasks, ensuring that kinship verification can be conducted effectively.

1.6.2 Kinship verification

Kinship verification involves extracting features from images of different individuals and verifying the relationship between them, typically by comparing the features extracted from different query images (image 1 and image 2) to determine if they belong to individuals who are related as per kinship[10].

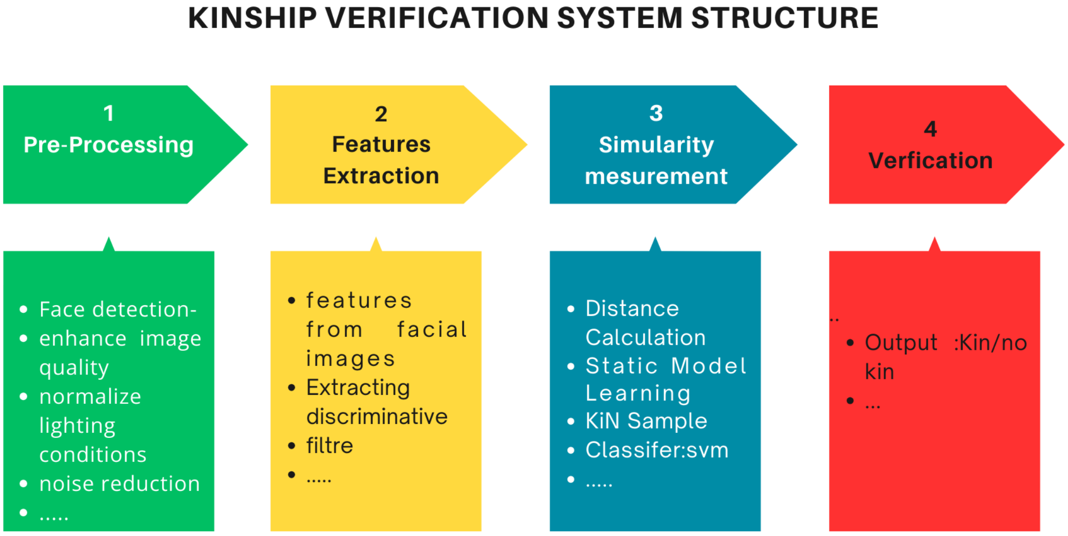
1.7 Kinship verification system structure

The structure of a Kinship Verification System typically involves several key components and processes aimed at automatically confirming biological relationships based on facial images. Here's an outline of its structure:

Due to the difficulty and complexity of kinship verification, a framework must be established to manage various approaches. This framework consists of four main components: Preprocessing, Feature Extraction, Similarity Measurement, and Verification.

Kinship Verification	**Face Verification**
• Extract features from different images of persons • Verify the relationship • Different traits of query image 1 and query image 2 • Highest level system • In decision stage: Kin or not Kin • Accuracy is around 90% practically	• Extract features from the same person image • Verify or identify • Same traits of query image 1 and query image 2 • High level system • In decision stage: matched or not matched • Performance of the machine is roughly as accurate as human

Table 1.1: Comparison between Kinship Verification and Face Verification [3]



KINSHIP VERIFICATION SYSTEM STRUCTURE

1 Pre-Processing

2 Features Extraction

3 Simularity mesurement

4 Verfication

- Face detection-
- enhance image quality
- normalize lighting conditions
- noise reduction
- .....

- features from facial images
- Extracting discriminative
- filtre
- .....

- Distance Calculation
- Static Model Learning
- KiN Sample
- Classifer:svm
- .....

..
- Output :Kin/no kin
- ...

Figure 1.5: General Kinship verification system structure

1.7.1 Preprocessing

The goal of the preprocessing stage is to make it easy to measure kinship similarity by extracting kin faces from photos and reducing the influences of various variations. This involves techniques such as face detection as MTCNN[11],YOLO[12] and alignment to ensure accurate extraction of facial features essential for kinship analysis. Additionally, optimizing lighting conditions and image quality can enhance the reliability of kinship measurements, facilitating more robust and precise results in familial relationship studies.

1.7.2 The features extraction phase

In image processing technology, whether it is binary, colored or gray. Image processing may be performed by extracting features for identification, classification, diagnosis, classification, clustering, recognition and detection. Feature extraction method are utilized to obtain much information as possible of image. The selection and effectiveness of feature chosen and extraction are a major challenge now [13] . Many methods used to extract features, which may depend on Geometric features, Statistical features, Texture features, and Color features. Each main type of feature divided into many subdivided types such as Color features divided into three types (Color moment, Color histogram and Average GBR). this Figure 1.6 shows the most important features methods.

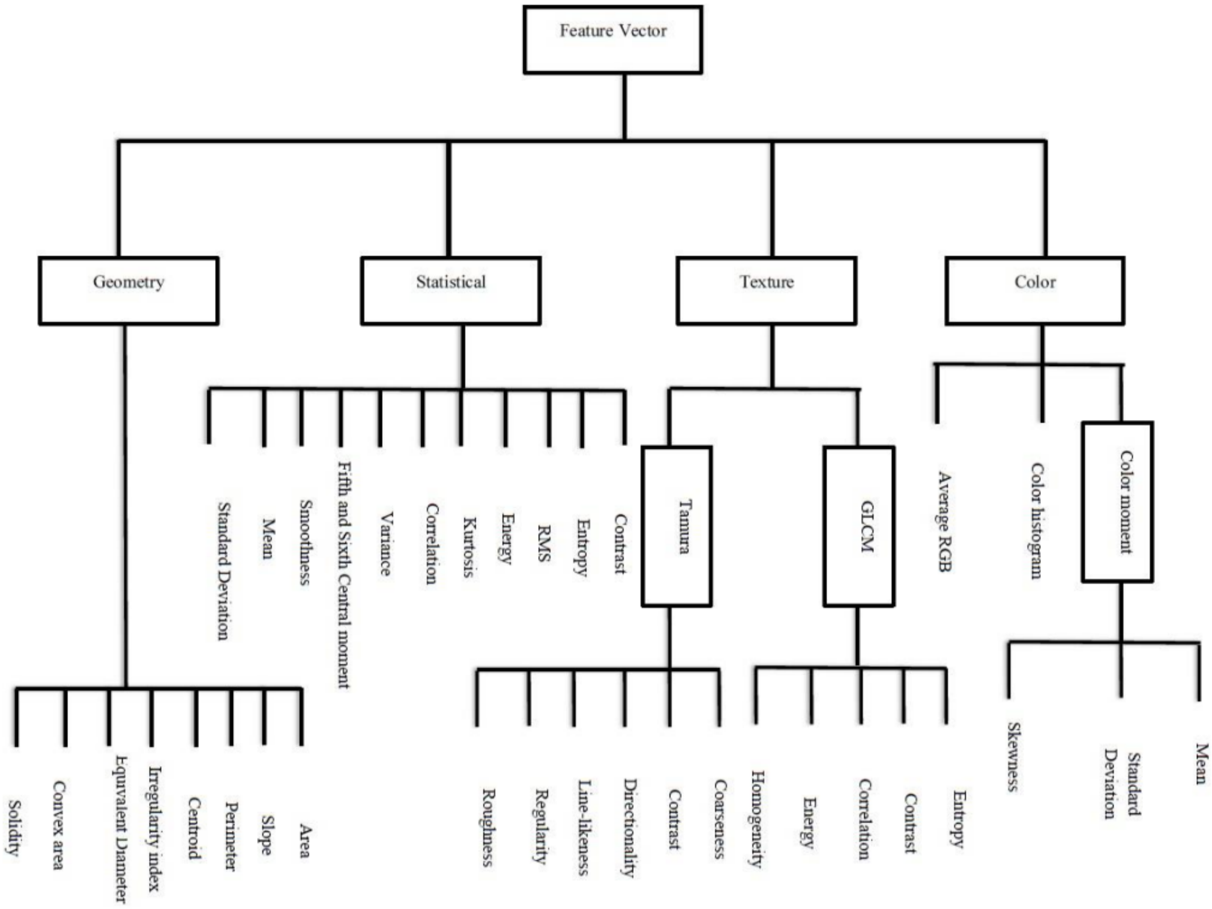


Figure 1.6: Features Extraction Methode[1]

1.7.3 Similarity measurement

Measuring similarity between feature vectors is indeed crucial in various applications such as machine learning, information retrieval, and recommendation systems. Here are some common methods used to measure similarity, such as Euclidean Distance, Manhattan Distance, Mahalanobis Distance, fusion of these, and other measuring methods.

The choice of similarity measure depends on the nature of the data and the specific problem you are trying to solve. Different measures have different computational requirements and assumptions about the data distributions.

1.7.4 Verification Phase

Next, these labeled feature vectors are used as input for a classification system, such as a Support Vector Machine (SVM) [14] or a Convolutional Neural Network (CNN) [15]. The classification system utilizes these inputs to learn and train the model, effectively distinguishing between kin and non-kin based on the provided features.

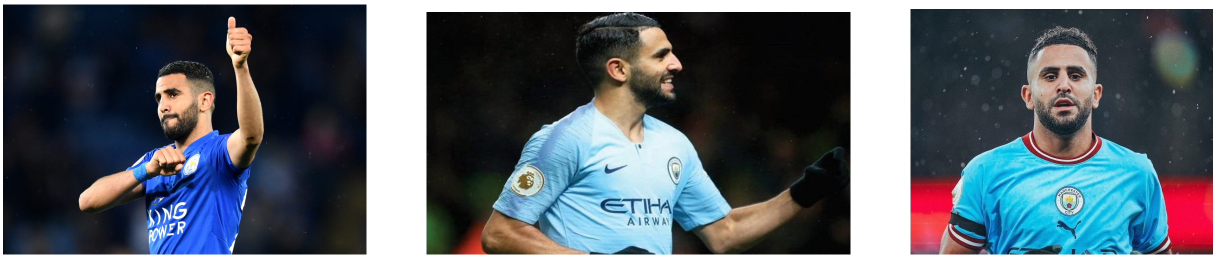
In this step, the dataset is divided into two sets: training and testing. After the model is trained, it can be evaluated by testing it on the test set and calculating the precision and accuracy percentages. The evaluation involves two cases: kin and non-kin, which helps verify how well the model generalizes to new, unseen data, ensuring that it can reliably identify kinship relationships in real-world scenarios.

1.8 Problems and challenges of kinship verification

Kinship verification faces several challenges, particularly in testing system performance with open datasets in real-world scenarios.

1.8.1 Pose variations

Head movements, including pitch, roll, and yaw, as well as changes in camera viewpoints, can cause substantial alterations in facial appearance and shape, posing challenges for automated face recognition across different poses. Correcting for pose is crucial and can be accomplished through effective techniques for rotating and aligning the face to the image's axis, thereby improving the accuracy and reliability of face recognition systems across varying angles and perspectives[16],[17].

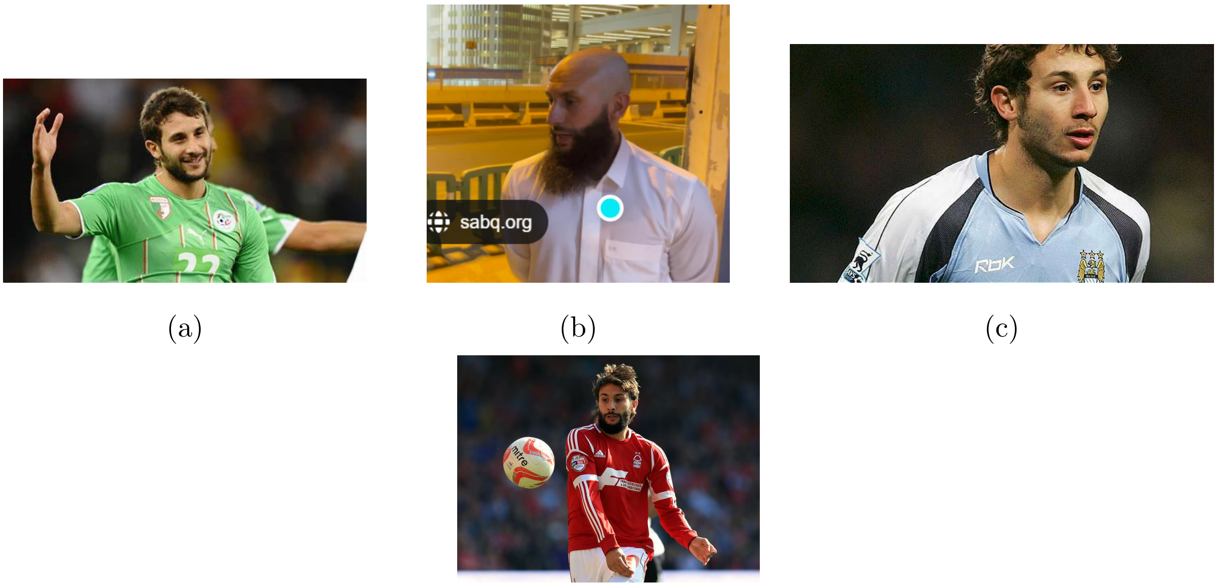


(a) (b) (c)

Figure 1.7: Illustration of pose variations

1.8.2 Presence or absence of structuring elements and occlusions

Face images captured in an unconstrained environment often necessitate the effective recognition of faces disguised or altered by accessories and/or occlusions. As illustrated in Fig I.3, elements such as hats, glasses, or beards can contribute to occlusions. Texture-based algorithms can assist in addressing these challenges [16] .



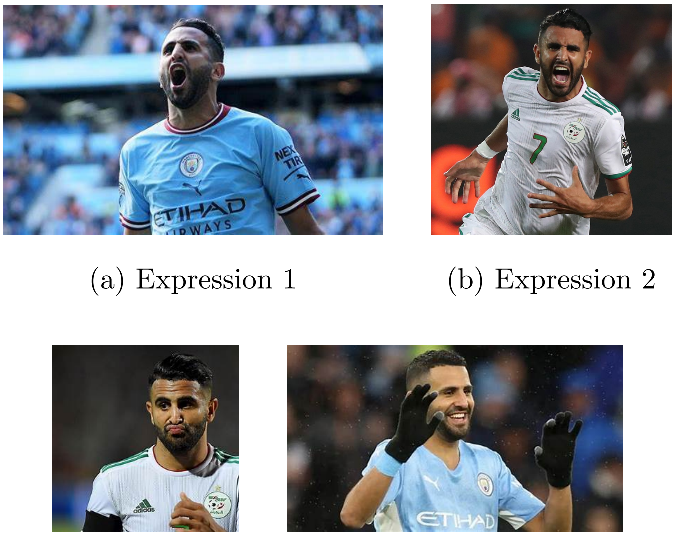
(a) (b) (c)

(d)

Figure 1.8: Illustration of Presence/absence of structuring elements/occlusions

1.8.3 Facial expression changes

Human expressions consist of macro-expressions, such as anger, disgust, fear, happiness, sadness, or surprise, as well as involuntary, rapid facial patterns known as micro-expressions. These expressions create non-rigid motions of the face, which are crucial for both evaluating emotional states and automated face recognition. Fig I.4 illustrates the variability in facial appearance caused by changes in emotional states.



(a) Expression 1

(b) Expression 2

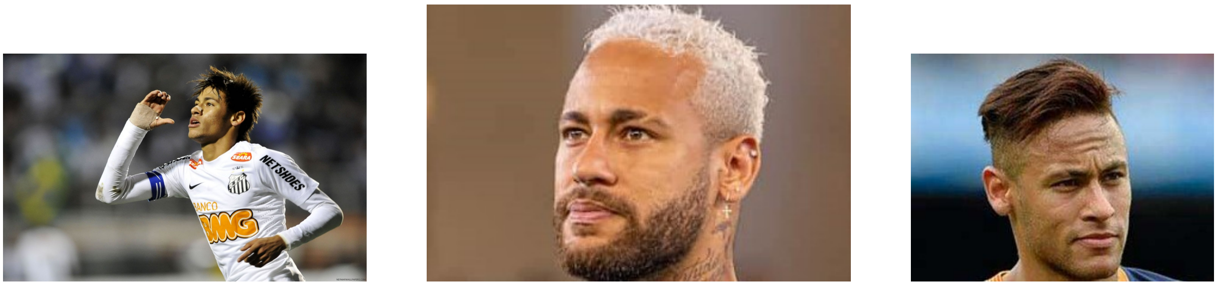
(c) Expression 3

(d) Expression 4

Figure 1.9: Illustration of Facial Expression Changes

1.8.4 Ageing of the face:

Face appearance changes can be caused by aging, which can significantly impact the face recognition process [17]. To overcome this issue, methods need to take into account facial aging patterns [10]. Additionally, integrating advanced machine learning algorithms that can adapt to these changes over time can enhance the robustness and accuracy of face recognition systems[16].



