POSE INVARIANT FACE RECOGNITION USING CNN

Major project report submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology
in
Electronics and Communication Engineering

Submitted by

B. LIKITHA	**(15B81A0466)**
K. SIDDHARTHA	**(15B81A04B9)**
P. SRI HARSHA VARDHAN	**(15B81A04C0)**

Department of Electronics & Communication Engineering
CVR COLLEGE OF ENGINEERING
(An Autonomous Institution & Affiliated to JNTUH)
Ibrahimpatnam (M), Ranga Reddy (D), Telangana

2015-2019

POSE INVARIANT FACE RECOGNITION USING CNN

Major project report submitted in partial fulfillment of the requirements for the degree of

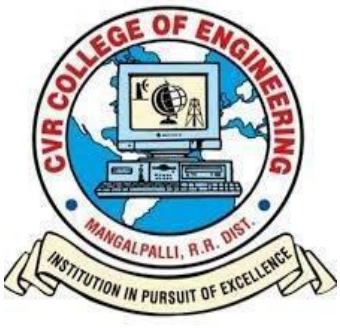
Bachelor of Technology
in
Electronics and Communication Engineering

Submitted by

B. LIKITHA	**(15B81A0466)**
K. SIDDHARTHA	**(15B81A04B9)**
P. SRI HARSHA VARDHAN	**(15B81A04C0)**

Under the Supervision of

P. HEMA SREE
Associate Professor



Department of Electronics & Communication Engineering
CVR COLLEGE OF ENGINEERING
(An Autonomous Institution & Affiliated to JNTUH)
Ibrahimpatnam (M), Ranga Reddy (D), Telangana

2015-2019

Cherabuddi Education Society's
CVR COLLEGE OF ENGINEERING
(An Autonomous Institution)
ACCREDITED BY NATIONAL BOARD OF ACCREDITATION, AICTE
(Approved by AICTE & Govt. of Telangana and Affiliated to JNT University)
Vastunagar, Mangalpalli (V), Ibrahimpatan (M), R.R. District, PIN - 501 510
Web : http://cvr.ac.in, email : info@cvr.ac.in
Ph : 08414 - 252222, 252369, Office Telefax : 252396, Principal : 252396 (O)

CERTIFICATE

This is to certify that the project titled "**Pose Invariant Face Recognition using CNN"** submitted to the **CVR College of Engineering,** affiliated to **JNTU, Hyderabad** by **B. Likitha (15B81A0466), K. Siddhartha (15B1A04B9)** and **P. Sri Harsha Vardhan (15B81A04C0)** is a bonafide record of the work done by the students towards partial fulfillment of requirements for the award of the degree of **Bachelor of Technology in Electronics & Communication Engineering**.

Supervisor

Signature:

Mrs. P. Hema Sree
Associate Professor
Department of Electronics &
Communication Engineering

Head of the Department

Signature:

Dr. K. Lalithendra
HOD & Professor
Department of Electronics &
Communication Engineering

Place: Hyderabad

Date:

ACKNOWLEDGEMENT

Completion of this project and thesis would not have been possible without the help of many people, to whom we are very thankful. First of all, we would like to convey our sincere thanks to **Dr. K. S. Nayanathara**, **Principal**, CVR College of Engineering for her constant support and encouragement.

We would like to express our sincere gratitude to our supervisor, **Mrs. P. Hema Sree, Associate Professor** for her constant motivation, guidance and support which helped us a great deal to achieve this feat.

We would like to thank **Dr. K. Lalithendra, HOD and Professor of ECE**, for guiding and inspiring us in many ways. We are also thankful to other faculty members and staff of Electronics and Communication department for their support.

We would like to thank **Mr. D. Bhanu Prakash, Associate Professor** for his constant support and cooperation throughout the course of the project. We would also like to thank all our friends within and outside the department for their encouragement, motivation and the experiences that they shared with us.

We also thank our coordinators **Dr. V. Arthi, Associate Professor, Mrs. G. Snehalatha, Assistant Professor** and **Mr. G. Ravi Kumar Reddy, Assistant Professor** for their constant support in guiding us about the academic related work.

We are also thankful to our friends who have helped us by contributing their pictures to build our database.

We wish a deep sense of gratitude and heartfelt thanks to management for providing excellent lab facilities and tools. Finally, we thank all those whose guidance helped us in this regard.

ABSTRACT

This abstract aims at the development of a robust face recognition system which is suitable in identifying faces irrespective of various alignments or poses of the face. Most of the existing algorithms in face detection are only successful in controlled environments and fail drastically when non linear functions such as pose variation, illumination and occlusion are affecting the image. Face recognition under uncontrolled environment is most important aspect in enhancing HCI (Human Computer Interface). Convolutional Neural Networks is used in applying feature extraction to normalize the data causing the system to cope with faces subject to pose variation.

CNN is a class of deep neural networks which is most commonly applied in analysing and classification of visual imagery. It uses relatively little pre-processing compared to other image classification algorithms. The main benefit of using CNN over simple ANN (Artificial Neural Networks) is that CNN's are constrained to deal with image data exclusively. One of the main features of this algorithm is **weight sharing,** as a result it reduces the number of weights significantly. Furthermore processing of high quality images which consists of very high number of pixels is a tedious task when implemented in ANN, here we can understand the need of CNN and how they can scale to high resolution images also. Another advantage of it is that they are very good **feature extractors.** This means that you can extract useful attributes from an already trained CNN with its trained weights by feeding your data on each level and tune it a bit for the specific task. It is also used in video recognition, recommender systems, natural language processing, etc.

In this project, we implement CNN based features for extraction of various features of a face which will be helpful in detecting the face irrespective of non linear functions as stated above and we compare the performance of existing algorithms and proposed algorithm.

KEYWORDS: Face Recognition, CNN, Image processing and Pose Variation.

TABLE OF CONTENTS

Certificate iii

Acknowledgement iv

Abstract v

Table of Contents vi

Abbreviations ix

List of Symbols x

List of Figures xi

List of Tables xiii

1 INTRODUCTION 1

1.1 Context 1

1.2 Applications 2

1.3 Difficulties 4

1.3.1 Illumination 4

1.3.2 Pose 5

1.3.3 Facial Expressions 5

1.3.4 Partial Occlusions 6

1.3.5 Other Type of Variations 6

1.4 Project Significance 7

1.5 Project Approach 7

1.6 Project Outcomes 8

1.7 Thesis Structure 8

1.8 Conclusion 8

2 LITERATURE REVIEW 9

2.1 Introduction 9

2.2 Emerging Trends in Face Recognition 9

2.3 Face Recognition Structure 13

2.4 Face Recognition Structure 14

2.4.1 Holistic Matching Methods 14
2.4.2 Feature Based Methods 15
2.4.3 Hybrid Methods 15
2.5 Face Recognition Applications 16
2.5.1 Applications and Examples 17
2.6 Conclusion 20
3 POSE INVARIANT FACE RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK (CNN) 21
3.1 Introduction 21
3.1.1 Neural Networks 22
3.1.2 Perceptron 22
3.1.3 Multi-Layer Perceptron 23
3.1.4 Training Neural Networks 23
3.1.5 Backpropagation Algorithm 24
3.1.6 Advantages of Using Neural Networks 27
3.1.7 Disadvantages of Using Neural Networks 27
3.2 Convolutional Neural Networks 28
3.2.1 Design 28
3.2.2 Layers 32
3.2.2.1 Convolutional Layer 32
3.2.2.2 Pooling Layer 33
3.2.2.3 Activation Functions 35
3.2.3 Classification 38
3.2.3.1 Flatten 39
3.2.3.2 Fully-Connected 39
3.2.3.3 Logistic Regression Classifier 39
3.2.3.4 Softmax Classifier 42
3.3 Pose-Invariant Face Recognition 42
3.4 CNN Block Diagram 44
3.4.1 Specification of Architecture 44
3.4.2 Advantages of CNN 45
3.4.3 Limitations of CNN 45
3.4.4 Case Studies 45
3.5 Conclusion 46

4 TOOLS & LIBRARIES USED 47
4.1 Introduction 47
4.2 Programming Languages 47
4.2.1 Python 3.7 47
4.2.2 Matlab 2018 48
4.3 Applications Used 50
4.3.1 Irfan View 50
4.3.2 Google Colaboratory 51
4.4 Conclusion 51
5 RESULTS AND PROJECT ANALYSIS 52
5.1 Databases Used 52
5.1.1 ATT Database 52
5.1.2 FERET Database 53
5.1.3 WEBCAM Database 54
5.1.4 Pose Database 55
5.1.5 OWN Database 56
5.2 Training Results 57
5.3 Testing Results 59
5.4 Existing Methods Comparison 60
5.5 Alternative Methods 60
5.5.1 Generation of Frontal Face Using PCA 60
5.5.2 Pose Estimation Using Patch Based Recognition 61
5.6 Hardware Comparison 61
CONCLUSION & FUTURE SCOPE 62
REFERENCES
APPENDIX

ABBREVATIONS

NFFR	**Near Frontal Face Recognition**
PIFR	**Pose Invariant Face Recognition**
CNN	**Convolutional Neural Network**
HCI	**Human Computer Interaction**
NN	**Neural Network**
FR	**Face Recognition**
ANN	**Artificial Neural Network**
MLP	**Multi-Layer Perceptron**
SVM	**Support Vector Machine**
PCA	**Principal Component Analysis**
KNN	**K Nearest Neighbors**
BCNN	**Bilinear Convolutional Neural Network**
DNN	**Deep Neural Network**
LDA	**Linear Discriminant Analysis**
BP	**Backpropagation**
ReLU	**Rectified Linear Unit**
GPU	**Graphic Processing Unit**

LIST OF SYMBOLS

1) Φ **Heaviside step function**

2) ∇ **Gradient**

3) δ **Local gradient**

4) λ **Regularization parameter**

5) $\sigma(z)$ **Softmax Classifier**

LIST OF FIGURES

1.1	**Block diagram of Face Recognition**	**1**
1.2	**An example of faces under a fixed view and varying illumination**	**4**
1.3	**An example of faces under varying pose**	**5**
1.4	**An example of faces under fixed illum. and pose but varying facial expression**	**6**
1.5	**An example of faces with occlusions**	**6**
1.6	**An example of faces with other type of variations**	**7**
2.1	**Block Diagram of Facial Recognition**	**13**
2.2	**Eigen Faces**	**15**
2.3	**FR based on features**	**15**
2.4	**Hybrid Feature extraction using SVM**	**16**
2.5	**Face access control**	**17**
2.6	**An exemplar airport security system**	**18**
2.7	**FR in Surveillance**	**18**
2.8	**Card & FR Access Control**	**19**
2.9	**Police using FR to catch criminals**	**19**
2.10	**Data Science tools for Fraud Investigations**	**20**
2.11	**Face ID Recognition**	**20**
3.1	**Representation of a CNN**	**21**
3.2	**Perceptron**	**22**
3.3	**Different types of activation functions**	**23**
3.4	**Typical CNN Architecture**	**28**
3.5	**Convolution**	**29**
3.6	**Pooling**	**30**