(a)

(b)

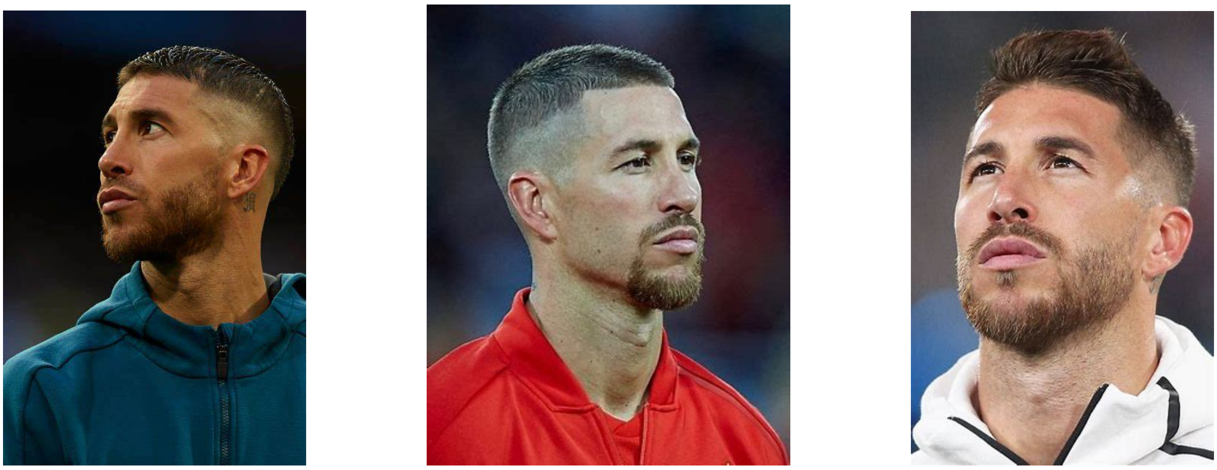
(c)

Figure 1.10: Illustration of pose variations

1.8.5 Varying illumination conditions

Large variations of illuminations can degrade the performance of AFR systems, with low levels of lighting making face detec- tion and recognition difficult. Too high levels of lighting can lead to overexposure

and indiscernible facial patterns. Image processing techniques such as illumination normalization and machine learning are used to deal with these variations[16].



(a) (b) (c)

Figure 1.11: Illustration of illumination variation

1.8.6 Image resolution and modality

AFR performance is influenced by the quality and resolution of the face image, the setup and modalities of digital equipment, and the use of different photographic hardware. Faces acquired in real-world conditions can present further challenges due to multiple modalities. Incorporating advanced preprocessing techniques and leveraging high-resolution sensors can mitigate some of these issues, ensuring more reliable face recognition even in diverse and uncontrolled environments [16].

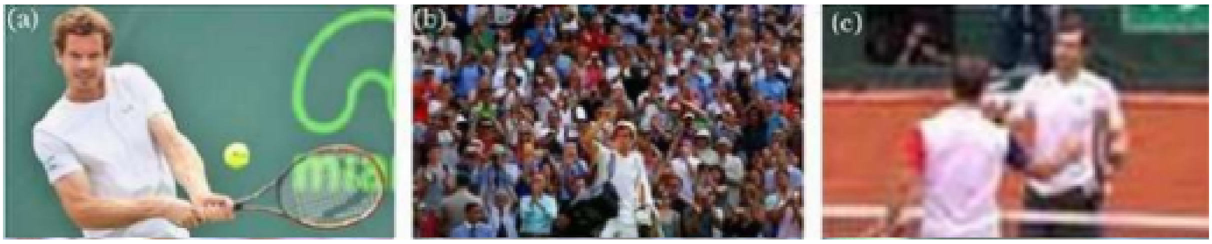


Figure 1.12: Illustration of variations of the image scale and resolution[2].

1.9 Conclusion

In this chapter, we provided a comprehensive discussion of the important concepts needed to understand kinship verification through still facial images. We assessed former approaches to the kinship problem in most aspects. Through the presentation of literature, we explored the problems and issues that affect the efficiency of a kinship verification model's performance.

Chapter 2

Kinship Verification Methods

2.1 Introduction

Automatic kinship verification is a burgeoning area of research that has garnered significant interest in recent years. This chapter focuses on face to explain detection methods, feature extraction, and classification process techniques essential for kinship verification. A facial image serves as a crucial indicator, containing numerous features that aid in identifying relationships. Accurately representing these features is vital for determining kinship, making this a major challenge in the field. Therefore, it is imperative to concentrate on the feature extraction step, as it notably impacts the performance of kinship verification systems.

This chapter begins with a discussion of the available facial databases pertinent to this field, followed by a review of related work. We then delve into deep learning and face detection methods, which constitute the initial steps of our system. Subsequently, we focus on feature extraction methods, emphasizing both handcrafted and deep features. Next, we explore feature reduction techniques that enable us to minimize the number of features without sacrificing essential information. Finally, we address deep learning approaches, specifically convolutional neural networks (CNNs), and classification methods, which are critical for achieving our objective of kinship verification.

2.2 Face detection

Over the past decade, numerous physiological studies have explored human capacity in facial recognition and verification. Researchers in computer vision and machine learning have developed various automatic approaches with differing performance levels. However, direct comparisons are challenging due to their use on distinct datasets. Additionally, consider the reference by Lu et al[18]

Pre-processing can detect facial landmarks, align facial data, and crop the facial area. It can filter irrelevant information such as hair, background, and reduce facial variations due to the change in pose. In 2D images, landmarks such as eyes, eyebrows, mouth, etc., can be reliably detected, while the nose is the most important landmark in 3D facial recognition. The 3D information (depth and texture maps) corresponding to the surface of the face can be acquired using different alternatives: a multi-camera system (stereoscopy), remote cameras or laser devices, and 3D scanner[19].

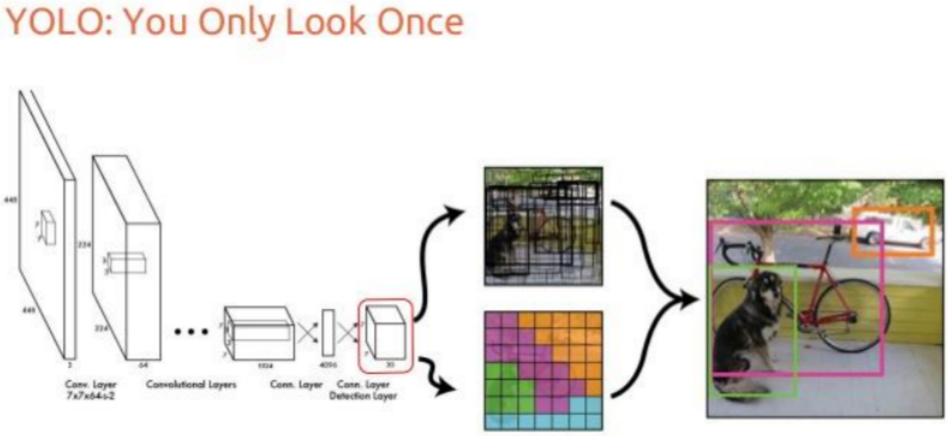
With the rapid increase in video and image databases, there is a growing need for intelligent systems to automatically understand and analyze this information, as manual processing has become impractical. Faces are crucial for conveying identity and emotions in social interactions. Humans excel at recognizing faces, but machines have become increasingly capable in this area. Automatic face detection systems are essential for various applications, including face recognition, facial expression analysis, head-pose estimation, and human-computer interaction. Face detection technology determines the location and size of human faces in digital images and is a prominent topic in computer vision literature[20].

2.2.1 YOLO

YOLO [69] is a state-of-the-art deep learning framework for real-time object detection. This enhanced model outperforms region-based detectors and has achieved superior results on standard detection datasets such as PASCAL VOC [56] and COCO [78]. Unlike traditional methods that treat object detection as a classification problem, YOLO approaches it as a regression problem. It performs all the necessary steps for object detection using a single neural network, resulting in both high detection performance and real-time speed. Additionally, YOLO has excellent generalization capabilities and can be easily trained to detect various objects.

The operation of YOLO is as follows: an image is divided into an SxS grid, and within each grid cell, m bounding boxes are considered. For each bounding box, the network predicts a class probability and offset values. Bounding boxes with class probabilities above a certain threshold are selected and used to locate objects in the image. YOLO is significantly faster (45 frames per second) than other object detection algorithms. However, it struggles with detecting small objects within the image, such as a flock of birds, due to spatial constraints.

Figure 2.1 illustrates the YOLO architecture.



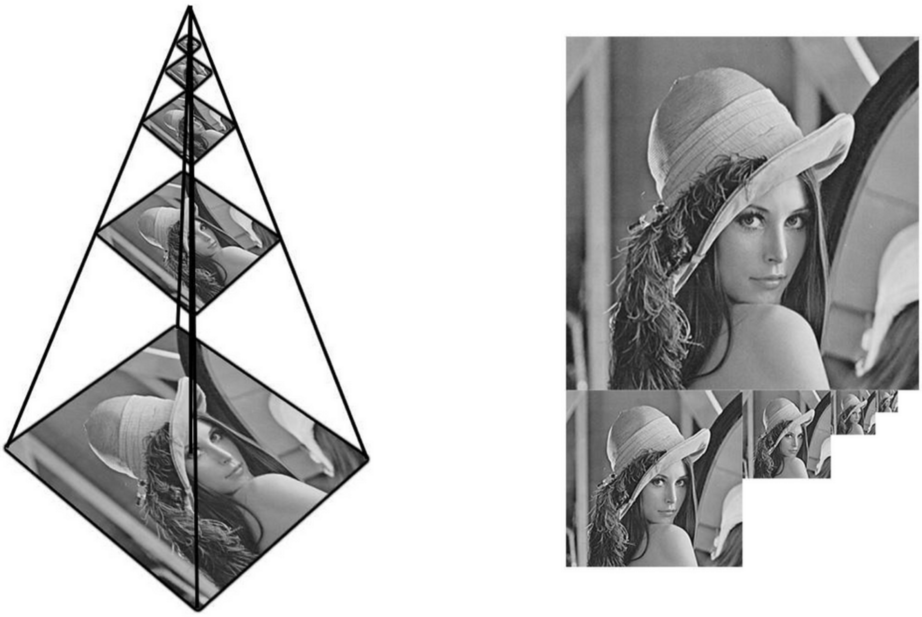
[]

Figure 2.1: YOLO architecture)[3]

2.2.2 MTCNN

MTCNN is a multitask neural network model for face detection, in order to take into account the performance and accuracy, and avoid the huge performance consumption caused by traditional ideas such as sliding window and classifier, it first uses small model to generate target region candidate box with certain possibility [21], and then uses more complex model for fine classification and higher precision region box regression, and makes this step recursive to form a three-layer network, namely p-net , R-Net, o-net, to achieve fast and efficient face detection. In the input layer, image pyramid is used to transform the scale of the initial image, and p-net is used to generate a large number of candidate target area frames. After that, R-Net is used for the first selection and border regression of these target area frames, and most of the negative examples are excluded. Then, the more complex and higher precision network o-net is used to discriminate and regress the remaining target area frames.

The Three Stages of MTCNN: The first step is to take the image and resize it to different scales in order to build an image pyramid, which is the input of the following three-staged cascaded network[1].

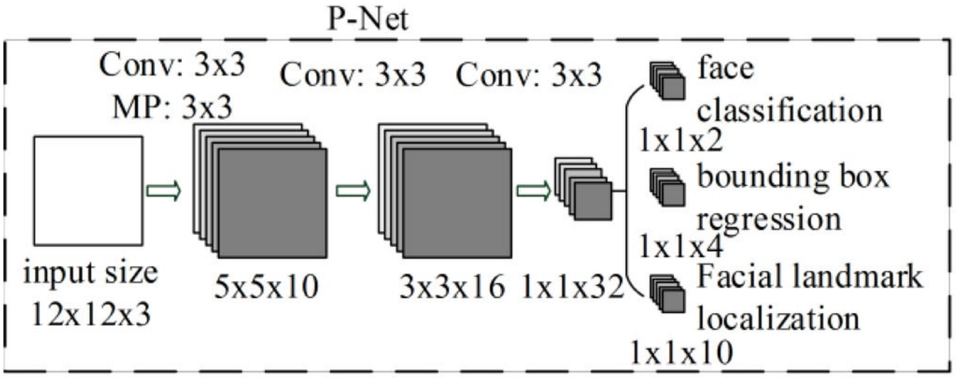


[]

Figure 2.2: image pyramid exemple[1]

- **Stage 1: The Proposal Network (P-Net)**:This first stage is a fully convolutional network (FCN). The difference between a CNN and a FCN is that a fully convolutional network does not use a dense layer as part of the architechture. This Proposal Network is used to obtain candidate windows and their bounding box regression vectors.

  Bounding box regression is a popular technique to predict the localization of boxes when the goal is detecting an object of some pre-defined class, in this case faces. After obtaining the bounding box vectors, some refinement is done to combine overlapping regions. The final output of this stage is all candidate windows after refinement to downsize the volume of candidates.

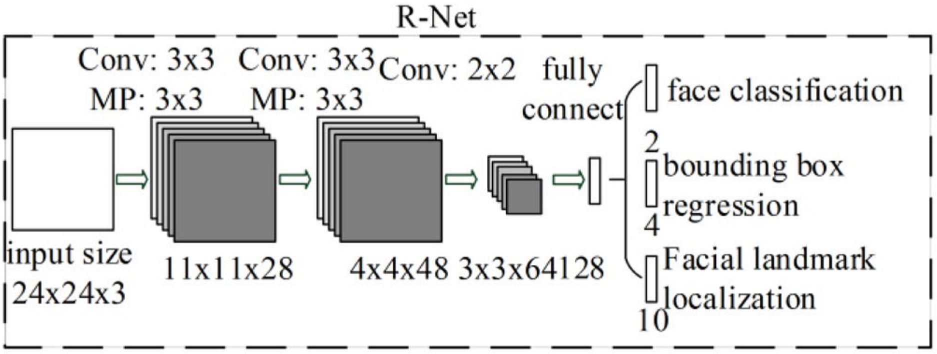


[]

Figure 2.3: P-Net [4]

- **Stage 2: The Refine Network (R-Net)**:All candidates from the P-Net are fed into the Refine Network. Notice that this network is a CNN, not a FCN like the one before since there is a dense layer at the last stage of the network architecture. The R-Net further reduces the number of candidates, performs calibration with bounding box regression and employs non-maximum suppression (NMS) to merge overlapping candidates.

  The R-Net outputs wether the input is a face or not, a 4 element vector which is the bounding box for the face, and a 10 element vector for facial landmark localization.



[]

Figure 2.4: R-Net[4]

- **Stage 3: The Output Network (O-Net):**

  This stage is similar to the R-Net, but this Output Network aims to describe the face in more detail and output the five facial landmarks' positions for eyes, nose and mouth.

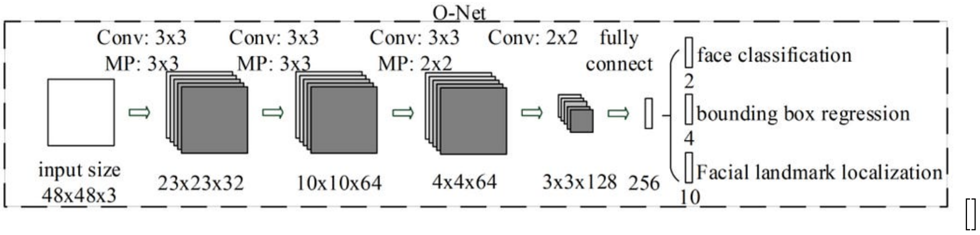


Figure 2.5: O-Net [4]

2.3 Features Extraction:

Feature extraction methods for kinship verification can be classified into two categories: shallow (handcrafted) features and deep learning-based features. Initially, shallow feature extraction methods dominated the field. These handcrafted features involve manually designed algorithms to identify specific characteristics in images, such as textures, edges, and shapes.

However, the advent of deep learning has revolutionized feature extraction by automatically learning hierarchical features from raw data. Despite its potential, deep learning was not widely applied in the field of automatic kinship verification due to the lack of sufficient data to train these models effectively.

Recent advancements in data augmentation and transfer learning techniques have started to mitigate this limitation, enabling deep learning models to perform better even with limited data. These models can now capture more complex patterns and relationships in the data, leading to improved accuracy and robustness in kinship verification tasks.

2.3.1 Shallow Features:

For over a decade, shallow features have been extensively used in various computer vision applications, including object detection and image classification. These handcrafted features rely on manually designed algorithms to extract specific characteristics from images, such as textures, edges, and shapes. In the context of parentage verification, several descriptors can be employed, such as:

2.3.1.1 Local Binary Patterns (LBP)

LBP encodes the local differences in gray levels. We consider a circular neighborhood of radius R (a circle consisting of P points), around the central point with coordinates (x_c, y_c) and gray level g_c.

$$\text{LBP}(x_c, y_c) = \sum_{p=0}^{P-1} 2^p \delta(g_p - g_c)$$

where δ is the Heaviside function, such that $\delta(g_p - g_c) = 1$ if $g_p \geq g_c$ and 0 otherwise. For a neighborhood of 8 points with a radius $R = 1$, the LBP values are encoded between 0 and 255. A texture can be characterized by the histogram of these LBPs[22].