3.7	**Fully Connected**	**30**
3.8	**Neurons connected to their Receptive Field**	**31**
3.9	**Weight sharing in CNNs**	**31**
3.10	**Convolutional Layer**	**33**
3.11	**Pooling Layers**	**34**
3.12	**Linear Activation Function**	**35**
3.13	**Non-Linear Activation Function**	**35**
3.14	**Sigmoid Activation Function**	**36**
3.15	**Tanh Activation Function**	**36**
3.16	**ReLU Activation Function**	**37**
3.17	**ReLU v/s Leaky ReLU**	**37**
3.18	**Classification Using Log. Reg. Function**	**39**
3.19	**CNN Block Diagram**	**44**
5.1	**ATT Database Images**	**52**
5.2	**FERET Database Images**	**53**
5.3	**Webcam Database Images**	**54**
5.4	**Pose Database Images**	**55**
5.5	**Own Database Images**	**56**
5.6	**ATT Training Result**	**57**
5.7	**FERET Training Result**	**57**
5.8	**OWN Database Training Result**	**58**
5.9	**WEBCAM Database Training Result**	**58**
5.10	**Testing Comparison**	**59**
5.11	**Output of PCA**	**60**
5.12	**Output of Pose Estimator**	**61**
5.13	**Hardware Comparison**	**61**

LIST OF TABLES

3.1	**Different Activation Functions**	**38**
5.1	**Overall Comparison**	**59**
5.2	**Existing Methods**	**60**

Chapter One

INTRODUCTION

1.1. Context

Face recognition has been one of the most intensively studied topics in computer vision for more than four decades. Compared with other popular biometrics such as finger print, iris, and retina recognition, face recognition has the potential to recognize uncooperative subjects in a non-intrusive manner. Therefore, it can be applied to surveillance security, border control, forensics, digital entertainment, etc. Indeed, numerous works in face recognition have been completed and great progress has been achieved, from successfully identifying criminal suspects from surveillance cameras to approaching human level performance. These successful cases, however, may be unrealistically optimistic as they are limited to near-frontal face recognition (NFFR). Recent studies reveal that the best NFFR algorithms perform poorly in recognizing faces with large poses. In fact, the key ability of pose-invariant face recognition (PIFR) desired by real-world applications remains largely unsolved.

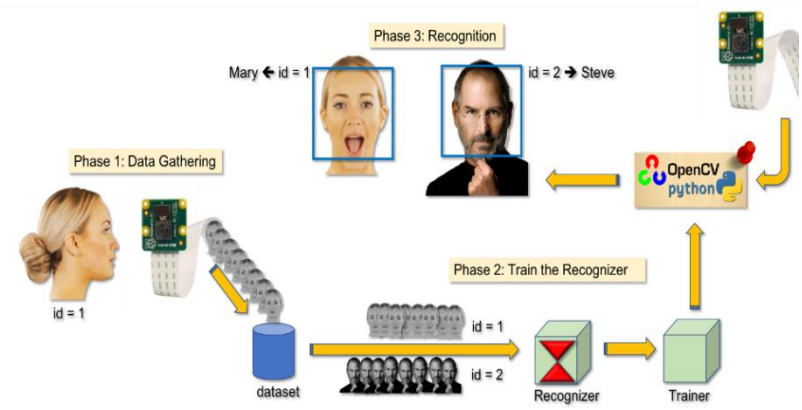


Fig. 1.1 Block diagram of Face Recognition

In the context of human-computer interaction (HCI), it might also be important to detect the position of specific facial characteristics or recognize facial expressions, in order to allow, for example, a more intuitive communication between the device and the

user or to efficiently encode and transmit facial images coming from a camera. Thus, the automatic analysis of face images is crucial for many applications involving visual content retrieval or extraction.

The principal aim of facial analysis is to extract valuable information from face images, such as its position in the image, facial characteristics, facial expressions, the person's gender or identity. We will outline the most important existing approaches to facial image analysis and present novel methods based on Convolutional Neural Networks (CNN) to detect, normalize and recognize faces and facial features. CNNs show to be a powerful and flexible feature extraction and classification technique which has been successfully applied in other contexts, i.e. hand-written character recognition, and which is very appropriate for face analysis problems as we will experimentally show in this work. We will focus on the processing of two-dimensional gray-level images as this is the most widespread form of digital images and thus allows the proposed approaches to be applied in the most extensive and generic way. However, many techniques described in this work could also be extended to color images, 3D data or multi-modal data.

1.2. Applications

There are numerous possible applications for facial image processing algorithms. The most important of them concern face recognition. In this regard, one has to differentiate between closed world and open world settings. In a closed world application, the algorithm is dedicated to a limited group of persons, e.g. to recognize the members of a family. In an open world context, the algorithm should be able to deal with images from "unknown" persons, i.e. persons that have not been presented to the system during its design or training. For example, an application indexing large image databases like Google images or television programs should recognize learned persons and respond with "unknown" if the person is not in the database of registered persons.

Concerning face recognition, there further exist two types of problems: face identification and face verification (or authentication). The first problem, face identification, is to determine the identity of a person on an image. The second one only

deals with the question: “Is ‘X’ the identity of the person shown on the image?” or “Is the person shown on the image the one he claims to be?”. These questions only require “yes” or “no” as the answer. Possible applications for face authentication are mainly concerned with access control, e.g. restricting the physical access to a building, such as a corporate building, a secured zone of an airport, a house etc. Instead of opening a door by a key or a code, the respective person would communicate an identifier, e.g. his/her name, and present his/her face to a camera. The face authentication system would then verify the identity of the person and grant or refuse the access accordingly. This principle could equally be applied to the access to systems, automatic teller machines, biometric scanners, mobile phones, and Internet sites etc. where one would present his face to a camera instead of entering an identification number or password. Clearly, also face identification can be used for controlling access. In this case the person only has to present his/her face to the camera without claiming his/her identity. A system recognizing the identity of a person can further be employed to control more specifically the rights of the respective persons stored in its database. For instance, parents could allow their children to watch only certain television programs or web sites, while the television or computer would automatically recognize the persons in front of it.

Video surveillance is another application of face identification. The aim here is to recognize suspects or criminals using video cameras installed at public places, such as banks or airports, in order to increase the overall security of these places. In this context, the database of suspects to recognize is often very large and the images captured by the camera are of low quality, which makes the task rather difficult. With the vast propagation of digital cameras in the last years the number of digital images stored on servers and personal home computers is rapidly growing. Consequently, there is an increasing need of indexation systems that automatically categorize and annotate this huge number of images in order to allow effective searching and so-called content-based image retrieval. Here, face detection and recognition methods play a crucial role because a great part of photographs actually contain faces. A similar application is the temporal segmentation and indexation of video sequences, such as TV programs, where different scenes are often characterized by different faces.

1.3. Difficulties

There are some inherent properties of faces as well as the way the images are captured which make the automatic processing of face images a rather difficult task. In the case of face recognition, this leads to the problem that the intra-class variance, i.e. variations of the face of the same person due to lighting, pose etc., is often higher than the inter-class variance, i.e. variations of facial appearance of different persons, and thus reduces the recognition rate. In many face analysis applications, the appearance variation resulting from these circumstances can also be considered as noise as it makes the desired information, i.e. the identity of the person, harder to extract and reduces the overall performance of the respective systems. In the following, we will outline the most important difficulties encountered in common real-world applications.

1.3.1. Illumination

Changes in illumination can entail considerable variations of the appearance of faces and thus face images. Two main types of light sources influence the overall illumination: ambient light and point light (or directed light). The former is somehow easier to handle because it only affects the overall brightness of the resulting image. The latter however is far more difficult to analyze, as face images taken under varying light source directions follow a highly non-linear function. Additionally, the face can cast shadows on itself..

Fig. 1.2 An example of faces under a fixed view and varying illumination

Many approaches have been proposed to deal with this problem. Some face detection or recognition methods try to be invariant to illumination changes by implicitly

modeling them or extracting invariant features. Others propose a separate processing step, a kind of normalization, in order to reduce the effect of illumination changes.

1.3.2. Pose

The variation of head poses or, in other words, the viewing angle from which the image of the face was taken is another difficulty and essentially impacts the performance of automatic face analysis methods. For this reason, many applications limit themselves to more or less frontal face images or otherwise perform a pose-specific processing that requires a preceding estimation of the pose, like in multi-view face recognition approaches.

If the rotation of the head coincides with the image plane the pose can be normalized by estimating the rotation angle and turning the image such that the face is in an upright position. This type of normalization is part of a procedure called face alignment or face registration.

Fig. 1.3 An example of faces under varying pose

1.3.3. Facial Expressions

The appearance of a face with different facial expressions varies considerably (see Fig. 1.4).In general, the mouth is subject to the largest variation. The respective person on an image can have an open or closed mouth can be speaking, smiling, laughing or even

making grimaces. Eyes and eyebrows are also changing subject to varying facial expressions, e.g. when the respective person blinks, sleeps or widely opens his/her eyes.

Fig. 1.4 An example of faces under fixed illum. and pose but varying facial expression

1.3.4. Partial Occlusions

Partial occlusions occur quite frequently in real-world face images. They can be caused by a hand occluding a part of the face, e.g. the mouth, by long hear, glasses, sun glasses or other objects or persons.

Fig. 1.5 An example of faces with occlusions

1.3.5. Other types of variations

Appearance variations are also caused by varying make-up, varying hair-cut and the presence of facial hear (beard, mustache etc.). Varying age is also an important factor

influencing the performance of many face analysis methods. There are also variations across the subjects' identities, such as race, skin color or, more generally, ethnic origin. The respective differences in the appearance of the face images can cause difficulties in applications like face or facial feature detection or gender recognition.

Fig. 1.6 An example of faces with other type of variations

1.4. Project Significance

The basic idea behind this project is to identify faces under different pose variations and achieve a higher recognition rate using CNN architecture.

1.5. Project Approach

The goals pursued in this work principally concern the evaluation of Convolutional Neural Networks (CNN) in the context of facial analysis applications. More specifically, we will focus on the following objectives:

- Evaluate the performance of CNNs w.r.t. appearance-based facial analysis.
- Investigate the robustness of CNNs against classical datasets like FERET, ATT, etc.
- Propose different CNN architectures designed for specific facial analysis problems such as face alignment, facial feature detection and face recognition.

- Extend the project to appearance-based facial feature detection, face alignment as well as face recognition under real-world conditions.
- Extend the project to synthesize and train images for facial 3D pose estimation.

1.6. Project Outcomes

To identify images with higher recognition rate with minimum pre-processing. In terms of performance, CNNs outperform NNs on conventional image recognition tasks and many other tasks.

1.7. Thesis Structure

Chapter1 describes the face recognition technology development stages, project significance, approach and outcomes.

Chapter 2 gives information about FR technology, applications and challenges.

Chapter 3 is about neural networks (NNs), PIFR using CNN.

Chapter 4 explains about the software tools and libraries used.

Chapter 5 is about Project implementation and simulation.

Chapter 6 deals with the advantages, disadvantages and limitations of using CNN.

1.8. Conclusion

This chapter gives the brief introduction to project and describes the development in Face recognition.

Chapter Two

LITERATURE REVIEW

2.1. Introduction

Recognizing an individual is one thing that every human does effortlessly and without much conscious thought. But it remains a problem when a computer has to identify or recognize a person. Biometric is one of the areas where face recognition is used widely. Some of the biometric techniques are iris detection, finger print detection or voice detection. However, face recognition holds many advantages when compared with the above mentioned biometric technique. Below are some of the evolving techniques in FR field.

2.2. Emerging Trends in Face Recognition

- **Face Recognition by Support Vector Machines (2000):** Uses Cambridge ORL database. The experimental results show that this method is better than nearest center approach for face recognition.
- **Face Recognition with Support Vector Machines (2001):** the database includes the faces rotated in depth up to 40‟. This method shows that using facial components as facial features simplifies the recognition task rather than using whole face.
- **Facial Component Extraction and Face Recognition with Support Vector Machines (2002):** this method is quick and robust where the algorithm is applied to the faces of different sizes.
- **Face Recognition Using Support Vector Machines with the Robust Feature (2003):** The ORL database is used for testing. This algorithm introduces the Gabor wavelet, KCPA and SVM. The SVM is used to classify the robust features and this obtains high accuracy.
- **Improving the Performance of Multi-Class SVMs in Face Recognition with Nearest Neighbor Rule (2003):** NNR method is introduced to mainly reduce

training class levels. This process performs much faster than the available SVM methods and the classification process is much faster.

- **A SVM-Based Method for Face Recognition using a Wavelet PCA Representation of Faces (2004):** This method uses the concept of SVM. The proposed method includes a substantial reduction in error rate.
- **Bayesian Face Recognition Using Support Vector Machine and Face Clustering (2004):** We first develop a direct Bayesian based Support Vector Machine by combining the Bayesian analysis with the SVM. Then the experiment is carried out using the two traditional databases FERET and XM2VTS. This method also yields some acceptable results.
- **Face recognition in color image using PCA and FSVM (2005):** This method uses color images for recognition process. This method uses skin color as one clue to recognize the face. Experiments were conducted using the indoors photograph and the results demonstrated that the proposed method was efficient for the frontal face detection and recognition.
- **A SVM Face Recognition Method Based on Gabor-Featured Key Points (2005)**: A novel faces recognition approach based on Support Vector Machine and Gabor-Featured Key Points. This experiment is conducted on FERET and AT&T databases.
- **Facial Feature Selection Based on SVMs by Regularized Risk Minimization (2006):** Proposed method is an effective solution for high dimensional facial feature selection. And this is based on SVM by regularized risk minimization.
- **Face Recognition using Multiple Classifiers (2006):** Here various classification techniques like support vector machine (SVM), linear discriminate analysis (LDA) and K nearest neighbor (KNN) are studied.
- **Face Recognition Using Total Margin-Based Adaptive Fuzzy Support Vector Machines (2007):** A new classifier called total margin-based adaptive fuzzy support vector machines (TAF-SVM) is presented to deal with the several problems that occur in SVM. Results show that the TAF-SVM is superior to SVM in terms of the face-recognition accuracy.

- **Face recognition using Multi Scale PCA and support vector machine (2008):** This approach has obtained a good recognition performance through reorganizing the Gabor feature, reducing the dimension with MS-PCA and classifies SVM. This method has two limitations one is the selection of kernel parameters and other is the number of SSM classifiers used here are more.
- **Face Recognition System using SVM and Feature Extraction by PVCA and LDA Combination (2009):** First PCA is used for dimension reduction and LDA is used for feature extraction then finally SVM is used as a Classifier. The experiments results show that PCA+LDA+SVM method has higher recognition rate than the other two methods PCA+NCC and PCA+LDA+NCC (nearest neighbor classifier). ORL database is used here. The recognition rate of this system is 94.3%.
- **Face Recognition Based on Face Gabor Image and SVM (2009):** This is an effective algorithm for face recognition using face Gabor image and Support Vector Machine (SVM). Face Gabor image is firstly derived by down sampling and concatenating the Gabor wavelets representations which are the convolution of the face image with a family of Gabor kernels, and then the 2D Principle Component Analysis (2DPCA) method is applied to the face Gabor image to extract the feature space. Finally, Support Vector Machine (SVM) is used to classify. This method uses ORL data base.
- **ISVM for Face Recognition (2010):** Similarity of human faces, unpredictable variations and aging are the crucial obstacles in face recognition to handle this large stet of training samples are required which increases the complexity of the system. Since both classification and feature information are necessary for a recognition system DCT is used to lower the computational complexity and SVM for classification. Since SVM is a popular classification tool but the main disadvantage of SVM is its large memory requirement and computation time to deal with large data set. The biggest advantage of using the proposed technique is that it not only decreases the training time and updating time but also improves the classification accuracy rate up to 100 %.

- **Face recognition based on principle component analysis and support vector machine (2011):** Face recognition is done using PCA as feature extractor and SVM is as classifier. Experiments are carried out on ORL data base. Compares proposed method with PCA and nearest neighbors (PCA and NN) methods of face recognition and support vector machine on recognition rate and recognition time respectively. This method is beneficial only for small sample of training data.
- **Sparse Representation based Classification (2011):** Sparse representation-based classification (SRC) has been widely used for face recognition (FR). SRC first codes a testing sample as a sparse linear combination of all the training samples, and then classifies the testing sample by evaluating which class leads to the minimum representation error.
- **Collaborative Representation Based Face Recognition (2011):** This technique is the special case of sparse based classification.
- **Sparse Based Representation using k nearest subspace (2014):** (SRC) has been proven to be a robust face recognition method. For this modular method, face images are partitioned into a number of blocks first and then we propose an indicator to remove the contaminated blocks and choose the nearest subspaces. Finally, SRC is used to classify the occluded test sample in the new feature space.
- **Bilinear Convolutional Neural Networks (BCNN) (2015):** It is applied for facial recognition in large public data sets with pose variability. 4 sets of features are evaluated: traditional features (eyes, nose, mouth, and eyebrows), features correlated to accessories, features correlated with hair, and features correlated with background.
- **Deep Learning Face Recognition/ Deep Convolutional Neural Networks (DNN) (2016):** Deep learning does a better job than humans at figuring out which parts of a face are important to measure. Deep learning algorithm is employed in many real time applications for example face book uses DNN for face recognition.
- **Super Pixels (2017):** This is the newest technique for face recognition. Here the patterns or the similar pixels are grouped together for feature extraction and recognition is done. This method is still under development.

2.3. Face Recognition Structure

Let us consider a picture captured from a digital camera, and we would like to know who the person is in the picture. To achieve this, we perform face recognition in three following steps.

- Face Detection
- Feature Extraction
- Face Recognition

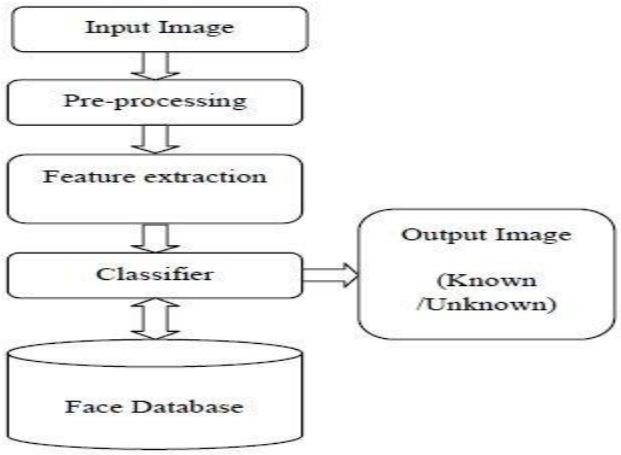


Fig. 2.1 Block Diagram of Facial Recognition

Face Detection:

The main aim of this step is to identify whether a face is present in the image. The expected outcomes are some patches containing the features of the face. Face alignment of the face is performed to know the scales and orientations of these patches. Face detection also helps in retargeting, video and image classification and also helps in concentrating the region of interest.

Face Extraction:

After the detection step patches are extracted by directly using the patches we cannot build a recognition system which is robust as image has some 1000 pixels, which is too large. Other disadvantages of directly using the patch is that each patch has

different illumination, pose, camera alignments which may clutter. To overcome these problems features are extracted so that dimension reduction and noise cleaning can be done. After this step, face patches are transformed to a vector.

Face Recognition:

This is the last step of this process. In order to process this step a face database has to be built. For a person several images are taken and are stored in this database. After obtaining a face that has to be recognized face detection and feature extraction are done then the patches are compared with the faces available in the database. There are many algorithms proposed for classification.

2.4. Face Recognition Methods

Developing a very effective face recognition system is not an easy task. Several factors have to be taken in to consideration. Few of them are stated below:

- Overall speed from detection to recognition should be acceptable.
- The system should be accurate.
- It should be easy to increase the size of the subjects at any given point of time.

In the beginning people used to recognize faces using distances e.g. measurement between eyes or measurements between other nodal points. But with the changing patterns in technology face recognition can be classified into three methods:

- Holistic Matching Methods
- Feature Based (structural) Methods
- Hybrid Methods

2.4.1. Holistic Matching Methods

Here in holistic approach complete face is taken into consideration. One of the best examples of holistic approach is Eigen faces. The other methods of this approach are PCA, LDA etc.

Eigen faces:

At first data base is constructed and it is termed as the training set, next Eigen faces are made by extracting face features. Then the images are normalized and resized so that they have same size. By using the mathematical tool called Principal Component Analysis (PCA) Eigen faces are extracted. When Eigen faces are created, each image will be a vector with weights. Now, when an image is given as query, the weights of the query image is compared with the weights of the images in database. The image with the closest weight is the recognized face.

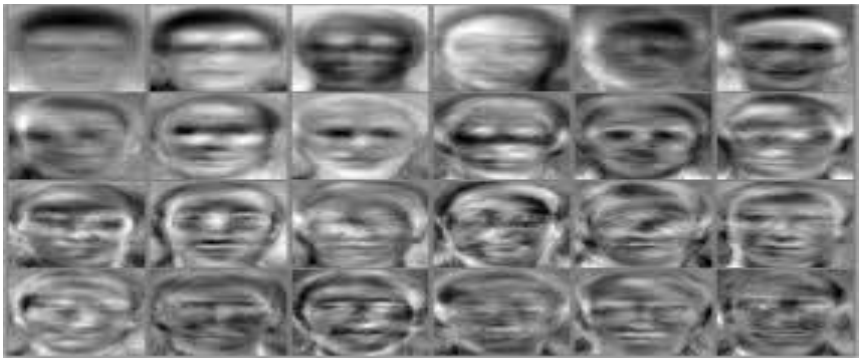


Fig. 2.2 Eigen Faces

2.4.2. Feature Based(structural) Methods

This is a local method were features such as eyes, nose, mouth are extracted and their locations and local statistics are stored into the classifier. This may be feature based or appearance based. The biggest disadvantage is the feature restoration i.e., it is difficult to retrieve an image with a large pose variation.

Fig. 2.3 FR based on features

The three different extraction methods are:

- Generic methods based on edges, lines, and curves
- Feature-template-based methods
- Structural matching methods that take into consideration geometrical Constraints on the features.