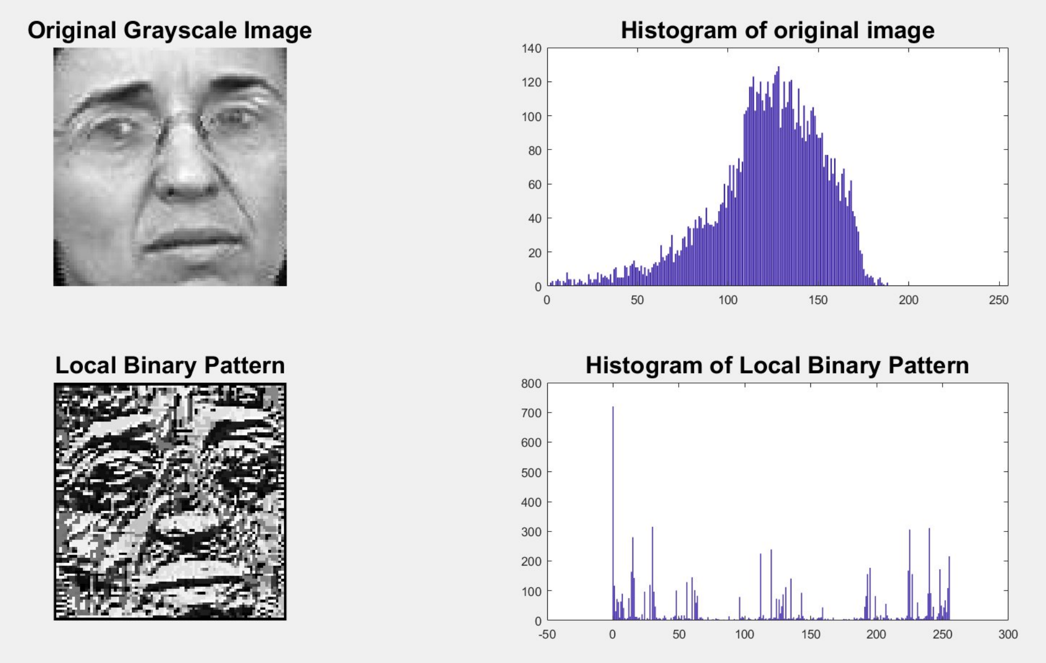


Figure 2.6: texture extraction using Local Binary Pattern (LBP)[?, 5]

2.3.1.2 Binarized Statistical Image Features (BSIF)

Unlike LBP , which can be used to calculate label statistics in local pixel neighborhoods, the local descriptor called BSIF (Binarized Statistical Image Features), which was recently proposed by Kannala and Rahtu, uses a predefined set of manually designed linear filters and binarizes the filter responses [23].

Given a patch image X of size $l \times l$ pixels and a linear filter W_i of the same size, the filter response s_i is obtained by:

$$s_i = \sum_{u,v} W_i(u,v)X(u,v) = w_i^T x \tag{2.5}$$

Given n linear filters W_i, we can stack them into a matrix W and compute all responses at once.

Where vector notation is introduced in the last step, namely, vectors w and x contain the pixels of W_i and X, respectively. The binarized function b_i is obtained by:

$$b_i = \begin{cases} 1 & \text{if } s_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

Where b_i is the i-th element of b. This way, an n-bit binary code can be computed for each pixel, and subsequently, the image region can be represented by histograms of these binary codes of the pixels.

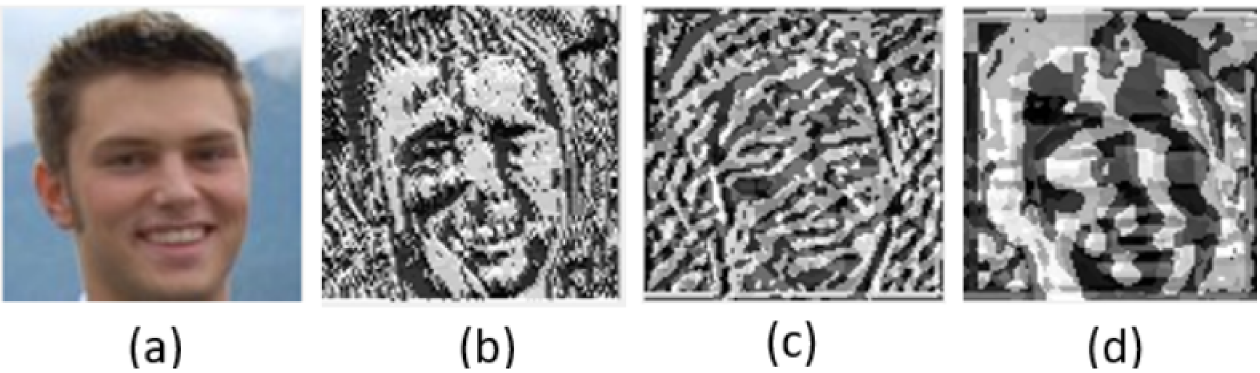


Figure 2.7: Sample images of Binarized statistical image features (BSIF) descriptor.With: a) Input Image, b-c-d-e)result of BSIF with filterof 3x3, 7x7 and 11x11 dimension respectively. [5]

2.3.1.3 Local Optimal Oriented Patterns (LOOP)

The Local Optimal Oriented Patterns (LOOP) [24], a recent texture descriptor, encodes local structures and repeated patterns within images. Compared to commonly used feature descriptors such as Local Directional Pattern (LDP) and Local Binary Pattern (LBP), LOOP has shown superior performance in various image recognition tasks. It excels in capturing intricate local information, making it suitable for applications like facial recognition.

To derive the final codes, three steps are involved. First, edge responses of pixels with gray values g_i ($i = 0, 1, \ldots, 7$) in eight directions are determined using Kirsch masks. Second, binarization weights w_i are assigned to pixels based on the rank of the mask response value. Finally, these weights are integrated into the LBP formula to compute the final code relative to the center pixel:

$$\mathrm{LOOP}(x_c, y_c) = \sum_{i=0}^{7} \xi(g_i - g_c) \cdot 2^{w_i} \tag{2.7}$$

where the function ξ is defined as:

$$\xi(\eta) = \begin{cases} 1, & \text{if } \eta \geq 0 \\ 0, & \text{if } \eta < 0 \end{cases} \tag{2.8}$$

Here, g_c refers to the gray level of the center pixel located at (x_c, y_c).

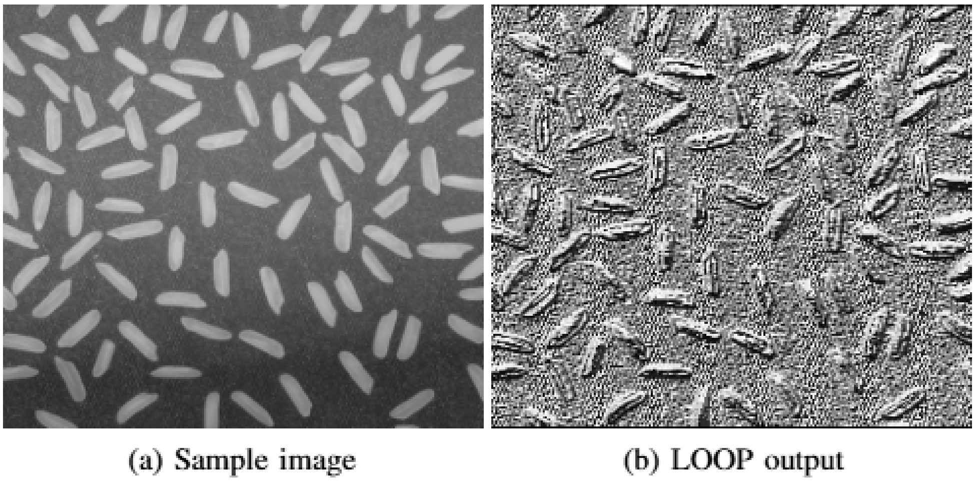

(a) Sample image
(b) LOOP output

Figure 2.8: Standard test image (rice.png) and the LOOP output[6].

While these shallow features have shown effectiveness in many applications, the emergence of deep learning has introduced more sophisticated and automated feature extraction methods. Deep learning models, with their ability to learn hierarchical features directly from data, are gradually becoming more prominent in kinship verification as data availability and computational power increase.

2.3.2 Feature Extraction Using Deep Learning

Deep neural networks (DNNs) can serve as powerful feature extractors. For instance, you can take a pre-trained CNN (such as VGG16 or ResNet) and remove its fully connected

(FC) layers, leaving only the convolutional layers.

2.3.2.1 VGG

This method is initially based on a deep convolutional neural network (ConvNet) architecture first proposed by K. Simonyan and A. Zisserman [25] from the Visual Geometry Group at the University of Oxford in 2014. The VGG architecture forms the basis of revolutionary object recognition models. Developed as a deep neural network, VGGNet also surpasses benchmark performances in various tasks and datasets beyond the network itself. Furthermore, it remains one of the most popular image recognition architectures today.

VGG Architecture

- **Input:** VGGNet takes an input image of 224×224. For the ImageNet competition, the model's creators cropped the central 224×224 patch of each image to maintain consistent input size.
- **Convolutional Layers:** VGG's convolutional layers utilize a minimal receptive field, i.e., 3×3, the smallest size possible to capture top/bottom and left/right features. Additionally, there are 1×1 convolution filters acting as a linear transformation of the input. These are followed by a ReLU unit, a major innovation of AlexNet that reduces training time. ReLU stands for ”rectified linear unit activation function”; it is a piecewise linear function that outputs the input if positive; otherwise, the output is zero. The convolution stride is set to 1 pixel to preserve spatial resolution after convolution (stride is the number of pixel shifts over the input matrix).
- **Hidden Layers:** All hidden layers of the VGG network use ReLU. VGG typically does not use Local Response Normalization (LRN) because it increases memory consumption and training time without improving overall accuracy.
- **Fully Connected Layers:** VGGNet includes three fully connected layers. Among these, the first two layers have 4096 channels each, and the third has 1000 channels, one for each class.

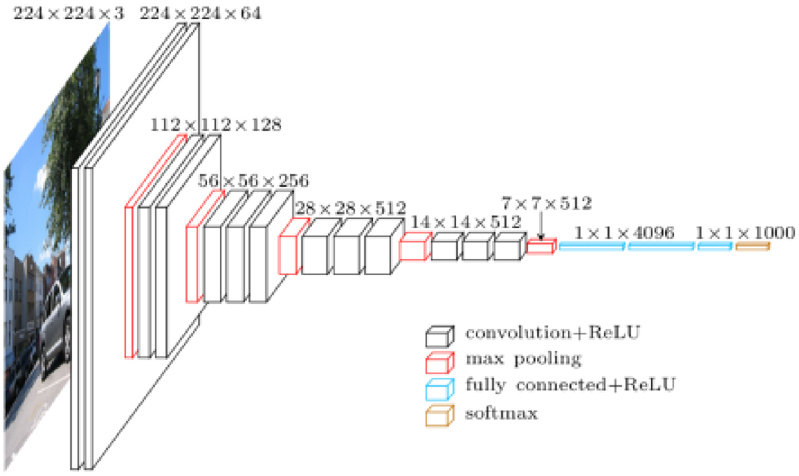


Figure 2.9: VGG Neural Network Architecture

- **VGG16**: The VGG model, or VGGNet, supporting 16 layers is also known as VGG16. It is a convolutional neural network model proposed by A. Zisserman and K. Simonyan from the University of Oxford. The VGG16 model achieves an accuracy of nearly 92.7

- **VGG19**: The concept of the VGG19 model (also VGGNet-19) is similar to VGG16, except that it supports 19 layers. The numbers ”16” and ”19” denote the number of weight layers in the model (convolutional layers). This means VGG19 has three more convolutional layers than VGG16. The VGG network is constructed with very small convolutional filters. VGG-16 consists of 13 convolutional layers and three fully connected layers.

2.3.2.2 Residual Networks (ResNets)

: Residual Networks, or ResNets, learn residual functions with reference to the layer inputs, instead of learning unreferenced functions. Instead of hoping each few stacked layers directly fit a desired underlying mapping as $H(x)$, ResNets let the stacked nonlinear layers fit another mapping $F(x) = H(x) - x$. The original mapping is recast into $F(x) + x$ [26].

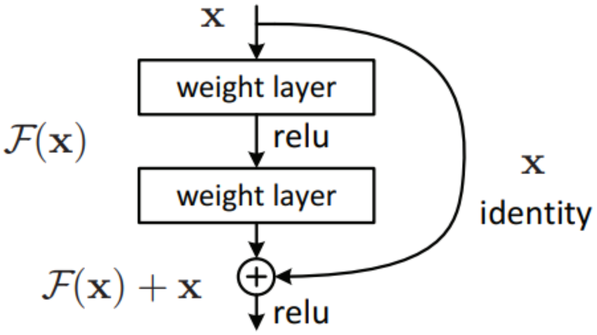

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.10: Residual Network Architecture [7]

.

2.4 Classification Algorithms

A classification algorithm is a supervised learning technique used to identify the category of new observations based on a training dataset. In classification, a program learns from a labeled dataset and then classifies new observations into one of several predefined classes. For example, classes can be "Yes" or "No," "0" or "1," "Spam" or "Not Spam," "cat" or "dog," etc. These classes can also be referred to as targets, labels, or categories. Unlike regression, where the output variable is a continuous value, the output variable in classification is categorical, such as "green or blue," "fruit or animal," etc. As a supervised learning technique, classification algorithms require labeled input data, meaning the input data comes with corresponding output labels.

In a classification algorithm, a discrete output function (y) is mapped to an input variable (x). The primary goal of the classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output of categorical data. The algorithm that implements classification on a dataset is known as a classifier. Below, we list some of the most commonly used classification algorithms:

2.4.1 The k-NN algorithm

Definition: The k-NN algorithm, also known as KNN, is a non-parametric supervised learning method. It uses proximity to perform classifications or predictions for an individual data point. Specifically, it identifies the *k* nearest neighbors to a given data point and assigns a class label based on a majority vote. For classification problems, the most frequently represented class label around the data point is used. Note that the term "majority vote" is commonly used, even if it technically requires a majority above 50%. In multi-class scenarios, a label can be assigned with a vote exceeding 25%[27].

- **Regression:** In regression problems, k-NN computes the average of the k nearest neighbors' values to make predictions. Unlike classification (which deals with discrete labels), regression handles continuous values.
- **Distance Metric:** Before classification, a distance metric must be defined. The Euclidean distance is commonly used for this purpose.
- **Lazy Learning:** KNN is part of the "lazy learning" family, meaning it stores only the training data and performs calculations during classification or prediction. It relies heavily on memory and is sometimes called an instance-based or memory-based learning method.

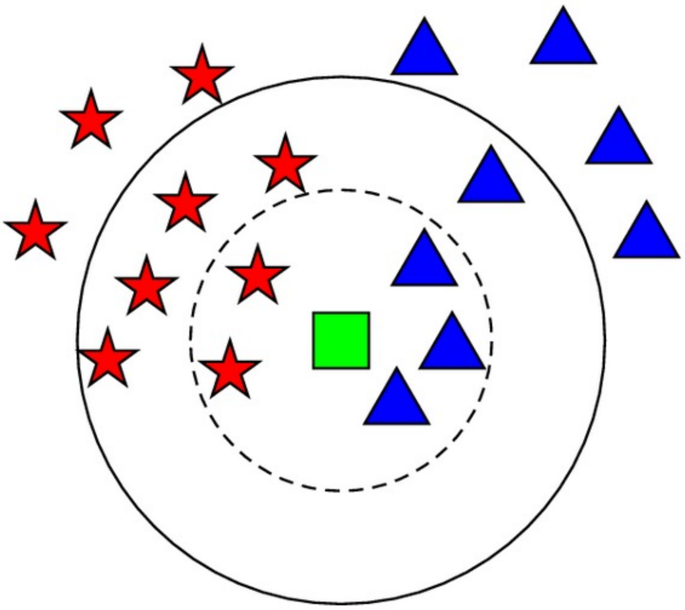


Figure 2.11: An example of k-NN classification..

2.4.2 A decision tree

A decision tree is a classifier represented as a recursive partition of the instance space. The decision tree [28] consists of nodes forming a rooted tree, meaning it is a directed tree with a node called the "root" that has no incoming edges. All other nodes have exactly one incoming edge. A node with outgoing edges is called an internal or test node. All other nodes are called leaves (also known as terminal or decision nodes). In a decision tree, each internal node splits the instance space into two or more sub-spaces according to a discrete function of the input attributes. The root and internal nodes are associated with attributes, while the leaf nodes are associated with classes.

Essentially, each non-leaf node has an outgoing branch for every possible value of the attribute associated with the node. To determine the class for a new instance using a decision tree, starting from the root, successive internal nodes are visited until a leaf node is reached. At the root node and each internal node, a test is applied. The result of the test determines the branch traversed and the next node visited. The class of the instance is the class of the final leaf node. The estimation criterion [29] in the decision tree algorithm involves selecting an attribute to test at each decision node of the tree. The goal is to select the most useful attribute for classifying the examples. A good quantitative measure of an attribute's value is a statistical property called information gain, which measures the ability of a given attribute to separate training examples based on their target classification. This measure is used to select among the candidate attributes at each step during the tree's growth [30].

The following figure (2.12) shows an example of a decision tree used to predict "yes" or "no" in response to the question "Is the weather suitable for playing outside?"

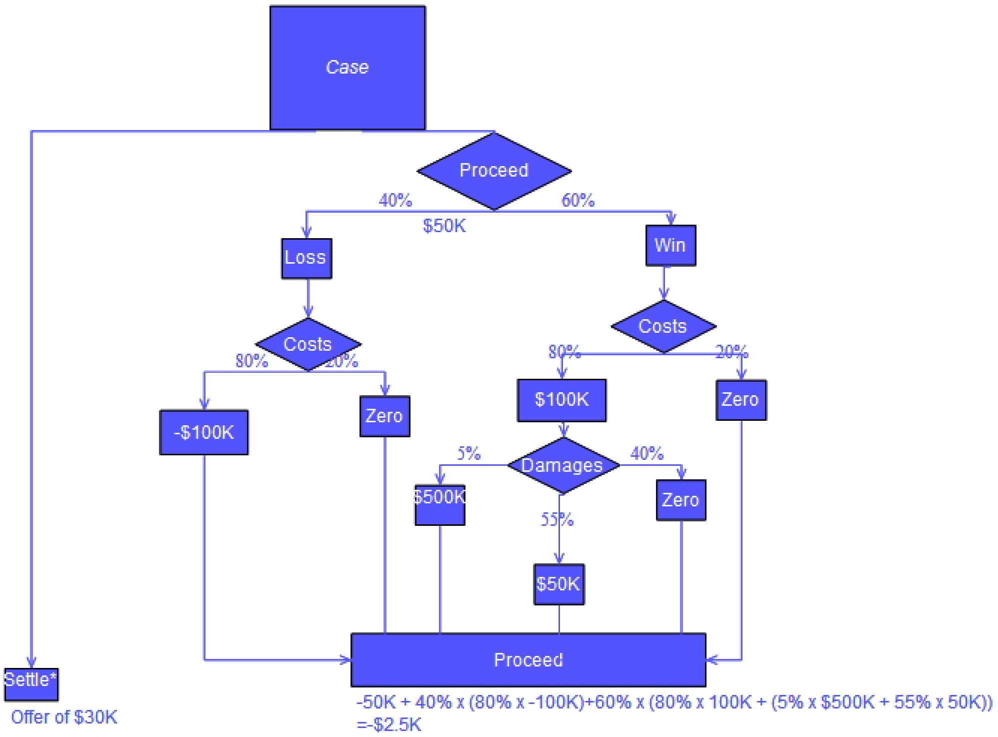


Figure 2.12: An example of a decision tree predicting "yes" or "no" for playing outside based on the weather[8].