2.4.3. Hybrid Methods

Hybrid methods are those approaches that use both holistic and local feature-based methods. The main problem that affects the performance of a hybrid system is how to select the features that are to be combined and how to combine them so that their advantages are preserved and disadvantages are removed. For example, components of a hybrid system, either feature or classifier, should be both accurate and diverse such that a complementary advantage can be feasible.

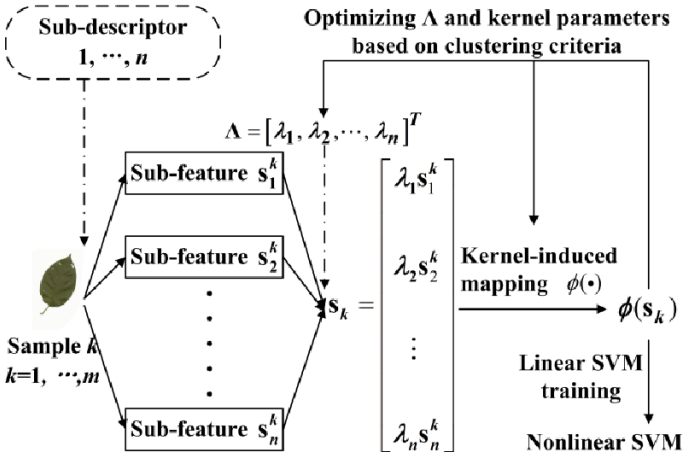


Fig 2.4 Hybrid Feature extraction using SVM

2.5. Face Recognition- Applications

Face recognition is useful in virtual reality, multimedia, computer entertainment, information security e.g. operating system, medical records, online banking., Biometric e.g. Personal Identification - Passports, driver licenses, Automated identity verification - border controls, Law enforcement e.g. video surveillances, investigation, Personal Security - driver monitoring system, home video surveillance system.

2.5.1. Applications and Examples

Face Identification:

Face recognition systems identify people using their faces rather using pins or pass codes for valid identification. Passwords, pins or other codes may be hijacked but a person's face cannot be hijacked hence face recognition system is used in identification.

Access Control:

In many access control applications face recognition is used such as office access or computer logon, automated teller machines. Face recognition applications in these types of applications can give high accuracy without the cooperation of the respective users. It does not involve in touching other objects by palms or finger or no need to place certain objects on eyes as in case of other recognition techniques like finger print detection or iris detection. Face recognition gives high accuracy when compared with the above techniques thus providing user satisfaction.

For example, University of Missouri – Rolla campus has a nuclear reactor which is 200kW research facility the access to this nuclear requires an authentication which involves face recognition process.

Fig. 2.5 Face access control

Security:

Security is the primary concern at the airports for both employees and also for the passengers. An airport protection system that uses face recognition technology has been

deployed in many airports around the world. Below figure shows the typical airport security system.

The following example explains how face recognition technology helps in airport security. In 2001, FYI deployed a system which recognizes faces of individuals who resemble a known terrorist. The recognized person is sent to further investigations.

Fig. 2.6 An exemplar airport security system

Surveillance:

Surveillance using face recognition system has also been employed in public areas to find the suspects. In Virginia being the second city in USA to be employed with the surveillance system on its public streets to scan the pedestrians and to compare with 2500 images of culprits, missing persons and runaways.

Fig. 2.7 FR in Surveillance

Smart Cards:

Smart card has an embedded microprocessor or a microchip on it which provides processing power to many applications. Memory cards can be used just for storing the

data whereas as smart cards are not only used for storing the data it can add delete, and manipulate the information. It also has built in security features. The application of face recognition on smart card is a combination of two which can be explained using below two examples. Smart cards store the mathematical characteristics of face during the enrolment stage. These characteristics are read during the verification stage.

Fig. 2.8 Card & FR Access Control

Law Enforcement:

With the face recognition systems investigators can find the suspects very easily and quickly. FRT can empower the law enforcement authorities the ability to search and identify a person even with incomplete information of their identity. For example, Images system provides information about the criminal history of the person with the database containing images of the person giving an invaluable tool for their law enforcement and surveillance work. With this face recognition and retrieval program, investigators no longer have to spend hundreds of hours trying to identify a suspect.

Fig. 2.9 Police Using FR to catch criminals

Image database investigations:

Searching image databases of licensed drivers benefit recipients, missing children, immigrants and police bookings.

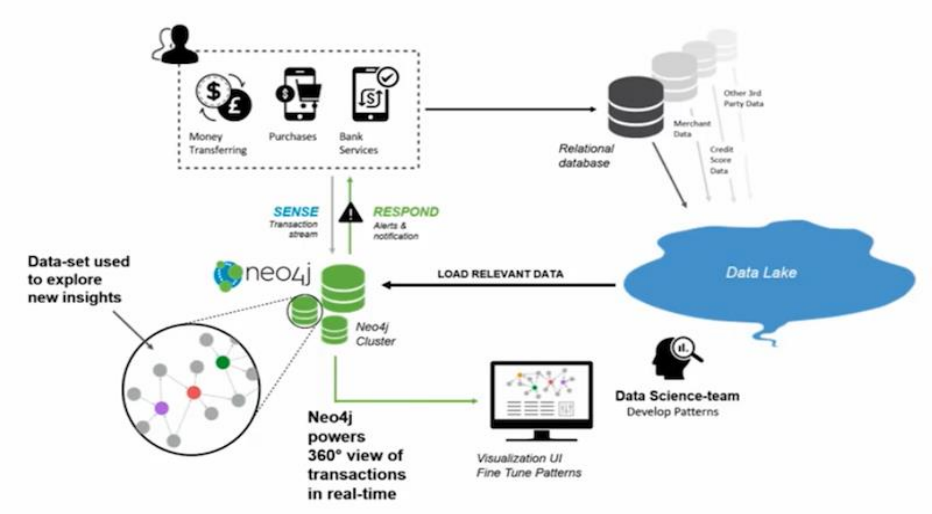


Fig. 2.10 Data Science tools for Fraud Investigations

General identity verification:

Face recognition is used for identification of National IDs, Passports, employee IDs.

Fig. 2.11 Face ID Recognition

2.6. Conclusion

This chapter gives the brief description of face recognition technology techniques, applications and challenges.

Chapter Three

POSE INVARIANT FACIAL RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK (CNN)

3.1. Introduction

In deeplearning, a **Convolutional Neural Network** (**CNN** or **ConvNet**) is a class of deep neural networks, most commonly applied to analyzing visual imagery. CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

Convolutional networks were inspired by biological processes. In that, the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing.

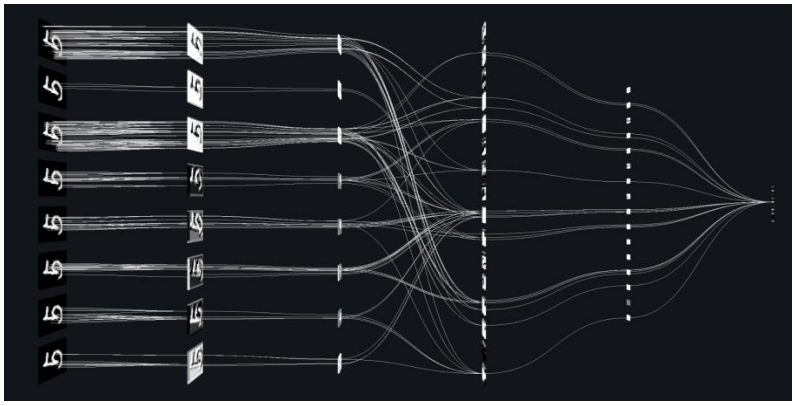


Fig. 3.1 Representation of a CNN

3.1.1. Neural Networks

Artificial Neural Networks (ANN), short Neural Networks (NN), denote a machine learning technique that has been inspired by the human brain and its capacity to perform complex tasks by means of inter-connected neurons performing each a very simple operation. Likewise, a NN is a trainable structure consisting of a set of inter-connected units, each implementing a very simple function, and together eventually performing a complex classification function or approximation task.

3.1.2. Perceptron

The most well-known type of neural unit is called Perceptron and has been introduced by Rosenblatt. Its basic structure is illustrated in Fig. 3.2. It has n inputs and one output where the output is a simple function of the sum of the input signals x weighted by w and an additional bias b. Thus,

$$y = \varphi (x \cdot w + b) \qquad (1)$$

Often, the bias is put inside the weight vector w such that w0 = b and the input vector x is extended correspondingly to have x0 = 1. Equation (1) then becomes:

$$y = \varphi (x \cdot w) \qquad (2)$$

where φ is the Heaviside step function :φ: R → R

$$\varphi(x) = 1 \text{ if } x \geq 0; 0 \text{ else.}$$

The Perceptron thus implements a very simple two-class classifier where w is the separating hyper plane such that $w \cdot x \geq 0$ for examples from one class and $w \cdot x < 0$.

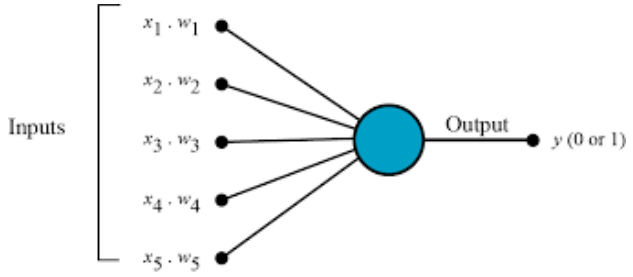


Fig. 3.2 Perceptron

3.1.3. Multi-Layer Perceptron

Multi-Layer Perceptrons (MLP) are capable of approximating arbitrarily complex decision functions. With the advent of a practicable training algorithm in the 1980's, the so-called Backpropagation algorithm, they became the most widely used form of NNs. There is an input layer, one or more hidden layer(s) and an output layer of neurons, where each neuron except the input neurons implements a perceptron as described in the previous section. Moreover, the neurons of one layer are only connected to the following layer. We call this type of network: feed-forward network, i.e. the activation of the neurons is propagated layer-wise from the input to the output layer. And there is a connection from each neuron to every neuron in the following layer. Further, the neurons' activation function has to be differentiable in order to adjust the weights by the Backpropagation algorithm. Commonly used activation functions are for example:

$\varphi(x) = x$	linear
$\varphi(x) = 1/(1 + e{-}cx)\ ;(c > 0)$	sigmoid
$\varphi(x) = (1 - e{-}x)/(1 + e{-}x)$	hyperbolic tangent

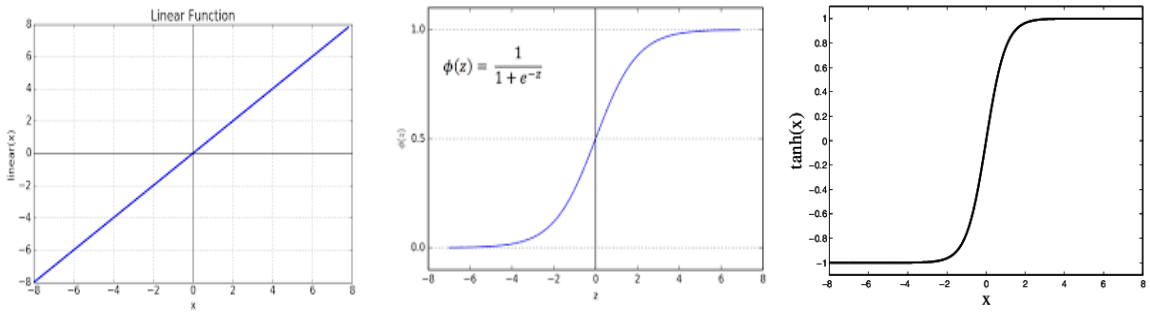


Fig. 3.3 Different types of activation functions

3.1.4. Training Neural Networks

In general, the parameters of a NN, i.e. the weights and biases, are learned using a training data set. However, as the space of possible weights can be very large and of high dimension the analytical determination of these weights might be very difficult or even infeasible. For this reason, an iterative approach is adopted in most cases. There are two principal training modes which determine the way the weights are updated:

Online training: After presentation of each training example, the error is calculated, and the weights are updated accordingly.

Offline training: The whole training set is propagated through the NN, and the respective errors are accumulated. Finally, the weights are updated using the accumulated error. This is also called batch training.

Many different NN training algorithms have been published in the literature. Some work only in online, some only in offline mode, and some can be executed in both ways. Which algorithm is best for a given problem depends on the NN architecture, the nature and cardinality of the training data set and the type of function to learn. Therefore, there is no basic rule for the choice of the training algorithm. In the following, we will focus on the Backpropagation algorithm since it is the most common and maybe most universal training algorithm for feed-forward NNs, especially for MLPs. Some alternative methods will also be presented at the end of this section.

3.1.5. Backpropagation Algorithm

In the context of NNs, the Backpropagation (BP) algorithm has initially been presented by Rumelhart et al. It is a supervised learning algorithm defining an error function E and applying the gradient descent technique in the weight space in order to minimize E. The combination of weights leading to a minimum of E is considered to be a solution of the learning problem. Note that the BP algorithm does not guarantee to find a global minimum which is an inherent problem of gradient descent optimization. However, we will discuss some approaches to overcome this problem in the following section. In order to calculate the gradient of E, at each iteration, the error function has to be continuous and differentiable. Thus, the activation function of each individual perceptron must also have this property. Mostly, a sigmoid or hyperbolic tangent activation function is employed, depending on the range of desired output values, i.e. [0, 1] or [-1, +1]. Note that BP can be performed in online or offline mode, i.e. He represents either the error of one training example or the sum of errors produced by all training examples. In the following, we will explain the standard online BP algorithm, also known as Stochastic Gradient Descent, applied to MLPs. There are two phases of the algorithm:

- The forward pass, where a training example is presented to the network and the activations of the respective neurons is propagated layer by layer until the output neurons.

- The backward pass, where at each neuron the respective error is calculated starting from the output neurons and, layer by layer, propagating the error back until the input neurons.

Now, let us define the error function as:

$$E=1/2\sum_{p=1}^{P} ||op - tp||^2 \qquad (6)$$

Where P is the number of training examples, op are the output values produced by the NN having presented example p, and tp are the respective target values. The goal is to minimize E by adjusting the weights of the NN. With online learning we calculate the error and try to minimize it after presenting each training example. Thus,

$$Ep = 1/2||op - tp||^2 = 1/2\sum_{k=1}^{K}(opk - tpk)^2 \qquad (7)$$

Where K is the number of output units. When minimizing this function by gradient descent, we calculate the steepest descent of the error surface in the weight space, i.e. the opposite direction of the gradient $\nabla Ep = (\partial Ep/\partial w1, \ldots, \partial Ep/\partial wk)$.In order to ensure convergence, the weights are only updated by a proportion of the gradient. Thus,

$$\Delta wk = -\lambda \partial Ep/\partial wk \qquad (8)$$

Now, let us define

$e_{pk} = o_{pk} - t_{pk}$	the error of pattern p at output neuron k
w^o_{kj}	the weight from hidden neuron j to output neuron k
w^h_{ji}	the weight from input neuron i to hidden neuron j
$z_{pj} = \phi\left(\sum_i w_{ji}\, x_{pi}\right)$	the output of hidden neuron j
$V_{pk} = \sum_j w^o_{kj}\, z_{pj}$	the weighted sum of all inputs z_{pj} of output neuron k
$V_{pj} = \sum_i w^h_{ji}\, x_{pi}$	the weighted sum of all inputs x_{pi} of hidden neuron j.

The subscript p always refers to pattern p, i refers to input neuron i, j to hidden neuron j and k to output neuron k. By applying the chain rule to equation (8), for one particular weight w_{kj}^{o} and training example p, we have:

$$\begin{aligned}
\Delta w_{kj}^{o} &= -\lambda \frac{\partial E_p}{\partial w_{kj}^{o}} \\
&= -\lambda \frac{\partial E_p}{\partial e_{pk}} \frac{\partial e_{pk}}{\partial o_{pk}} \frac{\partial o_{pk}}{\partial V_{pk}} \frac{\partial V_{pk}}{\partial w_{jk}^{o}} \\
&= -\lambda \, e_{pk} \, \phi'(V_{pk}) \, z_{pj} \\
&= -\lambda \, \delta_{pk} \, z_{pj} \quad ,
\end{aligned}$$

$$\delta_{pk} = e_{pk} \, \phi'(V_{pk})$$

This holds for output neurons. For the hidden neurons the respective equations are slightly different:

$$\begin{aligned}
\Delta w_{ji}^{h} &= -\lambda \frac{\partial E_p}{\partial w_{ji}^{h}} \\
&= -\lambda \frac{\partial E_p}{\partial z_{pj}} \frac{\partial z_{pj}}{\partial V_{pj}} \frac{\partial V_{pj}}{\partial w_{ji}^{h}} \\
&= -\lambda \left(\sum_{k=1}^{K} e_{pk} \frac{\partial e_{pk}}{\partial z_{pj}} \right) \phi'(V_{pj}) \, x_{pi} \\
&= -\lambda \left(\sum_{k=1}^{K} e_{pk} \frac{\partial e_{pk}}{\partial o_{pk}} \frac{\partial o_{pk}}{\partial V_{pk}} \frac{\partial V_{pk}}{\partial z_{pj}} \right) \phi'(V_{pj}) \, x_{pi} \\
&= -\lambda \left(\sum_{k=1}^{K} e_{pk} \, \phi'(V_{pk}) \, w_{kj}^{o} \right) \phi'(V_{pj}) \, x_{pi} \\
&= -\lambda \, \delta_{pj} \, x_{pi}
\end{aligned}$$

where

$$\delta_{pj} = \left(\sum_{k=1}^{K} \delta_{pk} w_{kj}^{o} \right) \phi'(V_{pj}) \qquad (10)$$

The local gradient for hidden neuron j (j = 1...J). Algorithm 3 summarizes the standard online Backpropagation algorithm. The respective variables are noted as functions of iteration n, e.g. wkj(n). ϱ is a small constant that determines the convergence criteria and max_iter is the maximum number of iterations.

3.1.6. Advantages of using Neural Networks

- Multi-layer feed-forward Neural Networks (NN) have shown to be a very powerful machine learning technique as they can be trained to approximate complex non-linear functions from high-dimensional input examples.

- Classically, standard Multi-Layer Perceptrons (MLP) has been utilized in pattern recognition systems to classify signatures coming from a separate feature extraction algorithm operating on the input data. However, the manual choice of the feature extraction algorithm and the features to classify is often empirical and therefore sub-optimal. Thus, a possible solution would be to directly apply the NN on the "raw" input data and let the training algorithm, e.g. Backpropagation, find the best feature extractors by adjusting the weights accordingly.

3.1.7. Disadvantages of using Neural Networks

- The problem with this approach is that when the input dimension is high, as in images, the number of connections, thus the number of free parameters are also high because each hidden unit would be fully connected to the input layer.

- This number may be in the order of several 10,000 or rather several 100,000 according to the application.

- The number of training examples, however, might be relatively small compared to the pattern dimension, which means that the NN would have a too high complexity and, thus, would tend to over fit the data.

- Its input layer has a fixed size and the input patterns have to be presented well aligned and/or normalized to this input window, which, in practice, is a rather complicated task. Thus, there is no built-in invariance w.r.t. small translations and local distortions.

- Finally, fully-connected NN architectures do not take into account correlations of neighboring input data. However, in pattern recognition problems there is generally a high amount of local correlation.

Thus, to avoid all the problems faces in Neural Networks as mentioned above, we go for a more sophisticated algorithm called Convolutional Neural Networks which we will discuss in the further sections. CNNs are an approach that tries to alleviate the above-mentioned problems. That is, they automatically learn local feature extractors, they are invariant to small translations and distortions in the input pattern, and they implement the principle of weight sharing which drastically reduces the number of free parameters and thus increases their generalization capacity compared to NN architectures without this property.

3.2. Convolutional Neural Networks

3.2.1. Design

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, RELU layer i.e. activation function, pooling layers, fully connected layers and normalization layers.

Description of the process as a convolution in neural networks is by convention. Mathematically it is a cross-correlation rather than a convolution (although cross-correlation is a related operation). This only has significance for the indices in the matrix, and thus which weights are placed at which index.

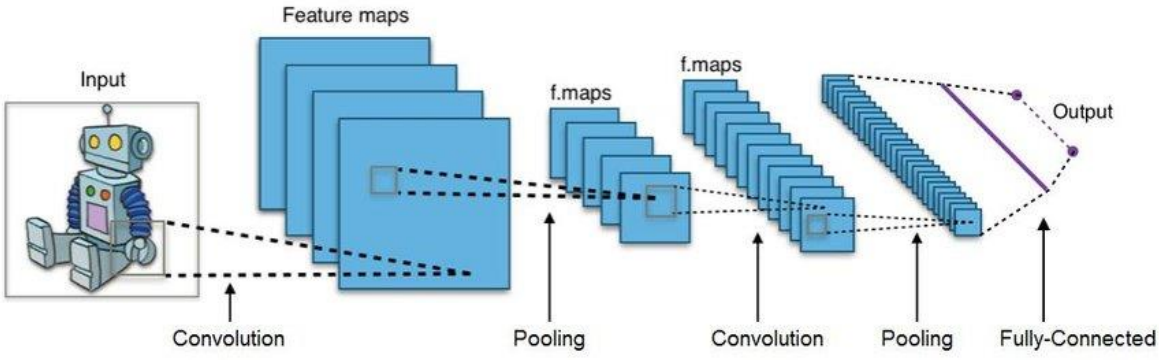


Fig. 3.4 Typical CNN Architecture

Convolutional:

Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli.

Although fully connected feed forward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable.

For instance, a fully connected layer for a (small) image of size 100 x 100 has 10000 weights for *each* neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi-layer neural networks with many layers by using backpropagation.