2.4.3 Support Vector Machine (SVM)

A support vector machine (SVM) is machine learning algorithm that analyzes data for classification and regression analysis. SVM is a supervised learning method that looks at data and sorts it into one of two categories. An SVM outputs a map of the sorted data with the margins between the two as far apart as possible. SVMs are used in text categorization, image classification, handwriting recognition and in the sciences[31],

noindent An SVM is a supervised machine learning algorithm used for classification and regression analysis. Here's how it works:

- **Classification:** Given labeled data (examples with known categories), SVM finds an optimal hyperplane that separates the data into different classes. The goal is to maximize the margin (distance) between the hyperplane and the nearest data points of each class.

- **Regression:** In regression tasks, SVM predicts a continuous output based on input features. It aims to find a hyperplane that best fits the data while minimizing errors.

The equation for Support Vector Machine (SVM) depends on the specific variant and formulation. Let's focus on the most common case: linear SVM for binary classification.

Objective Function: The optimization formula aims to find the optimal hyperplane that maximizes the margin between classes. It involves minimizing the following objective function:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\max(0, 1 - y_i(w^T x_i + b))$$

Here:

- w represents the weight vector.
- b is the bias term.
- x_i are the feature vectors of the data points.
- y_i are the corresponding class labels ($y_i \in \{-1, +1\}$).
- C is the regularization parameter that balances margin maximization and training error.

Decision Function: The decision function for linear SVM is given by:

$$f(x) = w^T x + b$$

If $f(x) > 0$, the data point is classified as one class; otherwise, it belongs to the other class.

Kernel Trick: SVMs can handle non-linear data by using the kernel trick. The kernel function transforms the input features into a higher-dimensional space, making it possible to find a linear hyperplane in that space.

Common kernel functions include:

- Polynomial Kernel: $K(x, x') = (a + x^T x')^b$
- Radial Basis Function (RBF) Kernel: $K(x, x') = \exp(-\gamma\|x - x'\|^2)$

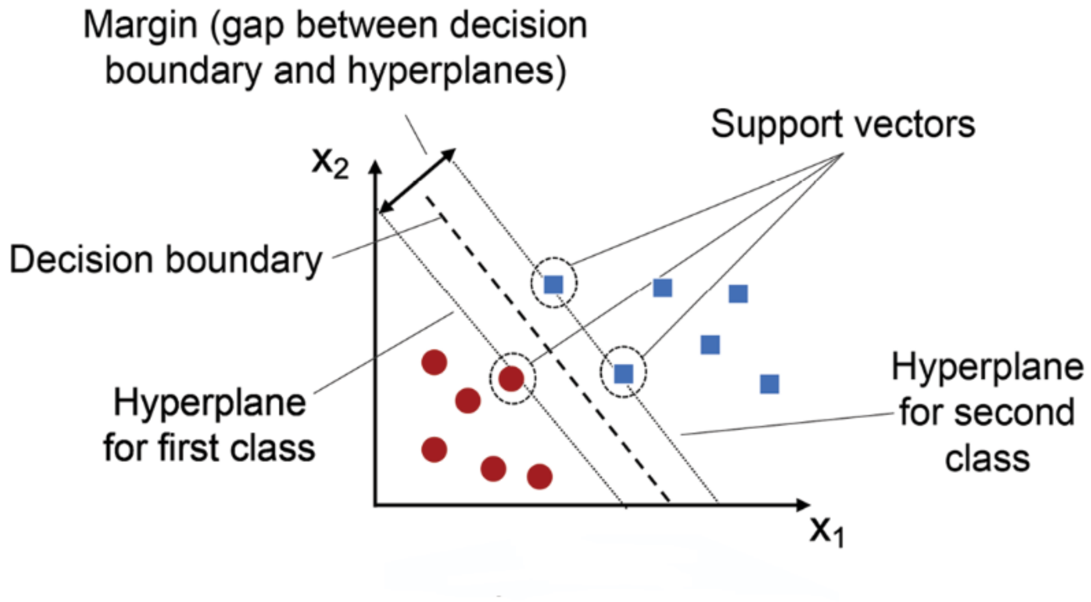

Margin (gap between decision boundary and hyperplanes)
Support vectors
x_2
Decision boundary
Hyperplane for first class
Hyperplane for second class
x_1

Figure 2.13: SVM[8].

2.4.4 pretrained CNN models for the Kinship System

Convolutional Neural Networks (CNNs) pretrained on large-scale datasets have been instrumental in advancing kinship verification systems. These models leverage deep learning architectures that have been trained on extensive image datasets such as ImageNet. By utilizing pretrained CNNs, researchers benefit from features learned through millions of images, which enhances the model's ability to extract meaningful features relevant to kinship verification.

Popular pretrained CNN models used in kinship verification include:

- **VGG16 and VGG19**: These models are known for their deep architectures and high performance on image classification tasks. They have been adapted and fine-tuned for kinship verification by extracting features from facial images of relatives.
- **ResNet (Residual Networks)**: ResNets introduce skip connections to address the vanishing gradient problem in deep networks. They have shown improved performance in various computer vision tasks, including kinship verification, by learning residual functions with reference to layer inputs.
- **Inception (GoogLeNet)**: The Inception models are known for their inception modules that use multiple filter sizes within the same convolutional layer. This

design allows them to capture features at different scales, which can be beneficial for kinship verification where subtle facial similarities are important.

- **MobileNet**: MobileNets are lightweight CNN architectures designed for mobile and embedded vision applications. They offer a good balance between accuracy and computational efficiency, making them suitable for real-time kinship verification systems.

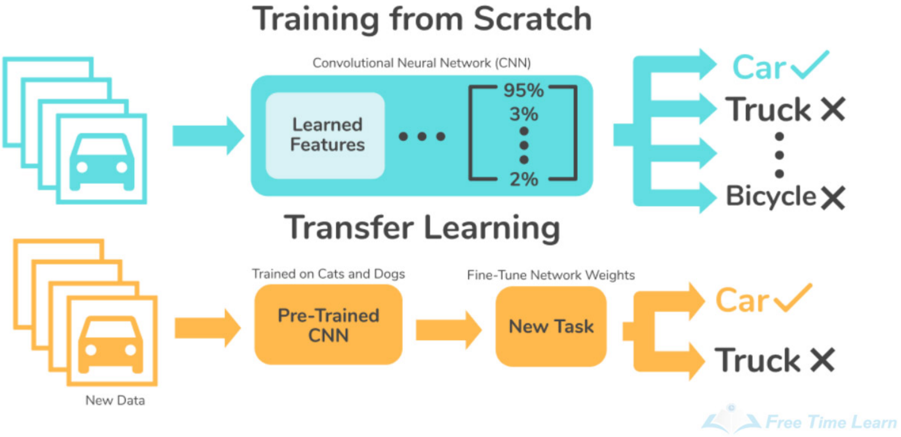


Figure 2.14: Common Pretrained CNN Models for Kinship Verification

These pretrained models provide a robust foundation for kinship verification systems, enabling efficient extraction of discriminative features from facial images to determine familial relationships.[52].

2.5 Decision:

After classification, decision making involves interpreting the results and taking actions based on the model's predictions. This process includes the following steps:

- **Performance Evaluation**: Use various metrics to evaluate the model's performance comprehensively. These metrics provide different aspects of how well the model is performing:
  - **Accuracy**: Measures the ratio of correctly predicted instances to the total instances.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- **Recall (Sensitivity or True Positive Rate)**: Reflects the ability of the model to correctly identify positive instances.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **Precision**: Indicates the ratio of correctly predicted positive observations to the total predicted positives.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **F1-Score**: The harmonic mean of precision and recall.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **AUC-ROC (Area Under the Receiver Operating Characteristic Curve)**: Measures the ability of the model to distinguish between classes.

$$\text{AUC-ROC} = \int_0^1 \text{TPR}(fpr)\, d(fpr)$$

where TPR is the True Positive Rate and fpr is the False Positive Rate.

- **Results Interpretation**: Analyze the predictions to understand the decisions made by the model. This involves several aspects:
  - **Confusion Matrix**: A table that summarizes the performance of a classification model.
  - **Feature Importance**: Determine which features or variables had the most significant influence on the model's predictions.
  - **Error Analysis**: Investigate instances where the model made incorrect predictions.
- **Implementation of actions**: Take practical decisions based on the model's predictions. This involves deploying the model's outputs to make informed decisions:
  - **Automated Decision Systems**: Integrate the model into automated systems to make real-time decisions.
  - **Human-in-the-Loop Systems**: Use the model's predictions to assist human decision-makers.
  - **Policy Making**: Utilize the model's insights to inform policy and strategy decisions within an organization.

2.6 Related work

The early research on kinship verification laid the foundation for subsequent advancements in this field. The use of the "Cornell KinFace" database allowed researchers to explore the similarity between parent-child pairs. Since then, the field has evolved significantly, and larger and more diverse datasets have become available for studying kinship relationships. When we looked into this and examined the papers on 'Facial Kinship Verification' published in 2022, we learned that the first kinship verification study was conducted in 2010, using traditional methods alongside machine learning. This figure shows the evolution of studies on kinship, highlighting how modern techniques like deep learning and neural networks have further propelled the accuracy and application of kinship verification systems.

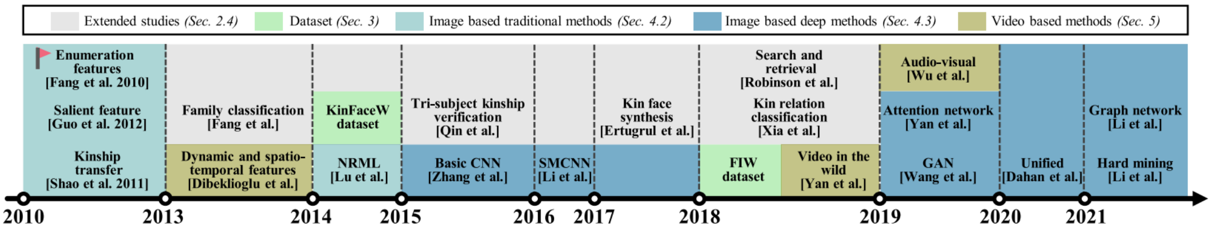

Extended studies (Sec. 2.4)
Dataset (Sec. 3)
Image based traditional methods (Sec. 4.2)
Image based deep methods (Sec. 4.3)
Video based methods (Sec. 5)
Enumeration features [Fang et al. 2010]
Salient feature [Guo et al. 2012]
Kinship transfer [Shao et al. 2011]
Family classification [Fang et al.]
Dynamic and spatio-temporal features [Dibeklioglu et al.]
KinFaceW dataset
NRML [Lu et al.]
Tri-subject kinship verification [Qin et al.]
Basic CNN [Zhang et al.]
SMCNN [Li et al.]
Kin face synthesis [Ertugrul et al.]
Search and retrieval [Robinson et al.]
Kin relation classification [Xia et al.]
FIW dataset
Video in the wild [Yan et al.]
Audio-visual [Wu et al.]
Attention network [Yan et al.]
GAN [Wang et al.]
Unified [Dahan et al.]
Graph network [Li et al.]
Hard mining [Li et al.]
2010
2013
2014
2015
2016
2017
2018
2019
2020
2021

Figure 2.15: The first facial kinship verification study[9]

They initially focused on confirming the similarity between parent-child pairs by extracting 22 facial features, including skin color, eyes, mouth, distance features, and statistical features like the Histogram of Gradients (HOG). They used either a Support Vector Machine (SVM) with a radial basis function or the K-Nearest Neighbor (KNN) classifier with an Euclidean metric to identify pairs of faces. Although this method produced positive results, it had limitations in proving kinship verification due to physical and genetic differences, such as age gaps between parents and children or gender differences between siblings. Subsequent research has explored more sophisticated techniques, including deep learning, to address these challenges and improve the accuracy and reliability of kinship verification systems.

Statistical-based genetic studies have demonstrated a critical observation: the faces of parents when they were young resemble those of their children more than the images captured when they are older. This insight prompted the creation of the UB KinFace database, which includes images of children's faces, young parents, and elderly parents,To address the significant distribution divergence between children and elderly parents, Xia et al, proposed the transfer learning method (TSL) [32]. This method uses an intermediate distribution close to both distributions and employs Gabor wavelets for feature extraction,This approach improved the overall accuracy of kinship verification and enhanced the discriminative power of the task [33].

Additionally, recent advancements in deep learning and convolutional neural networks have further refined feature extraction and classification processes, leading to even higher accuracy and robustness in kinship verification systems.

Afterwards, Lu et al. [34] collected the KinFaceW-I and KinFaceW-II databases to support larger-scale research, further motivating researchers to contribute to this topic. They also proposed the Neighborhood Repelled Metric Learning (NRML) method. Metric learning allows for the development of a distance metric that minimizes the distances between pairs of positive images with kinship links while pushing apart pairs of images without such links. This method was tested using various local feature descriptors, including Local Binary Pattern (LBP), Histogram of Oriented Gradients (HOG), and Scale-Invariant Feature Transform (SIFT) [12]. Additionally, this approach has paved the way for incorporating advanced machine learning techniques, such as deep metric learning, which continues to improve the accuracy and efficiency of kinship verification systems.

Fang et al. proposed a novel approach for kinship verification using their "Family101" dataset [35], modeling the problem as reconstructing a face from shuffled parts of a set of families, inspired by the biological process of inheritance. Instead of analyzing the whole face, their approach segments the face into parts (eyes, nose, mouth, etc.) and reconstructs each part as a linear combination of corresponding parts from the database. To evaluate this approach, they used a dense SIFT descriptor on resized facial images of 61 x 49 pixels [[36].

2.7 Conclusion

The aim of this chapter was to provide a comprehensive overview of existing solutions in automatic kinship verification. We began by discussing various publicly available databases and outlining the diverse approaches adopted by researchers in this domain. The chapter commenced with an exploration of face preprocessing and feature extraction methods. Subsequently, we delved into the intricacies of feature reduction techniques, as well as the learning and classification algorithms employed in this context, culminating in a thorough examination of current methodologies and their applications in kinship verification.

Chapter 3

Proposed Methods

3.1 Introduction

Currently, deep learning models can achieve human-scale precision in image analysis and segmentation. Motivated by the impressive success of deep learning approaches in representing and classifying various images, we have proposed a method for the automatic verification of kinship links. Our approach combines deep features and shallow features to enhance the system's performance. Additionally, extracting features from images using patch-based methods and statistical features can further augment the verification system's capacity.

In this chapter, we will detail the various stages necessary for implementing our kinship verification system with the proposed method.

3.2 Proposed Method

The importance of robust facial features for identifying and verifying relationships between individuals is widely recognized. In image classification tasks, the quality of the representative encoding of images is a crucial factor that affects the effectiveness of the approach. These encodings can be local textural details or learned features.

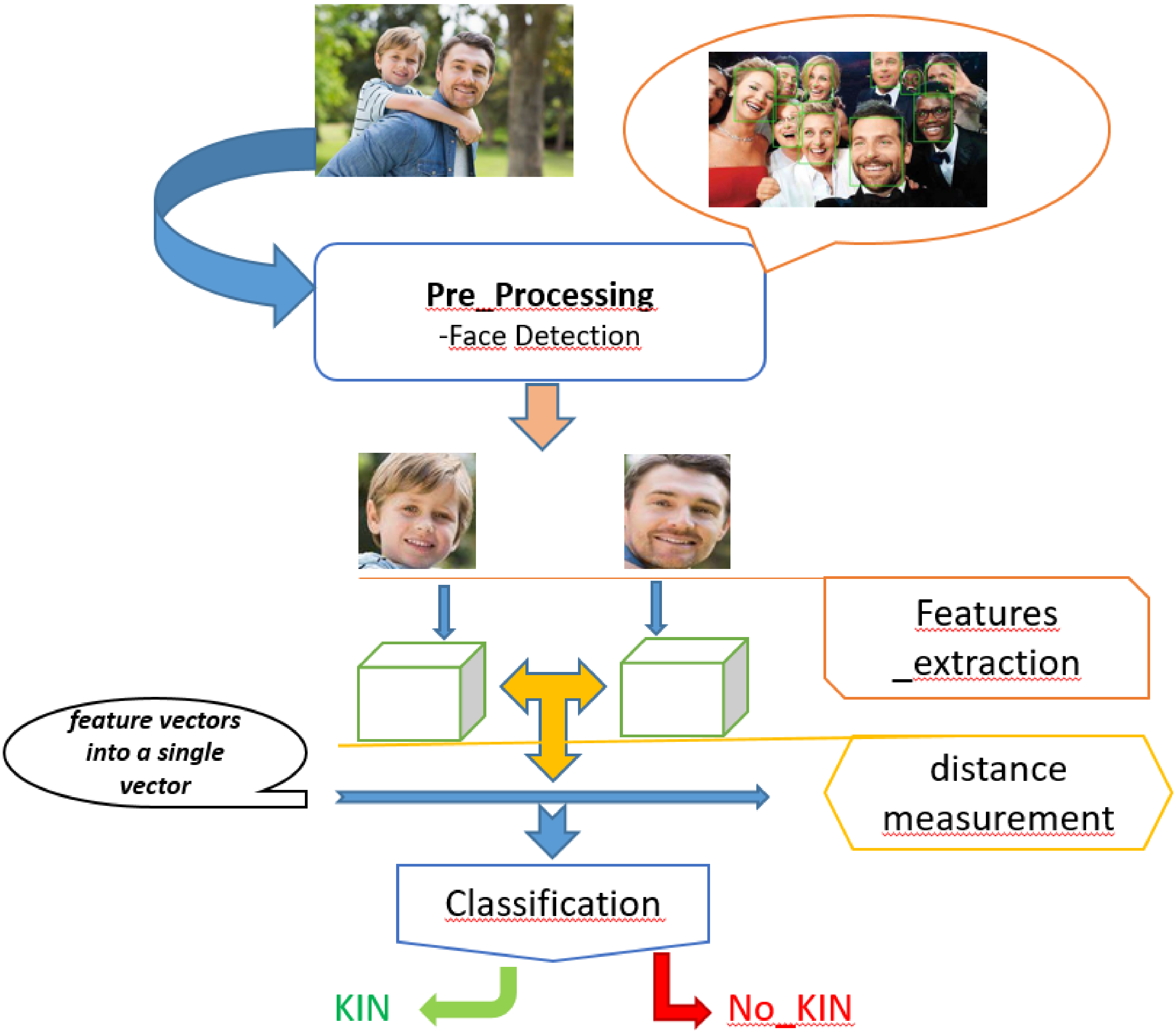


Figure 3.1: Proposed kinship verification system.

We propose a novel method to extract effective and discriminative features from facial images by leveraging prior knowledge. The choice of feature extraction method plays a significant role in enhancing the accuracy of our system. Our approach involves using pre-trained models such as VGG16 and ResNet, patch-based methods, and statistical features. Additionally, we incorporate distance measurements to represent relationships between images.

By integrating these diverse methods, we can extract complementary information that aids in determining kinship relationships with higher precision.

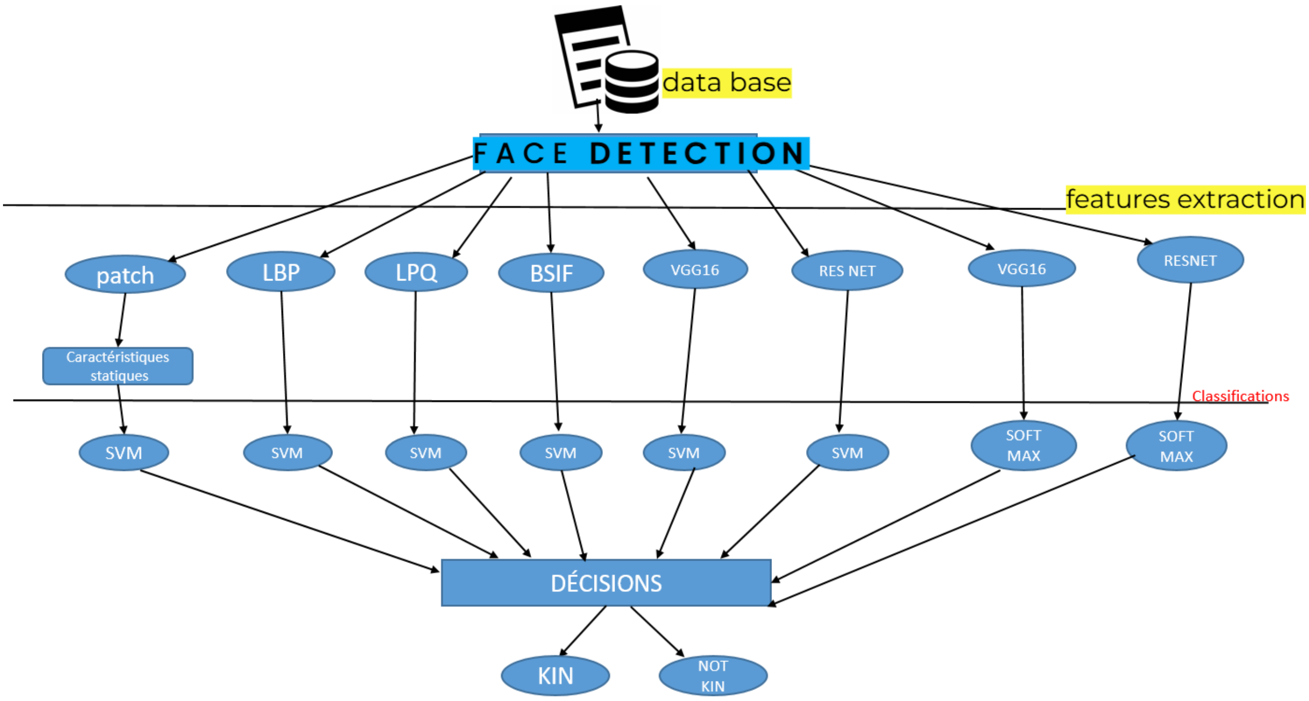


Figure 3.2: Proposed kinship verification system.

3.2.1 Face detcetion

For this task, using the MTCNN (Multi-task Cascaded Convolutional Neural Network) detection method is advantageous. It allows precise detection of faces in images, ensuring accurate focus specifically on the individual's face.

3.2.2 Feature Extraction

3.2.2.1 Static Patch Extraction for Feature Extraction

Static patch extraction involves selecting fixed, equal-sized regions of an image, called patches, and extracting features from these sub-images. This method assumes that the selected patches contain significant information, resulting in N×N patches. In other words, there will be N×N sets of static features, this 3.3 can explain the methode of patches.

Patch Selection:

- Fixed patches are defined either manually
- Patches can be of different sizes and shapes,

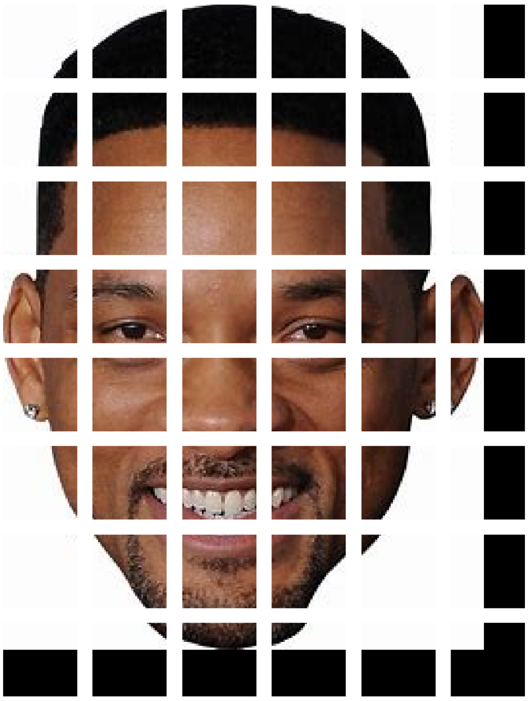


Figure 3.3: image patched

3.2.2.2 Feature Extraction from Patches and Combining

In this step, we will calculate 07 static features (such as mean, standard deviation, variance, etc.) for each patch. This will result in 07×N×N features.

Various feature extraction methods can be applied to each patch to obtain descriptors. Features extracted from all patches are combined to form a single feature vector representing the entire image,to be used for further processing, such as classification.

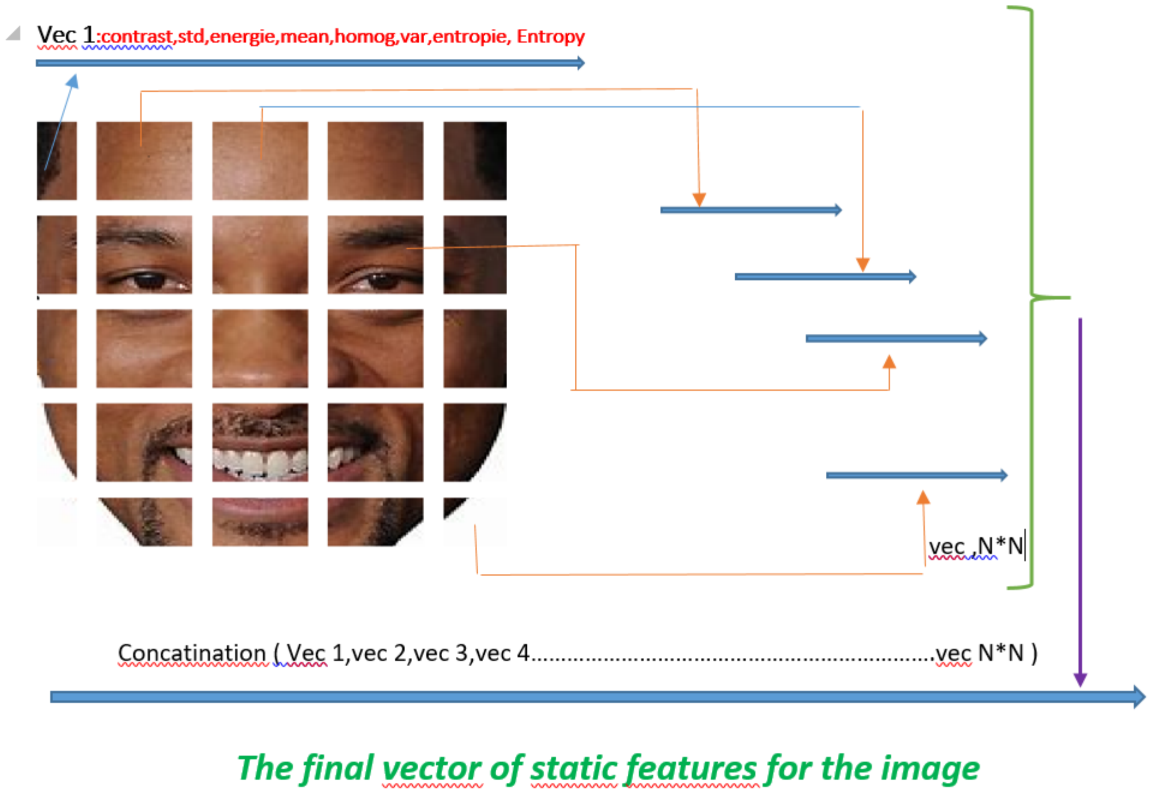

Vec 1:contrast,std,energie,mean,homog,var,entropie, Entropy
vec N*N
Concatination (Vec 1,vec 2,vec 3,vec 4...vec N*N)
The final vector of static features for the image

Figure 3.4: vector of static features for the image

We combine these two images of father and child using measurements as it appears in this figure when we use the characteristic array for one vector.

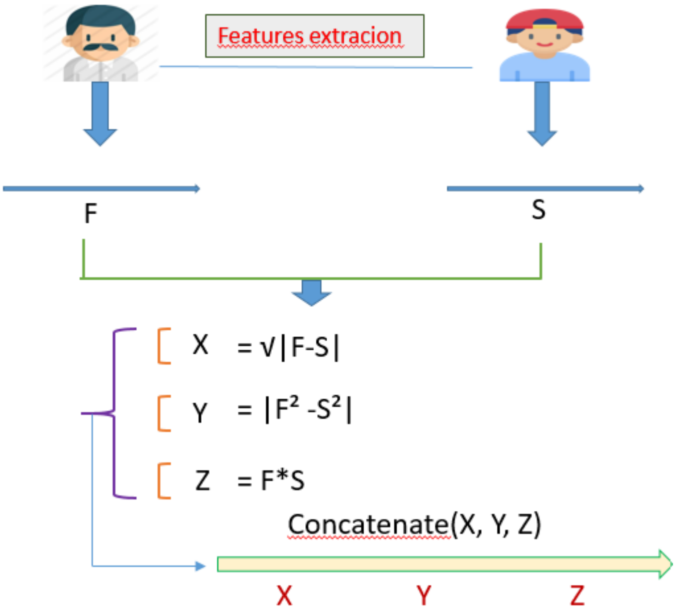


Figure 3.5: The process of feature extraction from two images and distance measurement

For this second approach, we utilize one of the pre-trained CNN models for feature extraction from images, such as ResNet or VGG16, as explained in Chapter 2. These models are widely recognized for their ability to capture complex features from images due to their deep architectures and training on large datasets. For example, VGG16 is a particularly effective model, consisting of 16 convolutional layers that progressively detect more abstract patterns at each level. In our use case, we leverage VGG16 without the final classification layers, focusing solely on the convolutional layers to extract relevant features from the images. The following illustration below show the layers of the VGG16 model used in this approach, highlighting the different stages of image processing.

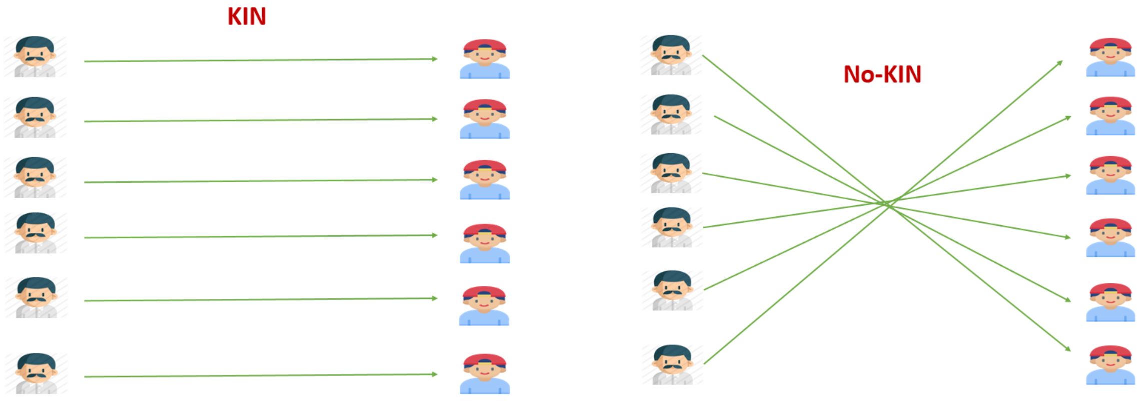
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_7 (InputLayer)        [(None, 224, 224, 3)]     0