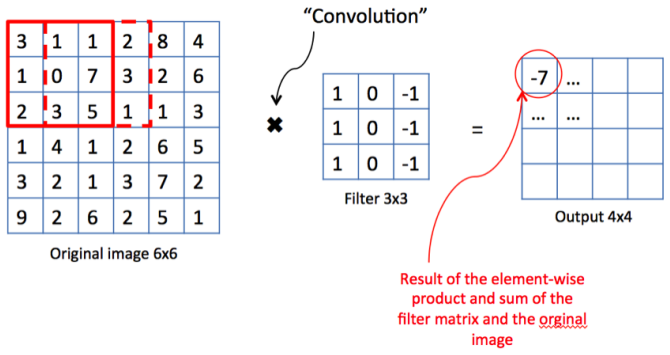


Fig. 3.5 Convolution

Pooling:

Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next

layer. For example, *max pooling* uses the maximum value from each of a cluster of neurons at the prior layer. Another example is *average pooling*, which uses the average value from each of a cluster of neurons at the prior layer.

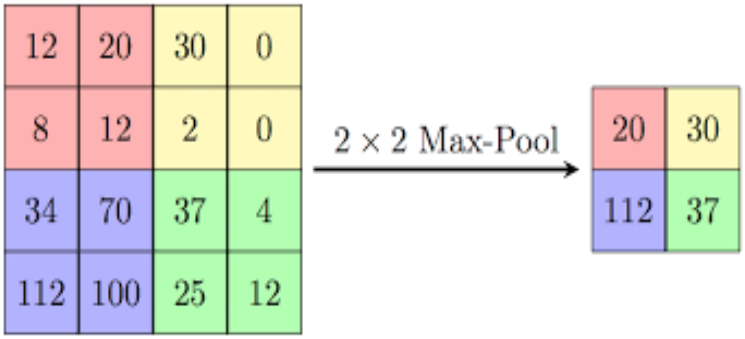


Fig. 3.6 Pooling

Fully Connected:

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

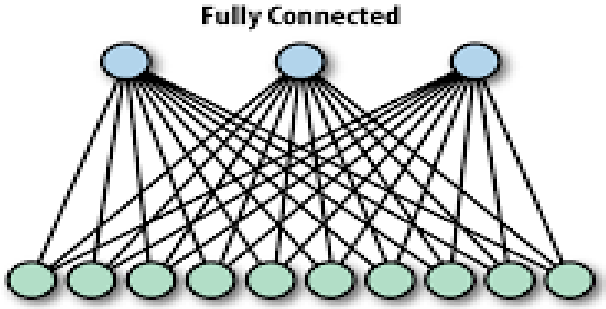


Fig. 3.7 Fully Connected

Receptive Field:

In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from *every* element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically, the subarea is of a square shape (e.g., size 5 by 5). The input area of a neuron is called its *receptive field*. So, in a

fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer.

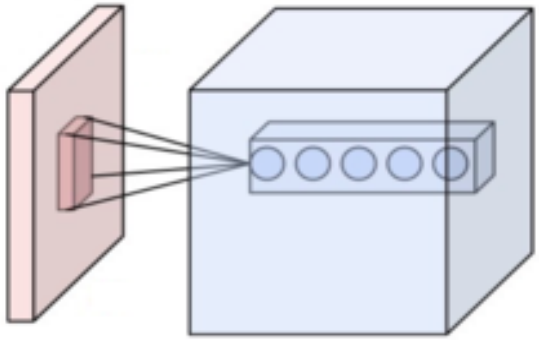


Fig. 3.8 Neurons(blue) connected to their Receptive Field(red)

Weights:

The function that is applied to the input values is specified by a vector of weights and a bias (typically real numbers). Learning in a neural network progresses by making incremental adjustments to the biases and weights. The vector of weights and the bias are called a *filter* and represent some feature of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons share the same filter. This reduces memory footprint because a single bias and a single vector of weights is used across all receptive fields sharing that filter, rather than each receptive field having its own bias and vector of weights.

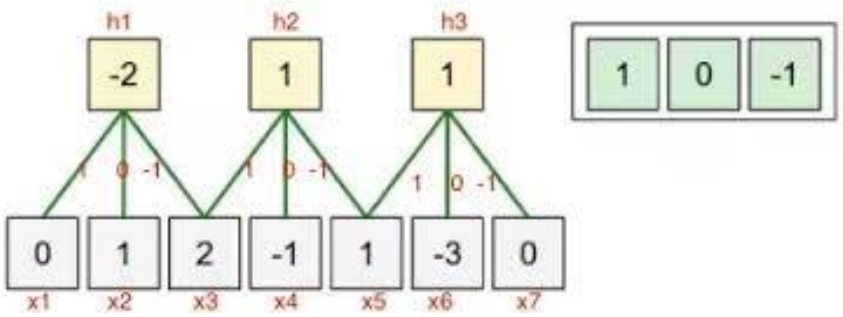


Fig. 3.9 Weight sharing in CNNs

3.2.2. Layers

3.2.2.1. Convolutional Layer

We now discuss the details of the neuron connectivity, their arrangement in space, and their parameter sharing scheme.

Local Connectivity:

When dealing with high-dimensional inputs such as images, as we saw above it is impractical to connect neurons to all neurons in the previous volume. Instead, we will connect each neuron to only a local region of the input volume. The spatial extent of this connectivity is a hyper parameter called the receptive field of the neuron (equivalently this is the filter size).

Example: suppose that the input volume has size [32x32x3], (e.g. an RGB CIFAR-10 image). If the receptive field (or the filter size) is 5x5, then each neuron in the Conv Layer will have weights to a [5x5x3] region in the input volume, for a total of 5*5*3 = 75 weights (and +1 bias parameter).

Spatial Arrangement:

Three hyperparameters control the size of the output volume: the **depth, stride** and **zero-padding**.

- **Depth:** It corresponds to the number of filters we would like to use, each learning to look for something different in the input.
- **Stride:** We must specify the **stride** with which we slide the filter. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2 then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially.
- **Zero-padding:** Sometimes it will be convenient to pad the input volume with zeros around the border. The size of this **zero-padding** is a hyperparameter. The

nice feature of zero padding is that it will allow us to control the spatial size of the output volumes.

Parameter Sharing:

A parameter sharing scheme is used in convolutional layers to control the number of free parameters. It relies on one reasonable assumption: if a patch feature is useful to compute at some spatial position, then it should also be useful to compute at other positions. In other words, denoting a single 2-dimensional slice of depth as a **depth slice**, we constrain the neurons in each depth slice to use the same weights and bias.

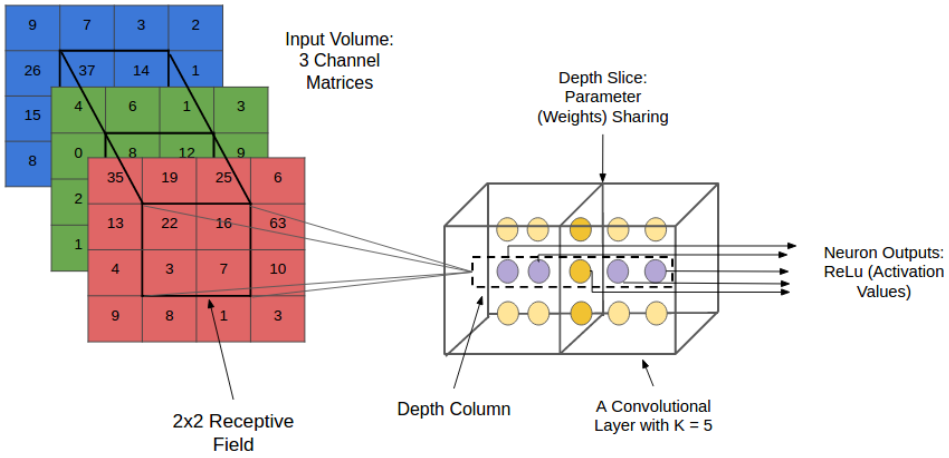


Fig. 3.10 Convolutional Layer

3.2.2.2. Pooling Layer

It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network, and hence to also control overfitting.

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 down samples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers.

The depth dimension remains unchanged. More generally, the pooling layer:

- Accepts a volume of size W1×H1×D1
- Requires two hyperparameters:
    - their spatial extent FF,
    - the stride SS,
- Produces a volume of size W2×H2×D2 where:
    - W2=(W1−F)/S+1
    - H2=(H1−F)/S+1
    - D2=D1
- Introduces zero parameters since it computes a fixed function of the input,
- For pooling layers, it is not common to pad the input using zero-padding.

General Pooling: In addition to max pooling, the pooling units can also perform other functions, such as average pooling or even L2-norm pooling. Average pooling was often used historically but has recently fallen out of favor compared to the max pooling operation, which has been shown to work better in practice.

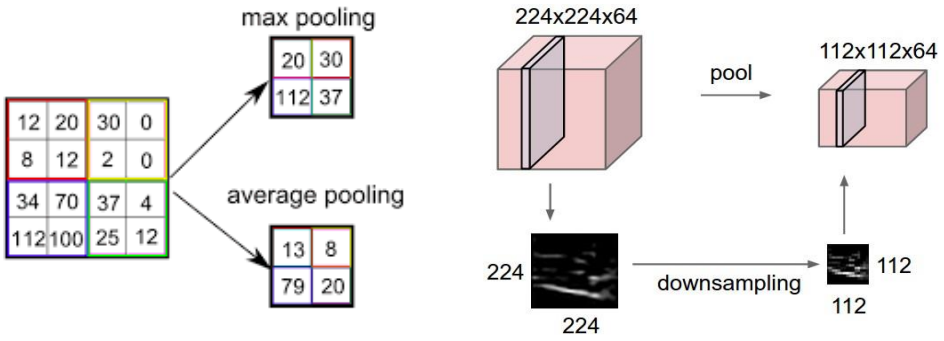


Fig. 3.11 Pooling Layers

In our project, we have used the max pooling layer.

3.2.2.3. Activation Functions

It's just a function that you use to get the output of node. It is also known as **Transfer Function**. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function).

The Activation Functions can be basically divided into 2 types-

- Linear Activation Function
- Non-linear Activation Functions

Linear or Identity Activation Function:

As you can see the function is a line or linear. Therefore, the output of the functions will not be confined between any range.

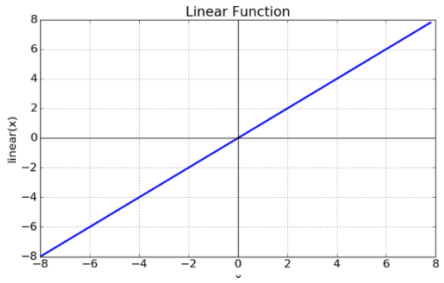


Fig. 3.12 Linear Activation Function

Non-Linear Activation Function:

The Nonlinear Activation Functions are the most used activation functions. Nonlinearity helps to makes the graph look something like this;

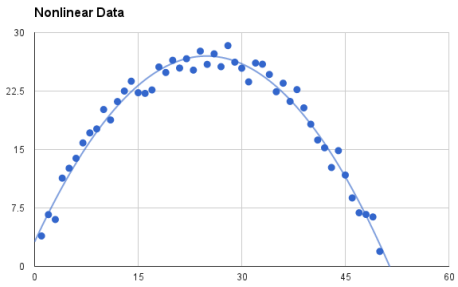


Fig. 3.13 Non-Linear Activation Function

The Nonlinear Activation Functions are mainly divided on the basis of their range or curves:

1) Sigmoid or Logistic Activation Function

The Sigmoid Function curve looks like a S-shape.The main reason why we use sigmoid function is because it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output.Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.The function is differentiable.That means, we can find the slope of the sigmoid curve at any two points.The function is monotonic but function's derivative is not.The logistic sigmoid function can cause a neural network to get stuck at the training time.