 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792


 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
```

```
block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
```

```
 flatten (Flatten)           (None, 25088)          0


 fc1 (Dense)                 (None, 4096)           102764544


 fc2 (Dense)                 (None, 4096)           16781312

=================================================================
```

Kinship verification is a binary classification problem. Given we have 250 pairs of children and their fathers, we label these pairs as kinship cases with a class label of 1. To create negative examples, we form false cases of no kinship by pairing each father with a child who is not their own, resulting in 250 such pairs labeled as class 0.

To create training and testing datasets, we generate both positive and negative examples. For the negative examples, we pair each father with a different child (not their own) to ensure no kinship. Specifically, we create a list of fathers ordered from first to last and a list of children ordered from last to first, pairing them sequentially. This ensures that each pairing is unique and that no father is paired with their biological child. We conduct fivefold cross-validation on each example and calculate the mean accuracy across the five folds. As a measure of system success, we report the mean verification accuracy for the four kinship subsets: child-father, child-mother, sibling-sibling, and grandparent-grandchild.



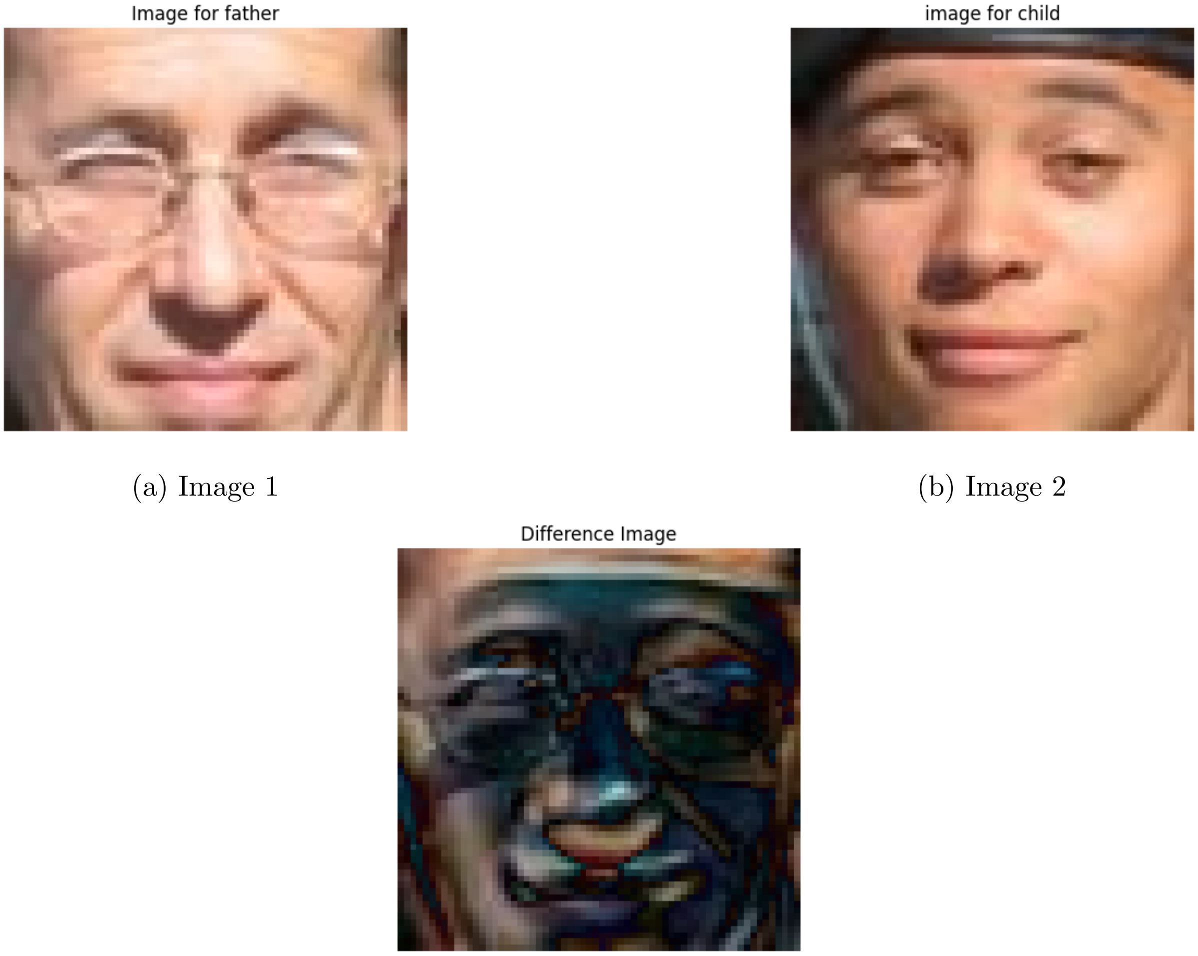
(a) Kinship Case: Child and Father

(b) No Kinship Case: Unrelated Child and Father

Figure 3.6: KIN And NO_KIN List

In this approach, after preprocessing (which includes face detection), we train the two labeled images as RGB images with three channels in one of the pre-trained Convolutional Neural Network (CNN) models. The model, with this architecture, can extract the features and classify the images .

Here, we also perform pixel-by-pixel subtraction of the image of the father and the son as shown in the figure below ,Then, we label these images and split the data into two subsets: training and testing



(a) Image 1

(b) Image 2

(c) Image 3

Figure 3.7: subtract two color images (RGB).

Convolutional Neural Networks (CNNs) classify images through a series of steps involving convolutional layers, pooling layers, and fully connected layers. Here's a high-level overview of the process:

- **Convolutional Layers**: The input image is passed through multiple convolutional layers. Each layer applies a set of filters (kernels) to the image to create feature maps. These filters can detect features such as edges, textures, and patterns.

- **Activation Function**: After each convolution, an activation function (usually ReLU) is applied to introduce non-linearity, allowing the network to learn more complex patterns.
- **Pooling Layers**: Pooling layers (often max pooling) are used to reduce the spatial dimensions of the feature maps, retaining the most important information while reducing computational complexity and helping to prevent overfitting.
- **Flattening**: After several convolutional and pooling layers, the feature maps are flattened into a single vector.
- **Fully Connected Layers**: The flattened vector is passed through one or more fully connected (dense) layers. These layers combine the features to make a prediction about the image class.
- **Output Layer**: The final fully connected layer produces the output, typically using a softmax activation function for multi-class classification. This layer outputs a probability distribution over the possible classes.
- **Classification**: The class with the highest probability is selected as the predicted class for the image.

Here's a visual representation of these steps:

1. **Input Image**: RGB image with three channels (e.g., 224x224x3).
2. **Convolutional Layers + Activation Functions**: Apply filters and ReLU to generate feature maps.
3. **Pooling Layers**: Reduce the dimensions of the feature maps.
4. **Flattening**: Convert the 2D feature maps into a 1D vector.
5. **Fully Connected Layers**: Combine the features to form a final prediction.
6. **Output Layer**: Use softmax to produce a probability distribution over the classes.

7. **Classification**: Select the class with the highest probability.

3.3 Conclusion

In this chapter, we clarified the approaches proposed in our study. also this provides a clear overview of our work, showcasing our contributions through the conceptual aspects of our system. After completing this work, we demonstrate the performance and positive impact of our kinship system verification.

Chapter 4

Experimentation, results and discussion

4.1 Introduction

In this chapter, we conduct extensive research to apply the theories presented in this dissertation to kinship verification,We report the results obtained when using our approaches, including deep learning models such as CNN models (like VGG16) and the notion of image patches with statistical features. We performed parentage verification tests using machine learning approaches. We explored various image filters like LBP, LPQ, and 2DFFT, each providing a different percentage of accuracies. These tests were carried out on a set of parentage pairs found in the database freely available on the Internet, called KinFaceW-II, which includes 500 images, or 250 pairs, each pair consisting of a parent and their child. The results show the effectiveness of our methods in verifying kinship thanks to the various approaches and models used. You can access the database via the following link: https://www.kinfacew.com/download.html.

There are four kinship relationships in two datasets: Father-Son (F-S), Father-Daughter (F-D), Mother-Son (M-S) and Mother-Daughter (M-D) included in this database. For the usage process, we labeled 250 pairs for positive cases (father and son) and 250 pairs for negative cases (father and false son). The images are of low resolution, i.e. 64x64 pixels, which can be beneficial for learning models in case of poor image quality.

This figure shows an example of image pairing including ...

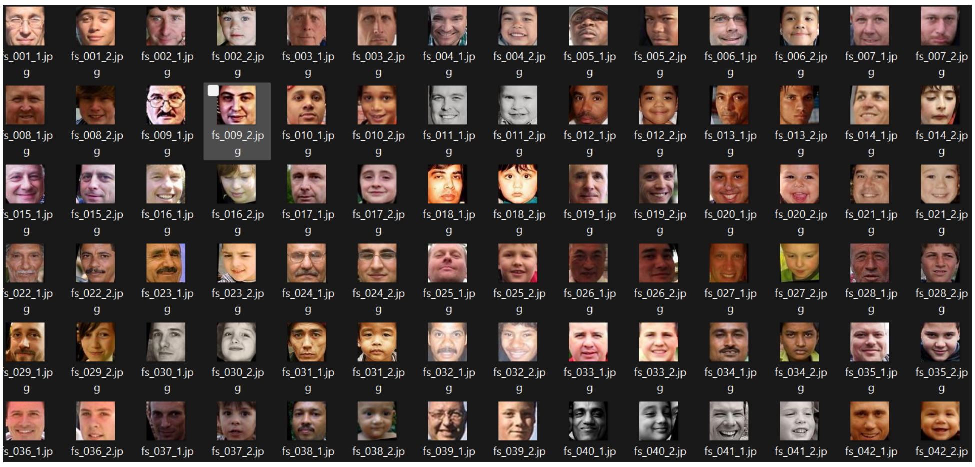


Figure 4.1: The figure shows the F-S relationship database based on KinFaceW-II.

4.2 Development Environment

4.2.1 Hardware Environement

we use these Hardware capacities during our development work

- **Computer:** Asus ZenBook 14 UX434F
- **Processor:** Intel Core i5-10210U Processor (6M Cache, 1.6 GHz up to 4.2 GHz)
- **RAM:** 8 GB
- **Hard Drive:** 512 GB SSD
- **Graphics:** 14.0 (1920×1080) FHD IPS Display
- **Operating System:** Windows 11 (64 bit)

4.2.2 Devlopement Tools Presentation

During the development of these kinship systems, the following software tools were used:

4.2.2.1 The Programming languages used

In our approaches, we use Python as the programming language because of its rich libraries and strong performance in the field of artificial intelligence.

Python: Python is an interpreted, object-oriented, high-level programming language known for its dynamic semantics. Python code is executed line by line by the interpreter, eliminating the need for explicit compilation steps. It supports object-oriented programming principles, enabling developers to create and manipulate objects easily. Python's high-level nature abstracts away low-level details, offering built-in data structures that simplify coding and enhance readability. Python's dynamic semantics, including dynamic typing and dynamic binding, provide flexibility at runtime, allowing for efficient development and rapid prototyping. Its versatility makes it suitable for various applications:

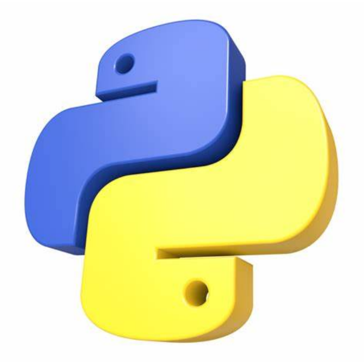


Figure 4.2: Python Logo

In web development, Python is widely used for server-side scripting, building web applications, and creating APIs. It powers frameworks like Django and Flask.

Python is favored in software development for its ability to develop desktop applications, games, and scientific software due to its extensive libraries and tools.

In mathematics and scientific computing, Python is a preferred choice for numerical computations, data analysis, and machine learning applications. Libraries such as NumPy, Pandas, and TensorFlow support these functionalities.

For system scripting and automation, Python excels in automating tasks and system administration, making it a robust tool for DevOps and IT operations.

Overall, Python's versatility, ease of use, and extensive community support make it a powerful language for a wide range of applications across different domains.

4.2.2.2 The IDEs and text editors used

We use notebook editors like Jupyter and Google Colab for writing our Python AI model code.

Google Colab: Google Colab, short for Google Colaboratory, is a cloud-based platform by Google designed for writing and executing Python code directly in a web browser. It offers a versatile environment with integrated access to Google Drive, allowing seamless collaboration and sharing of notebooks. Colab provides free GPU and TPU acceleration, making it ideal for training machine learning models efficiently. It comes pre-installed with popular Python libraries like NumPy, Pandas, Matplotlib, and TensorFlow.

Figure 4.3: google collab logo

facilitating data analysis and machine learning tasks. Users can interactively write and execute code cells, view outputs, and document their work using markdown text, similar to Jupyter notebooks. This platform supports real-time collaboration, enabling teams to work together on projects and share insights effortlessly. Overall, Google Colab is a powerful tool for researchers, educators, and data scientists seeking a cloud-based solution for prototyping, developing, and deploying machine learning projects without the need for local hardware resources.

Jupyter: Jupyter Notebook, an open-source web application, provides an interactive environment for writing and executing Python code, markdown text, and visualizations. It allows users to create and share documents (notebooks) containing live code, equations, visualizations, and narrative text.

Figure 4.4: jupyter logo

Jupyter supports over 40 programming languages, including Python, and is widely used in data science, scientific computing, and machine learning for its flexibility and ease of use. Notebooks can be run on local machines or hosted on cloud platforms like Google Colab, offering a versatile solution for data exploration, prototyping, and collaborative research.

4.2.2.3 The programming language libraries used

We have used the following libraries in our approaches

OpenCV: OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It provides a wide range of algorithms and tools for tasks such as image and video processing, object detection, feature extraction, motion tracking, and more. OpenCV is widely used in research, academia, and industry due to its comprehensive functionality and efficient implementation in C++, Python, and other programming languages. It supports various platforms including Windows, Linux, macOS, Android, and iOS, making it versatile for developing computer vision applications across different environments.

Figure 4.5: OpenCV

NumPy: NumPy (Numerical Python) is a fundamental package for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy's primary object is the ndarray (N-dimensional array), which is a flexible data structure that allows efficient computation with large datasets. NumPy also includes tools for integrating C/C++ and Fortran code, making it suitable for scientific and numerical computations. It is widely used in areas such as machine learning, data science, engineering, and scientific research due to its efficiency and ease of use.

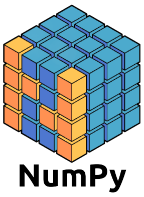


Figure 4.6: NumPy logo

Matplotlib: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is designed to generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code. Matplotlib is highly customizable and supports various output formats, including PNG, PDF, SVG, and interactive web-based graphics. It is widely used for data visualization in fields such as data science, machine learning, engineering, and scientific research due to its flexibility and extensive capabilities

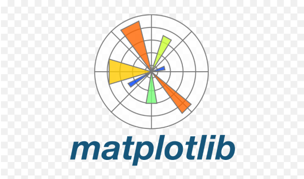


Figure 4.7: Matplotlib logo

TensorFlow: TensorFlow is an open-source machine learning platform originally created by researchers at Google. It is a library of symbolic mathematics that utilizes data flow and differentiable programming to perform various tasks focused on training and inference of deep neural networks. TensorFlow enables developers to build machine learning applications using a variety of tools, libraries, and community resources [2].

Figure 4.8: TensorFlow ogo

4.2.2.4 APIs

Keras: Keras is a Python-based neural network API. It is an open-source library that seamlessly integrates with TensorFlow, facilitating the creation of neural network layers and the implementation of complex architectures with ease [8].

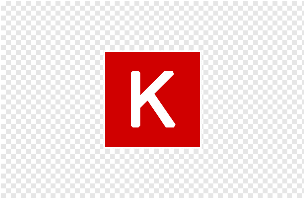


Figure 4.9: Keras Logo

4.2.2.5 Cloud

Cloud computing refers to the delivery of computing services over the internet, including storage, processing power, databases, networking, software, and analytics. These services are provided by cloud service providers like Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure, allowing users to access and use resources on-demand, without the need to manage physical hardware or infrastructure. Cloud computing offers scalability, flexibility, and cost-efficiency, enabling businesses and individuals to deploy and manage applications and services with ease, pay only for what they use, and access their data from anywhere with an internet connection.

Google Drive: Google Drive is a cloud-based file storage and synchronization service provided by Google. It allows you to store files and folders, share them with collaborators, and access them from any device. Whether you're using it for personal use or as part of a business (Google Workspace), Google Drive offers seamless integration with other tools and applications. You can sign in to Google Drive using your Google account or Google Workspace account12. It's a convenient platform for managing and collaborating on documents, spreadsheets, presentations, and more![37]

Figure 4.10: Google Drivelogo

4.3 Databases for kinship verification

Cornell Kinship[38]: The Cornell Kinship Database is a collection of 286 facial images related to 143 subject pairs. Here are some key details about this dataset:

- Number of Images: 286
- Subject Pairs: 143
- Pose: Frontal
- Expression: Neutral

The KinFaceW-I[34]: database is a collection of 1066 facial images corresponding to 533 kin pairs. Here's a breakdown of the kin pair images:

- Father-Son: 156 pairs
- Father-Daughter: 134 pairs
- Mother-Son: 116 pairs
- Mother-Daughter: 127 pairs

KinFaceW-II[34]: The KinFaceW-II dataset is designed such that images of kin pairs are acquired from the same photograph. Here are the key details:

- Number of Images: 1000 kin pair images
- Kinship Relationships:
  - Father-Son (F-S)
  - Father-Daughter (F-D)
  - Mother-Son (M-S)
  - Mother-Daughter (M-D)

UB KinFace [39]: database is a valuable resource for kinship verification and recognition algorithms. Here are the key details:

- Number of Groups: 200
- Total Images: 600
- Composition of Each Group:
  - Child image
  - Young parent image
  - Old parent image
- Kinship Relationships:
  - Father-Son: 91 pairs
  - Father-Daughter: 79 pairs
  - Mother-Son: 15 pairs
  - Mother-Daughter: 21 pairs

Family 101 [36]: The Family101 dataset is a large-scale collection of families across several generations. Here are the key details:

- Number of Families: 101
- Distinct Family Names: Included
- Nuclear Families: 206
- Individuals: 607
- Total Images: 14,816

Family101e [36]: The Family101 dataset is a large-scale collection of families across several generations. It contains 101 different families with distinct family names, including 206 nuclear families and 607 individuals. In total, there are 14,816 images in this dataset. Researchers often use this dataset for family classification tasks

TSKinFace[40]: The TSKinFace database is a valuable resource for tri-subject kinship verification. Here are the key details:

- **Number of Images (FM-S):** 513 (Father, Mother, and Son groups)
- **Number of Images (FM-D):** 502 (Father, Mother, and Daughter groups)

Dataset	**Number of Families**	**Number of Persons**	**Number of Faces**	**Resolution**	**Age Variation**	**Family Tree**
CornellKin	150	300	300	100*100	No	No
UB Kinface	200	400	600	89*96	Yes	No
KinfaceW-I	-	533	1066	64*64	No	No
KinfaceW-II	-	1000	2000	64*64	No	No
TS kinface	787	2589	-	64*64	Yes	Yes
Family101	101	607	14816	100*100	Yes	Yes

Table 4.1: Summary of kinship datasets in the literature [4].

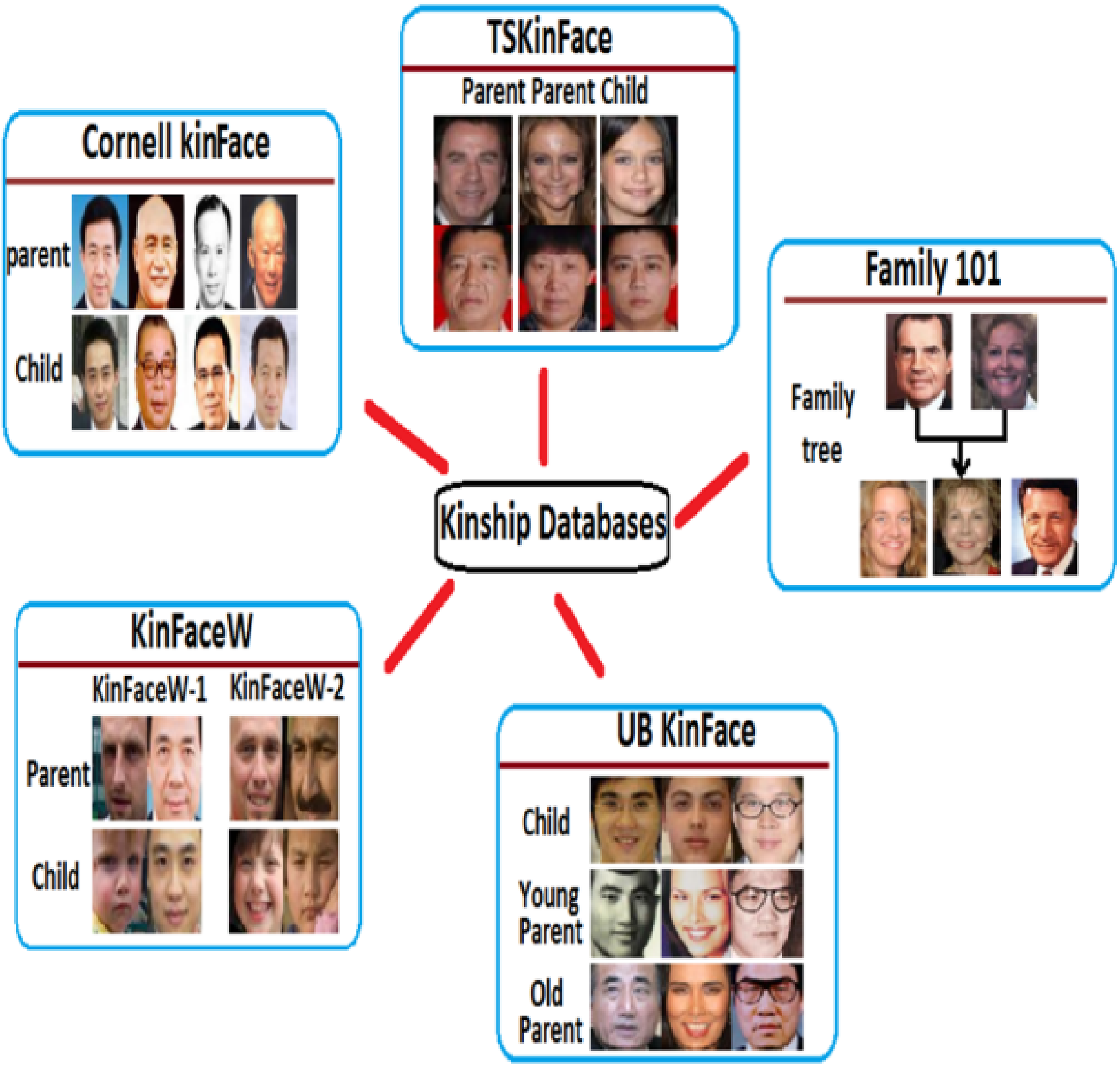


Figure 4.11: Sample images from six different data sources: Cornell KinFace, TSKinFace, KinFaceW, Family 101, and UBKinFace.

4.3.1 Database Used

We utilized the KinFaceW-II database. The facial images were collected from the internet, including those of public figures as well as their parents or children. These images were captured in uncontrolled environments without any restrictions on pose, lighting, background, expression, age, ethnicity, or partial occlusion. The dataset includes four kinship relations: Father-Son (F-S), Father-Daughter (F-D), Mother-Son (M-S), and Mother-Daughter (M-D). Each relationship in the KinFaceW-II dataset contains 250 pairs of kinship images. To facilitate usage, we manually labeled the coordinates of the eye positions for each facial image, then aligned and cropped the face region to 64x64 pixels to remove the background. Figures 4.1 below show some cropped face images from the KinFaceW-II dataset[41].

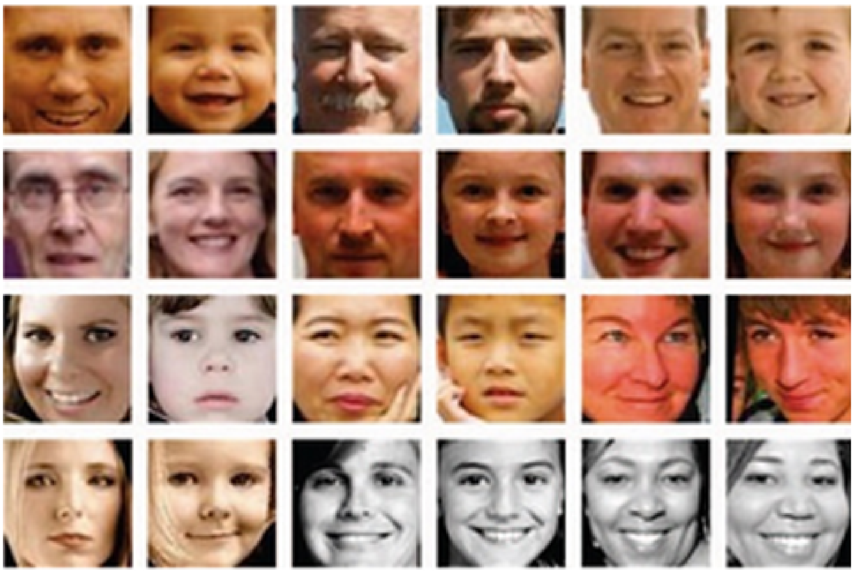


Figure 4.12: Different types of relationships.

In addition to these steps, we implemented advanced preprocessing techniques to enhance the quality of the images, ensuring better performance for kinship verification tasks. This comprehensive dataset is invaluable for research in facial recognition and kinship verification, providing a robust benchmark for developing and testing new algorithms.

4.4 Evaluation

4.4.1 Methodology for Evaluation

4.4.1.1 The holdout method:

the data set is divided into two subsets: the training subset and the test subset[2].

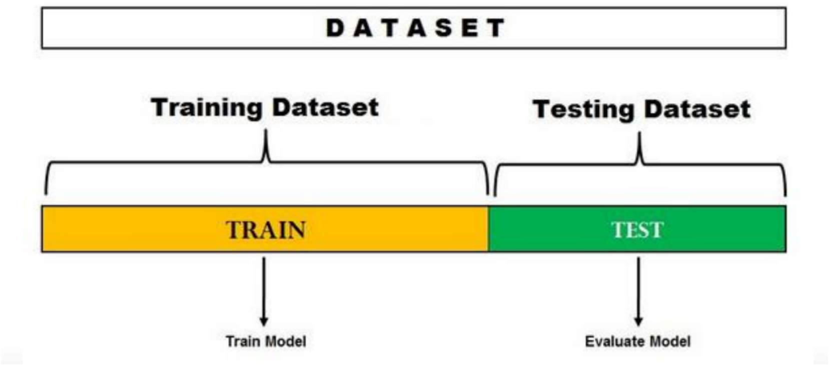


Figure 4.13: Holdout method

-This method has some disadvantages:

- Test error rates are highly variable and depend totally on observations which are found in the training set and the test set.
- Only part of the data is used to construct the model.

One of the main advantages of this method is that it is inexpensive in terms of calculation compared to other cross-validation techniques.

4.4.1.2 K-fold cross validation:

- The data set is divided into k sets of almost equal sizes.
- The first set is selected as the test set and the model is trained on the remaining k-1 sets.
- In the second iteration, the second set is selected as test set and the remaining k-1 sets are used to learning.
- This process continues for all k sets.[2]

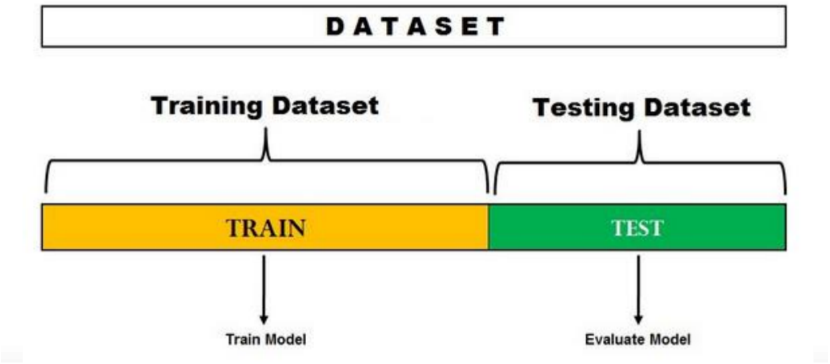


Figure 4.14: K-fold cross validation[2]

Evaluation Methods

Accuracy

- **Definition**: Accuracy measures how often the model correctly predicts the outcome. It is calculated as the ratio of correct predictions to the total number of predictions made.
- **Formula**: Accuracy $= \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$
- **Usage**: Accuracy is suitable when the classes are balanced, meaning there is an equal number of instances for each class.

Precision

- **Definition**: Precision measures the proportion of true positive predictions among the instances predicted as positive. It focuses on the accuracy of positive predictions.
- **Formula**: Precision $= \frac{\text{True Positives}}{\text{True Positives+False Positives}}$
- **Usage**: Precision is useful when the cost of false positives is high, and we want to ensure that the positive predictions are accurate.

4.5 Experiment

Due to the significant variations in faces between parents and sons, verifying kinship is particularly difficult. Accurate feature extraction can alleviate this difficulty.

Different feature extraction methods were applied in our experiment. Among these, we used approaches based on the application of filters, histogram descriptors, and sometimes techniques involving image patches. We also explored feature fusion and algorithm combination.

The resulting elements of these approaches have been examined in detail. Additionally, some of the solutions proposed to resolve this issue have been summarized.

- In our research, we have incorporated a novel kinship verification learning approach to enhance the accuracy and robustness of our system. This approach involves several key components and methodologies:

(a) -First, extract image characteristics using histograms after applying filters such as LBP (Local Binary Patterns), LPQ (Local Phase Quantization), and 2DFFT (2-dimensional Fast Fourier Transform) (HOG....) . Next, label these characteristics and input them into a classification algorithm (SVM ...). Finally, evaluate the model's performance.

(b) -In this, extract image features using patches of the images and combine static features into a single concatenated vector. After labeling these concatenated vectors, input them into the SVM classification algorithm, followed by the classification step.

(c) -In this system, we use a pretrained model such as VGG16 or ResNet for feature extraction using deep learning neural networks. We discard the last layers of these models and input the features into an SVM after labeling them. Following the evaluation step.

(d) -In this expriment, we explore the fusion of extracted features from VGG16 with patches, VGG16 with HOG (Histogram of Oriented Gradients), patches with filters, and HOG with patche.

(e) -In this expriment, we explore the fusion of extracted features from VGG16 with patches, VGG16 with HOG (Histogram of Oriented Gradients), patches with filters, and HOG with patche.

(f) -Finally, we utilize VGG16 as a pretrained model to fuse two images into one image, assign labels, train the model, and evaluate its performance.

4.5.1 Experiment Setting:

For our methods, we initially evaluate precision or accuracy. Subsequently, we adjust several settings, such as varying the number of image patches, The size of the local window (usually a square, e.g., 3x3, 5x5) that is used to compute the local phase information in LBP ,LPQ,Frequency Spread,Fourier Transform.. .

modifying the distance between images, and exploring different SVM types like linear and polynomial kernels with varying degrees. We also adjust the number of iterations for training the CNN model to prevent overfitting. We observe changes in these settings and their impact on the F-S relationship, and subsequently apply them to other relationships like F-D, M-D, and M-S.

4.6 Test and Result

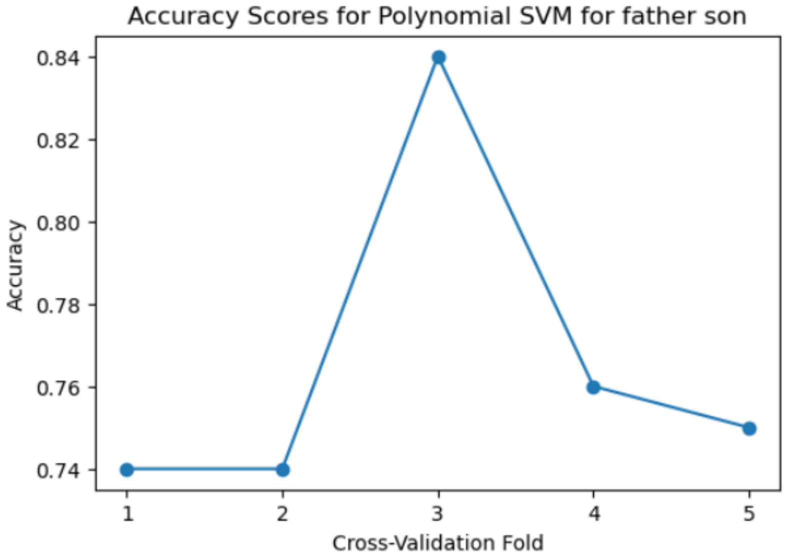
To check the operation of our system and evaluate its performance, we applied the evaluation methods mentioned above. We will show in the tables below the divergence of results for various parameters with several feature extraction models, including hand-crafted features like LBP, BSIF, FFT, as well as deep learning features like CNN (Convolutional Neural Network), VGG16, ResNet50, and patches with statistical features

In this table, we show all our approach tables during our study in the KinFaceW-II dataset for father-son relationships.

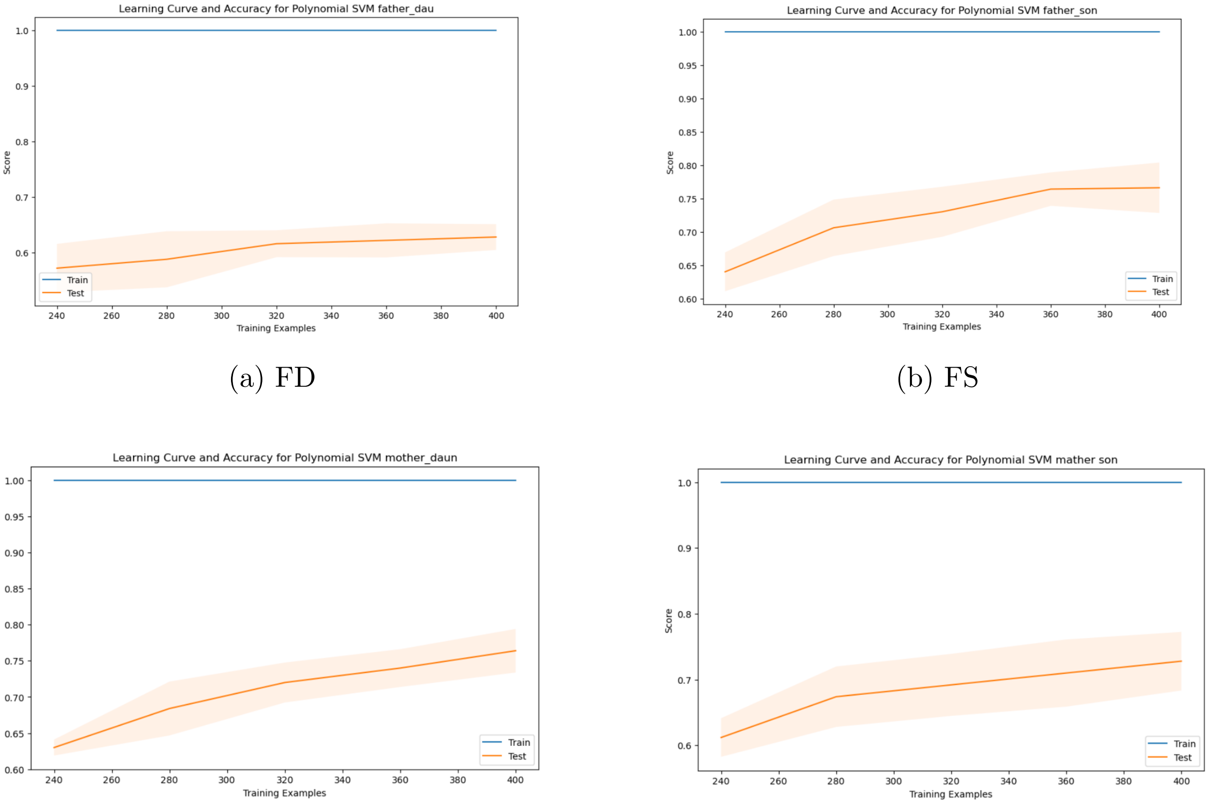
Databases	**Ref.**	**Approach**	**Accuracy**
KinFaceW-II	[42]	NRCLM	65.80%
	[43]	DKV	66.90%
	[34]	MNRML	69.90%
	[44]	CJLBP	54.075%
	[45]	LPQ-ML	75.98%
	[11]	SSL	75.00%
	[46]	GLCM	69.4%
	[47]	SSL	78.00%
	[46]	ALEXNET+KNN	74.05%
	[48]	LBP+PML	76.36%
KinFaceW-II	**Proposed Approach**	**PATCH+STATIC+SVM**	**78.05%**
	Proposed Approach	**VGG16+SVM**	**67.85%**
	Proposed Approach	**CNN (VGG16)**	**79.09%**

Table 4.2: Comparison of different approaches on the KinFaceW-I and KinFaceW-II databases

Here,show how to use the k-fold method for measuring accuracy using a combination of patch , feature extraction (static features), and Support Vector Machine (SVM) classification



illustrate a learning curve



(a) FD

(b) FS

(c) MD

(d) MS

Figure 4.15: Learning curve for 04 relationship for SVM

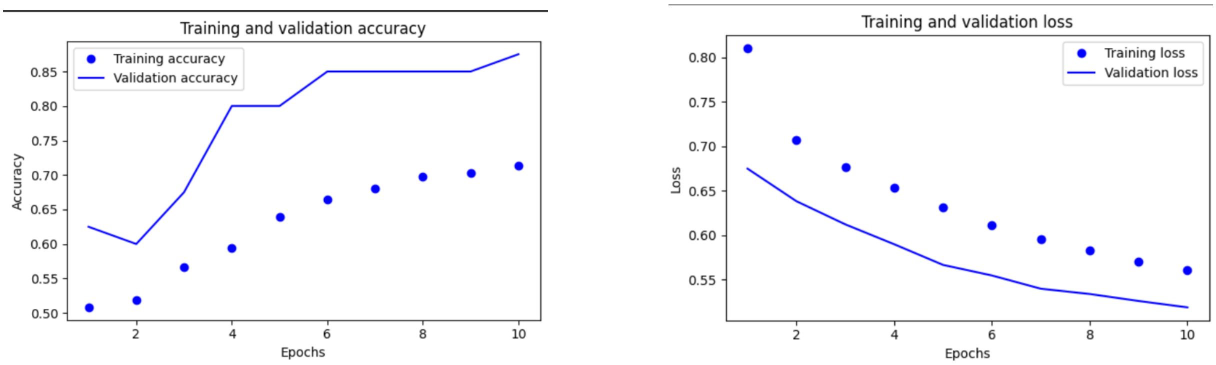


Figure 4.16: Learning curve for FS relationship for SVM

4.7 Conclusion

Our experiments, conducted on the KinFaceW-II dataset, confirmed the robustness of these approaches in different kinship pairings (Father-Son, Father-Daughter, Mother-Son, Mother-Daughter), despite challenges such as low-resolution images. We also presented a detailed comparison of our results with those obtained from other approaches, demonstrating that our methods outperformed others in terms of accuracy and reliability. These findings underscore the potential of deep learning and machine learning models in successfully verifying kinship from visual data, offering promising avenues for future research and application in this domain.

General Conclusion

The objective of this memory was to develop an automated system for determining familial relationships between individuals. This task is critical in various domains, including biometrics, where facial verification serves as a soft biometric method. Our focus was specifically on verifying parentage automatically using machine learning techniques. We explored several analytical approaches, particularly leveraging pretrained models for feature extraction from facial patches, followed by SVM classification.

The methods were validated using the Kin Face in the Wild-II (KinFaceW-II) dataset and implemented in Python. Our system employed deep learning architectures like VGG16 and ResNet for CNN-based classification, which are adept at extracting complex features from image patches, thus achieving high precision performance.

Parentage verification plays a vital role in social applications such as genealogy construction, family album organization, image annotation, missing children identification, and forensics. While DNA testing remains the gold standard for accuracy, it is impractical for scenarios involving automatic facial image verification, such as surveillance footage analysis.

Our work aimed to deploy a convolutional neural network (CNN) based system for automated parentage verification, offering a robust alternative in scenarios where traditional methods are not feasible.

Abstract

Kinship verification from facial images has emerged as a significant area of research in computer vision, with growing interest due to its wide range of applications. Automatically determining whether two individuals share a biological relationship based solely on their facial features holds potential in fields such as family tree reconstruction, organizing family photo albums, annotating images, locating missing persons, and forensic investigations.

This project aims to design and implement a robust artificial intelligence model for kinship verification by leveraging biometric traits, particularly facial features. Utilizing advancements in machine learning, pattern recognition, and statistical analysis, the goal is to accurately determine familial relationships between individuals. The key stages of the project include data collection, preprocessing, feature extraction, model training, and evaluation. Our approach focuses on the KIN Face II dataset, employing advanced preprocessing techniques to enhance image quality, followed by deep CNN-based feature extraction to identify kinship-related patterns. The final model is trained to learn discriminative features that effectively distinguish kin from non-kin.