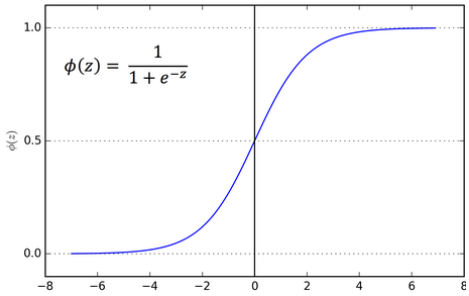


Fig. 3.14 Sigmoid Activation Function

2) Tanh or hyperbolic tangent Activation Function

Tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped). The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph. The function is differentiable. The function is monotonic while its derivative is not monotonic. The tanh function is mainly used classification between two classes.

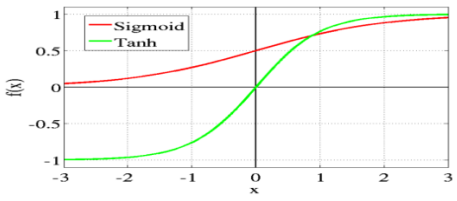


Fig. 3.15 Tanh Activation Function

3) ReLU (Rectified Linear Unit) Activation Function

The ReLU is the most used activation function in the world right now since it is used in almost all the convolutional neural networks which applies the non-saturating activation function **f(x)=max(0,x)**. It effectively removes negative values from an activation map by setting them to zero.

The function and its derivative **both are monotonic**.

But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately. We have used the ReLU activation function in our project.

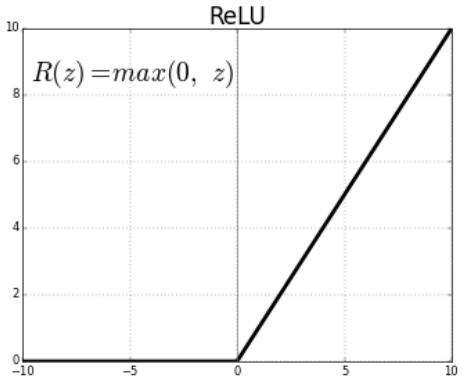


Fig. 3.16 ReLU Activation Function

4) Leaky ReLU:

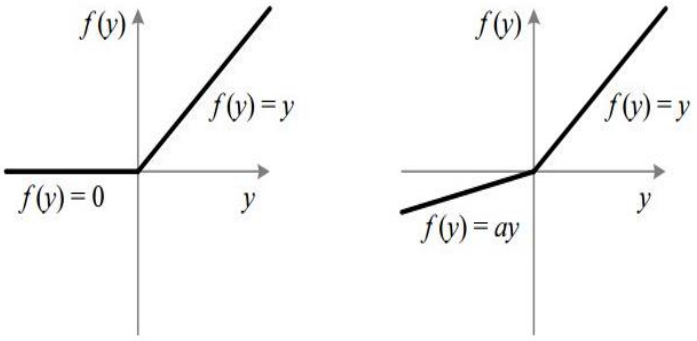
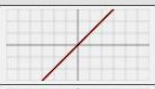
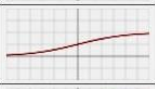
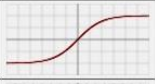
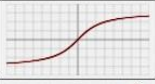
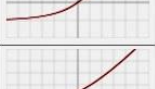


Fig. 3.17 ReLU v/s Leaky ReLU

The leak helps to increase the range of the ReLU function. Usually, the value of **a** is 0.01 or so, when **a is not 0.01** then it is called **Randomized ReLU**. Therefore, the **range** of the Leaky ReLU is (-infinity to infinity).Both Leaky and Randomized ReLU functions are monotonic in nature. Also, their derivatives also monotonic in nature.

Table 3.1 Different Activation Functions

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

3.2.3. Classification

In **machine learning** and statistics, **classification** is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.

3.2.3.1. Flatten

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. This phenomenon is

called ‘Flattening’. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the **Logistic Regression Classification** technique.

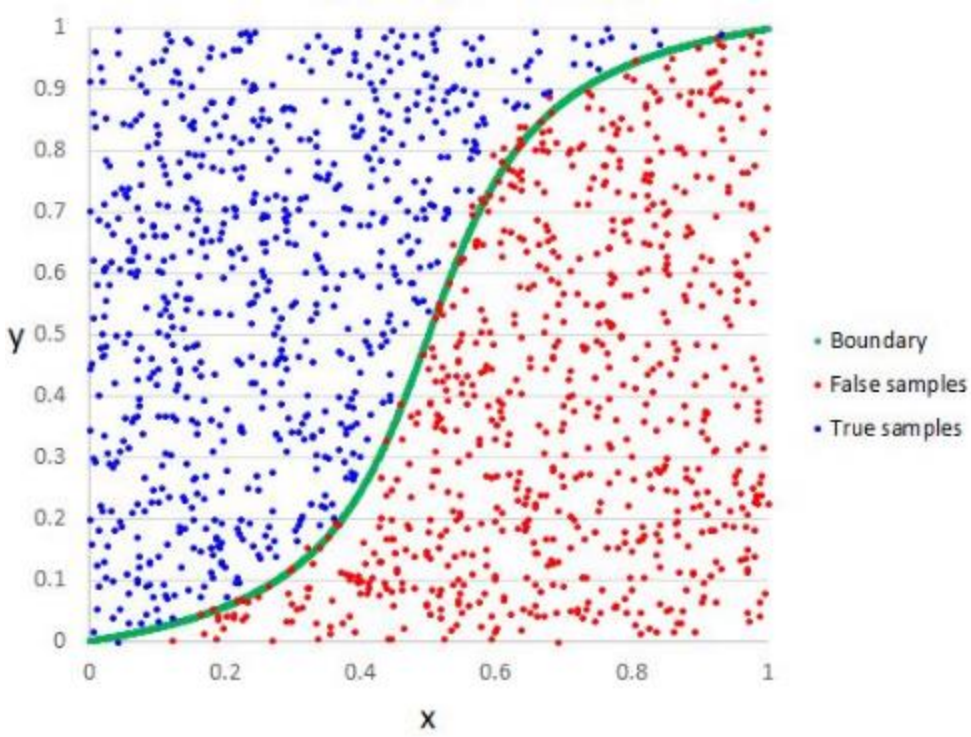


Fig. 3.18 Classification Using Log. Reg. Function

3.2.3.2. Fully-Connected

Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular (non-convolutional) artificial neural networks as shown in Fig. 3.7 Their activations can thus be computed as an affine transformation, with matrix multiplication followed by a bias offset (vector addition of a learned or fixed bias term).

3.2.3.3. Logistic Regression Classifier

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

Logistic regression uses an equation as the representation, very much like linear regression.

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary value (0 or 1) rather than a numeric value.

Below is an example logistic regression equation:

$$y = e^{\wedge} (b0 + b1*x) / (1 + e^{\wedge} (b0 + b1*x))$$

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

The actual representation of the model that you would store in memory or in a file is the coefficients in the equation.

Logistic regression models the probability of the default class (e.g. the first class).

For example, if we are modeling people's sex as male or female from their height, then the first class could be male and the logistic regression model could be written as the probability of male given a person's height, or more formally:

$$P(sex=male|height)$$

Written another way, we are modeling the probability that an input (X) belongs to the default class (Y=1), we can write this formally as:

$$P(X) = P(Y=1|X)$$

Note that the probability prediction must be transformed into a binary value (0 or 1) in order to actually make a probability prediction. More on this later when we talk about making predictions.

Logistic regression is a linear method, but the predictions are transformed using the logistic function. The impact of this is that we can no longer understand the predictions as a linear combination of the inputs as we can with linear regression, for example, continuing on from above, the model can be stated as:

$$p(X) = e^{(b0 + b1*X)} / (1 + e^{(b0 + b1*X)})$$

we can turn around the above equation as follows (remember we can remove the e from one side by adding a natural logarithm (ln) to the other):

$$\ln(p(X) / 1 – p(X)) = b0 + b1 * X$$

This is useful because we can see that the calculation of the output on the right is linear again (just like linear regression), and the input on the left is a log of the probability of the default class.

This ratio on the left is called the odds of the default class (it's historical that we use odds, for example, odds are used in horse racing rather than probabilities). Odds are calculated as a ratio of the probability of the event divided by the probability of not the event, e.g. 0.8/(1-0.8) which has the odds of 4. So, we could instead write:

$$\ln(odds) = b0 + b1 * X$$

Because the odds are log transformed, we call this left-hand side the log-odds or the probit. It is possible to use other types of functions for the transform (which is out of scope_, but as such it is common to refer to the transform that relates the linear regression equation to the probabilities as the link function, e.g. the probit link function.

We can move the exponent back to the right and write it as:

$$odds = e^{(b0 + b1 * X)}$$

All of this helps us understand that indeed the model is still a linear combination of the inputs, but that this linear combination relates to the log-odds of the default class.

In our project we have used the Linear + Log-Regression Classifier.

3.2.3.4. Softmax Classifier

The softmax activation is normally applied to the very last layer in a neural net, instead of using ReLU, sigmoid, tanh, or another activation function. The reason why softmax is useful is because it converts the output of the last layer in your neural network into what is essentially a probability distribution. If you look at the origins of the cross-entropy loss function in information theory, you will know that it "expects" two probability distributions as input. That's why softmax output with cross entropy loss is very common.

Just to reiterate; softmax is typically viewed as an activation function, like sigmoid or ReLU. Softmax is NOT a loss function, but is used to make the output of a neural net more "compatible" with the cross entropy or negative log likelihood loss functions.

The Softmax classifier is given by:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Which allows us to interpret the outputs as probabilities, while cross-entropy loss is what we use to measure the error at a softmax layer, and is given by

$$L(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} H(p_n, q_n) = -\frac{1}{N}\sum_{n=1}^{N} \left[y_n \log \hat{y}_n + (1-y_n) \log (1-\hat{y}_n) \right],$$

3.3. Pose-Invariant Face Recognition:

According to C. Ding and D. Tao-A comprehensive survey on pose-invariant face recognition, existing PIFR methods can be classified into four categories including:

1) multi-view subspace learning, 2) pose-invariant feature extraction, face synthesis, and 3) a hybrid approach of the above three. Our work belongs to the second category of extracting pose-invariant features. Some previous work in this category treat each pose separately by learning different models for face images with different poses.

Since pose variation is the most challenging one among other non-identity variations, and the proposed m-CNN already classifies all images into different pose groups, we propose to apply divide-and-conquer to CNN learning.

Specifically, we develop a novel pose-directed multi-task CNN (pCNN) where the pose labels can categorize the training data into three different pose groups, direct them through different routes in the network to learn pose-specific identity features in addition to the generic identity features.

Similarly, the loss weights for extracting these two types of features are learnt dynamically in the CNN framework. During the testing stage, we propose a stochastic routing scheme to fuse the generic identity features and the pose-specific identity features for face recognition that is more robust to pose estimation errors.