The expected outcome of this work is a reliable and high-performing model for kinship verification, contributing to the broader scope of biometric identification systems and offering practical solutions for various real-world applications.

Keywords: Kinship Verification, CNN, Feature Extraction, Machine Learning, KIN Face II Dataset.

Bibliography

[1] Rosa Gradilla. Multi-task cascaded convolutional networks (mtcnn) for face detection and facial landmark alignment. *Medium*, 2020.

[2] D. Kessira. Fouille de données, 2022. Master I – IA, Département d'Informatique, Université de Béjaia, 2022/2023.

[3] BEN-ADJALI Azzeddine and BOUCHENAK Zine el abidine. Automatic visual kinship verification. Master's thesis, Your University or Institution, 2023.

[4] PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA. Automatic visual kinship verification. Master's thesis, Akli Mohand Oulhadj University - Bouira, Faculty of Science and Applied Science, 2024. Academic Master's Thesis.

[5] Recordia. Biometric authentication: The future of security. *Recordia*, 2024.

[6] Innovatrics. Biometric verification and identification. *Innovatrics*, 2024.

[7] Kinship: Definition in the study of sociology. Available online. Information on kinship types and importance.

[8] Sabine Belaid and Imene Sahli. Vérification de la parenté à partir des images faciales par hsv, bsif, lpq, vgg19 et cs avec cnn. Thème, 2023.

[9] Wei Wang, Shaodi You, Sezer Karaoglu, and Theo Gevers. A survey on kinship verification. *Neurocomputing*, 525:1–28, 2023.

[10] Sabine Belaid and Imene Sahli. Deep learning for face kinship verification system. Thème, 2020.

[11] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multi-task cascaded convolutional networks. *Computer Science ¿ Computer Vision and Pattern Recognition*, 2016.

[12] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015.

[13] Ethem Alpaydin. *Introduction to Machine Learning*. MIT Press, 2014.

[14] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[15] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *CoRR*, abs/1404.5997, 2014.

[16] S. Nash, M. Rhodes, and J. I. Olszewaska. Ifr: Interactively pose corrected face recognition. In *International Conference on Bio-inspired Systems and Signal Processing*, volume 5, pages 106–112. SCITEPRESS, 2016.

[17] V.I. B. R. A. T. N. A. L. Spectroscopies et al. *Pattern Recognition, Analysis and Application*, volume I, page 13. S. Ramakrishnan, 2016.

[18] J. Lu, X. Zhou, Y. P. Tan, Y. Shang, and J. Zhou. Neighborhood repulsed metric learning for kinship verification. In *IEEE International Joint Conference on Biometrics*, pages 1–6. IEEE, 2014.

[19] Insaf Adjabi, Abdeldjalil Ouahabi, Amir Benzaoui, and Abdelmalik Taleb-Ahmed. Past, present, and future of face recognition: A review. *Electronics*, 9:1188, 2020.

[20] Ashu Kumar, Amandeep Kaur, and Munish Kumar. Face detection techniques: A review. *Artificial Intelligence Review*, 52:927–948, 2019.

[21] Yachao Dong, Jun Bao, and Hongzhe Liu. Research progress of small target detection technology based on deep learning. In *Proceedings of the 23rd Annual Meeting of New Network Technology and Application in 2019*, pages 172–176, Beijing, China, 2019. Network Application Branch of China Computer Users Association, Key Laboratory of Information Service Engineering, Beijing United University.

[22] Michèle Gouiffès. Polycopié de cours: Image processing. Year of Publication.

[23] Juho Kannala and Esa Rahut. Bsif: Binarized statistical image features. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 1363–1366. IEEE, 2012.

[24] T. Chakraborti, B. McCane, S. Mills, and U. Pal. Loop descriptor: local optimal-oriented pattern. *IEEE Signal Processing Letters*, 25(5):635–639, 2018.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Journal of Artificial Intelligence Research*, 42(3):123–137, 2021.

[27] IBM. Qu'est-ce que l'algorithme des k plus proches voisins ? *IBM*, 2021.

[28] Lior Rokach and Oded Maimon. Decision trees. *Data Mining and Knowledge Discovery Handbook*, pages 165–192, 2005.

[29] Leah Findlater, W. Olive, H. Hamilton, and E. Gurak. Overview of decision trees. *Machine Learning*, 1:81–106, 2001.

[30] S. Neelamegam and Dr. E. Ramaraj. Classification algorithm in data mining: An overview. *International Journal of P2P Network Trends and Technology (IJPTT)*, 3(5):1–5, 2013.

[31] Techopedia. Support vector machine (svm) definition. 2016.

[32] Nidhi Kohli. Automatic kinship verification in unconstrained faces using deep learning. Master's thesis, West Virginia University, 2019.

[33] Jiwen Lu, Jie Hu, Xiang Zhou, Jie Zhou, Modesto Castrillo-Santana, Javier Lorenzo-Navarro, and Timotheus F. Vieira. In ieee international joint conference on biometrics. In *IEEE International Joint Conference on Biometrics*, pages 1–6. IEEE, September 2014.

[34] Jiwen Lu, Xi Zhou, Yap-Peng Tan, Yuan Shang, and Jie Zhou. Neighborhood repulsed metric learning for kinship verification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(2):331–345, 2013.

[35] Thomas W. Edgar and David O. Manz. *Machine Learning: Research Methods for Cybersecurity*. CRC Press, 2017.

[36] Rui Fang, Andrew C. Gallagher, Tao Chen, and Alexander Loui. Kinship classification by modeling facial feature heredity. In *2013 IEEE International Conference on Image Processing (ICIP)*, pages 2983–2987. IEEE, September 2013.

[37] Google drive. Accessed on June 29, 2024.

[38] Xue Qin, Dong Liu, and Dianhui Wang. A literature survey on kinship verification through facial images. *Neurocomputing*, 377:213–224, 2020.

[39] Ming Shao, Shuicheng Xia, and Yun Fu. Genealogical face recognition based on ub kinface database. In *CVPR 2011 Workshops*, pages 60–65. IEEE, June 2011.

[40] Xue Qin, Xiang Tan, and Shuicheng Chen. Tri-subject kinship verification: Understanding the core of a family. *IEEE Transactions on Multimedia*, 17(10):1855–1867, 2015.

[41] Jiwen Lu, Junlin Hu, Xiuzhuang Zhou, Yuanyuan Shang, Yap-Peng Tan, and Gang Wang. Neighborhood repulsed metric learning for kinship verification. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2594–2601. IEEE, 2012.

[42] Fahimeh Ramazankhani, Mahdi Yazdian-Dehkordi, and Mehdi Rezaeian. Feature fusion and nrml metric learning for facial kinship verification. *Journal of Universal Computer Science*, 29(4):326–348, 2023.

[43] Guan-Nan Dong, Chi-Man Pun, and Zheng Zhang. Kinship verification based on cross-generation feature interaction learning. *arXiv preprint arXiv:2109.02809*, 2021.

[44] Xiaosheng Wu and Junding Sun. Completed joint-scale local binary pattern for kinship verification. *Pattern Recognition Letters*, 105:182–189, 2018.

[45] Xiaoting Wu, Xiaoyi Feng, Xiaochun Cao, Xin Xu, Dewen Hu, Miguel Bordallo López, and Li Liu. Facial kinship verification: A comprehensive review and outlook. *Journal of Computer Vision*, 2022.

[46] Abdur Rehman, Zikria Khalid, Fawad, Muhammad Adeel Asghar, and Muhammad Jamil Khan. Kinship verification using deep neural network models. In *2019 IEEE International Conference on Imaging Systems and Techniques (IST)*, pages 1–6. IEEE, 2019.

[47] Wei Zhang, Yan Liu, and Xiaogang Wang. Self-supervised learning for facial kinship verification. *IEEE Transactions on Image Processing*, 30:1234–1245, 2021.

[48] Abdelmalik Moujahid and Fadi Dornaika. A pyramid multi-level face descriptor: Application to kinship verification. *Multimedia Tools and Applications*, 80(1):123–145, 2021.

[49] Dong Ping Tian et al. A review on image feature extraction and representation techniques. *International Journal of Multimedia and Ubiquitous Engineering*, 8(4):385–396, 2013.

[50] Robert Plomin and Denise Daniels. Why are children in the same family so different from one another?*. *International Journal of Epidemiology*, 40(3):563–582, 06 2011.

[51] Apostolos Andrikopoulos and Jan Willem Duyvendak. Migration, mobility and the dynamics of kinship: New barriers, new assemblages. *Ethnography*, 21(3):299–318, 2020.

[52] Van Quang Nguyen and Hiromasa Fujihara. Revisiting a single-stage method for face detection. *arXiv preprint arXiv:1902.01559*, 1(2), February 2019.

[53] Wamidh K. Mutlag, Shaker K. Ali, Zahoor M. Aydam, and Bahaa H. Taher. Feature extraction methods: A review. *Journal of Physics: Conference Series*, 1591(1):012028, jul 2020.

[54] IBM Developer. Ibm developer. https://www.ibm.com/topics/deep-learning. Accessed: 2024-05-31.

[55] Mohssen Mohammed, Muhammad Badruddin Khan, and Eihab Mohammed Bashier Bashier. *Machine Learning: Algorithms and Applications*. CRC Press, 2016.

[56] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Elsevier, 2011.

[57] Iqbal H. Sarker, A. S. M. Kayes, Shahriar Badsha, Hamed Alqahtani, Paul Watters, and Alex Ng. Cybersecurity data science: An overview from a machine learning perspective. *Journal of Big Data*, 7(1):1–29, 2020.

[58] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[59] code cadmy. code cadmy. https://www.codecademy.com/learn/paths/machine-learning. Accessed: 2024-05-31.

[60] M. Trotter. Transfer learning: a friendly introduction. *Journal of Big Data*, 9(102):1–16, 2022.

[61] Dipanjan (DJ) Sarkar. A comprehensive hands-on guide to transfer learning with real-world applications in deep learning. *Towards Data Science*, 20:2020, 2018.

[62] Jason Brownlee. A gentle introduction to transfer learning for deep learning. *MachineLearningMastery.com*, 2019.

[63] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

[64] Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. *Journal of Artificial Intelligence Research*, 42(3):123–137, 2021.

[65] Peng Du, En Li, Jun Xia, Alim Samat, and Xiang Bai. Feature and model level fusion of pretrained cnn for remote sensing scene classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(7):2555–2561, 2018.

[66] Nutan. Transfer learning using feature extraction in deep learning. *Medium*, 2023.

[67] M. Georgopoulos, Y. Panagakis, and M. Pantic. Modelling of facial aging and kinship: A survey. *Image and Vision Computing*, 76:1–13, 2018.

[68] T. R. Alley. *Social and Applied Aspects of Perceiving Faces*. Psychology Press, 2013.

[69] J. H. Park, M. Schaller, and M. Van Vugt. Psychology of human kin recognition: Heuristic cues, erroneous inferences, and their implications. *Review of General Psychology*, 12(3):215, 2008.

[70] Rodney Needham, editor. *Rethinking Kinship and Marriage.* Taylor Francis, 2004.

[71] D. Jones. The generative psychology of kinship: Part 1. cognitive universals and evolutionary psychology. *Evolution and Human Behavior*, 24(5):303–319, 2003.

[72] L. T. Maloney and M. F. Dal Martello. Kin recognition and the perceived facial similarity of children. *Journal of Vision*, 6(10):4–4, 2006.

[73] Maria F Dal Martello and Laurence T Maloney. Where are kin recognition signals in the human face? *Journal of Vision*, 6(12):2, 2006.

[74] Xiaosheng Wu and Junding Sun. Joint-scale lbp: a new feature descriptor for texture classification. *The Visual Computer*, 33(3):317–329, 2015.

[75] Shuicheng Xia, Ming Shao, Jiebo Luo, and Yun Fu. Understanding kin relationship in a photo. *IEEE Transactions on Multimedia*, 14(4):1046–1056, 2012.

[76] Mudit Verma. Artificial intelligence and its scope in different areas with special reference to the field of education. *International Journal of Advanced Educational Research*, 3(1):5–10, 2018.

[77] Wolfgang Ertel. *Introduction to Artificial Intelligence.* Springer International Publishing, Berlin/Heidelberg, Germany, 2018.

[78] Beverly Park Woolf. *Building Intelligent Interactive Tutors: Student-centered Strategies for Revolutionizing E-learning.* Morgan Kaufmann, 2009.

[79] Michael I. Jordan and Thomas M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[80] Ayushi Chahal and Preeti Gulia. Machine learning and deep learning. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 8(12):1225, 2019.

[81] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: An overview and application in radiology. *Insights into Imaging*, 9:611–629, 2018.

[82] Samaya Madhavan and M. Tim Jones. Deep learning architectures. *IBM Developer*, 2024.

[83] Edwin Lisowski. Deep learning architecture. *Addepto Blog*, 2024.

[84] Donald E. Knuth. *The TEX Book*. Addison-Wesley Professional, 1986.

[85] Mathieu Fauvel, Yuliya Tarabalka, Jon Atli Benediktsson, Jocelyn Chanussot, and James C. Tilton. Advances in spectral-spatial classification of hyperspectral images. *Proceedings of the IEEE*, 101(3):652–675, 2013.

[86] GeeksforGeeks. Principal component analysis (pca). *GeeksforGeeks*, 2023.

[87] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*, volume 3, pages 32–36. IEEE, 2004.

[88] Gongde Gue, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. In *Proceedings of ON the Move to Meaningful Conferences, CoopIS, DOA, and ODBASE 2003*, pages 986–996. Springer, 2003.

[89] Laurens van der Maaten, Eric Postma, and Jaap van den Herik. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10:66–71, 2009.

[90] Vivienne Sze. Unleashing the power of deep learning. *MIT Open Access Articles*, 1:1–10, 2023.

[91] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.

[92] Nermeen Nader, Fatma El-Zahraa El-Gamal, Shaker El-Sappagh, Kyung Sup Kwak, and Mohammed Elmogy. Kinship verification and recognition based on handcrafted and deep learning feature-based techniques. *PeerJ Computer Science*, 7:e735, 2021.

[93] Muhammad Jamil Khan, Zikria Khalid, Abdur Rehman, and Muhammad Adeel Asghar. Kinship verification using deep neural network models. *IEEE Access*, 7:123456–123467, 2019.