This work utilizes all data in FERET, ATT, Webcam and a self-curated dataset, i.e., faces with the full range of pose variations, as the main experimental dataset — ideal for studying MTL for Pose-invariant face recognition. Since the ground truth label of the side task is unavailable, we use the estimated poses as labels for training. In summary, we make four contributions:

- We explore how and why pose estimations can help face recognition;
- We propose a dynamic-weighting scheme to learn the loss weights for different tasks automatically in the CNN framework.
- We develop a pose-directed multi-task CNN to handle pose variation.
- We also developed an algorithm in which we directly map faces using PCA and show the frontal face of the subject irrespective of the pose.

3.4. CNN Block Diagram

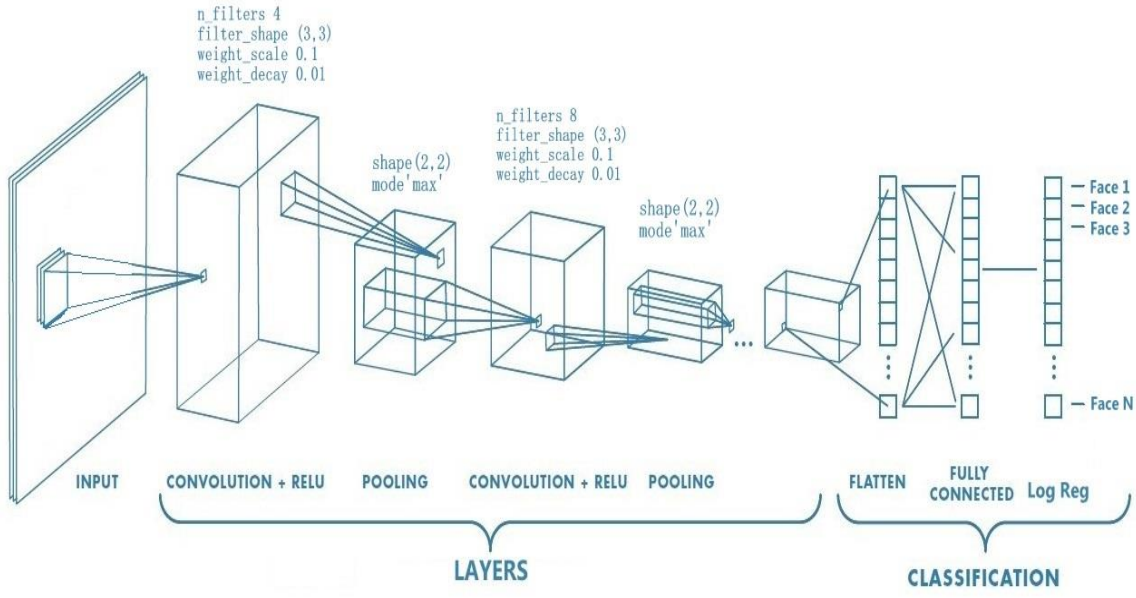


Fig. 3.19 CNN Block Diagram

Fig 3.4 shows the block diagram of a typical CNN and the parameters used in our project.

3.4.1. Specification of the Architecture

- In the first stage of architecture we used convolution layer with 4 filters and shape 3 x 3 so that we can grab minute details in face which will be helpful for the classifier to segment different faces.
- In RELU we generally suppress the negative values which are generated in convolution layer because practically we don't have negative pixel values.
- In the second stage we used 2 x 2 pooling layer with max mode so that we reduce the image size and only obtain high frequency components.
- Same layers can be added again and again with change in parameters for better performance.
- In flattening, we convert 2D images to vectors which represent the features of a person.
- Finally, we use Logistic Regression Layer for classification.

3.4.2. Advantages of CNN

- CNN (Convolutional Neural Networks) is more efficient in extracting the minute features from picture which is very much important in classifying human faces.
- We reduce the size of intermediate images required for processing using pooling layers.
- Convolution simplifies computation to a great extent without losing the essence of the data.
- Minimize computation compared to a regular neural network.
- We can use pre trained models using transfer learning options thus reducing time for training the model.
- We can add multiple layers easily with little modification for better classification of images.

3.4.3. Limitations of CNN

- Large number of training data and annotations are needed, which may not be practical in some problems.
- If you don't have a good GPU they are quite slow to train (for complex tasks).
- A convolution is a significantly slower operation than, say maxpool, both forward and backward. If the network is pretty deep, each training step is going to take much longer.
- As the data increases, time complexity and memory footprint of the architecture increases.
- Reconstruction of intermediate images is not possible.

3.4.4. Case Studies

There are several architectures in the field of Convolutional Networks that have a name. The most common are:

- **LeNet**. The first successful applications of Convolutional Networks were developed by Yann LeCun in 1990's. Of these, the best known is the LeNet architecture that was used to read zip codes, digits, etc.

- **AlexNet**. The first work that popularized Convolutional Networks in Computer Vision was the AlexNet, developed by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton. The AlexNet was submitted to the ImageNet ILSVRC challenge in 2012 and significantly outperformed the second runner-up (top 5 error of 16% compared to runner-up with 26% error). The Network had a very similar architecture to LeNet, but was deeper, bigger, and featured Convolutional Layers stacked on top of each other (previously it was common to only have a single CONV layer always immediately followed by a POOL layer).

- **ZF Net**. The ILSVRC 2013 winner was a Convolutional Network from Matthew Zeiler and Rob Fergus. It came to be known as the ZFNet (short for Zeiler & Fergus Net). It was an improvement on AlexNet by tweaking the architecture hyperparameters, in particular by expanding the size of the middle convolutional layers and making the stride and filter size on the first layer smaller.

- **GoogLeNet**. The ILSVRC 2014 winner was a Convolutional Network from Szegedy et al. from Google. Its main contribution was the development of an *Inception Module* that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M). Additionally, this paper uses Average Pooling instead of Fully Connected layers at the top of the ConvNet, eliminating a large number of parameters that do not seem to matter much. There are also several follow-up versions to the GoogLeNet, most recently Inception-v4.

3.5. Conclusion

In this chapter we study about the evolution of neural networks, the architecture of Convolutional Neural Networks and the block diagram.

Chapter Four

TOOLS & LIBRARIES USED

4.1. Introduction

In this project we have used a wide range of tools, applications and libraries. Most of these tools are open source tools and contains in-built functions for easy development of programs and applications.

4.2. Programming Languages

We mostly used two languages in the development of architecture and testing various methods related to pose. They are:

4.2.1. Python 3.7

Python is a popular platform used for research and development of production systems. It is a vast language with number of modules, packages and libraries that provide multiple ways of achieving a task.

Python and its libraries like NumPy, SciPy, Scikit-Learn, Matplotlib are used in data science and data analysis. They are also extensively used for creating scalable machine learning algorithms. Python implements popular machine learning techniques such as Classification, Regression, Recommendation, and Clustering.

To understand machine learning, you need to have basic knowledge of Python programming. In addition, there are a number of libraries and packages generally used in performing various machine learning tasks as listed below:

- **numpy**- is used for its N-dimensional array objects.
- **pandas** – is a data analysis library that includes data frames.
- **matplotlib** – is a 2D plotting library for creating graphs and plots.
- **scikit-learn** - the algorithms used for data analysis and data mining tasks.

Python offers ready-made framework for performing data mining tasks on large volumes of data effectively in lesser time. It includes several implementations achieved through algorithms such as linear regression, logistic regression, Naïve Bayes, k-means, K nearest neighbor, and Random Forest.

4.2.2. Matlab 2018

MATLAB (matrix laboratory) is a fourth-generation high-level programming language and interactive environment for numerical computation, visualization and programming. MATLAB is developed by Math Works.

It allows matrix manipulations; plotting of functions and data; implementation of algorithms; creation of user interfaces; interfacing with programs written in other languages, including C, C++, Java, and FORTRAN; analyze data; develop algorithms; and create models and applications.

It has numerous built-in commands and math functions that help you in mathematical calculations, generating plots, and performing numerical methods.

MATLAB's Power of Computational Mathematics

MATLAB is used in every facet of computational mathematics. Following are some commonly used mathematical calculations where it is used most commonly:

- Dealing with Matrices and Arrays
- 2-D and 3-D Plotting and graphics
- Linear Algebra
- Algebraic Equations
- Non-linear Functions
- Statistics
- Data Analysis

- Calculus and Differential Equations
- Numerical Calculations
- Integration
- Transforms
- Curve Fitting
- Various other special functions

Features of MATLAB

Following are the basic features of MATLAB −

- It is a high-level language for numerical computation, visualization and application development.
- It also provides an interactive environment for iterative exploration, design and problem solving.
- It provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.
- It provides built-in graphics for visualizing data and tools for creating custom plots.
- MATLAB's programming interface gives development tools for improving code quality maintainability and maximizing performance.
- It provides tools for building applications with custom graphical interfaces.
- It provides functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET and Microsoft Excel.

Uses of MATLAB

MATLAB is widely used as a computational tool in science and engineering encompassing the fields of physics, chemistry, math and all engineering streams. It is used in a range of applications including:

- Signal Processing and Communications
- Image and Video Processing
- Control Systems
- Test and Measurement
- Computational Finance
- Computational Biology

4.3. Applications Used

We used two applications during the course of our project:

4.3.1. Irfan View

This application is one stop solution for all problems related to format of the images and basic processing of the images. Features of this application are:

- 32 and 64 bit version
- Many supported file formats (.jpg, .png, .pgm, raw, gif etc)
- Thumbnail/preview option
- Paint option - to draw lines, circles, arrows, straighten image etc.
- Lossless JPG rotation, crop and EXIF date change (also in batch mode)
- Slideshow (save slideshow as EXE/SCR or burn it to CD)
- Support for Adobe Photoshop Filters

- Batch conversion (with advanced image processing of all files)
- Change color depth
- Scan (batch scan) support
- Cut/crop
- Add overlay text/image (watermark)
- Effects (Sharpen, Blur, Adobe 8BF, Filter Factory, Filters Unlimited, etc.)
- Screen Capturing

4.3.2. Google Colaboratory

Google Colab is a free cloud service and now it supports free GPU.

We can:

- Run complex neural network codes which take hours to execute with very high speed.
- Develop deep learning applications using popular libraries such as Keras, TensorFlow, Pytorch and OpenCV.
- Now it's not a problem with operating system compatibility and libraries because most of them are in-built in Google Colab.
- There is no need to utilize our resources (i.e) we can do other work on our computer while your code is running.

The most important feature that distinguishes Colab from other free cloud services is; **Colab** provides GPU and is totally free.

4.4. Conclusion

In this chapter we discussed about the various tools and their functions used in the developments of our architecture.

Chapter Five

RESULTS & PROJECT ANALYSIS

5.1. Databases Used

CNN technique is tested on the ATT, FERET, WEBCAM, POSE database and a real time database of my college friends.

5.1.1. ATT Database

ORL face database is composed of 400 images of size 112 x 92. There are 40 persons, 10 images per each person but we only used 10 people's data for simulation. The faces are in an upright position in frontal view, with a slight left-right rotation.

Fig. 5.1 ATT Database Images

5.1.2. FERET Database

This database has 10 subjects and each subject has 10 images with various poses from -60º to 60º approximately. Size of the images is 112 x 92.

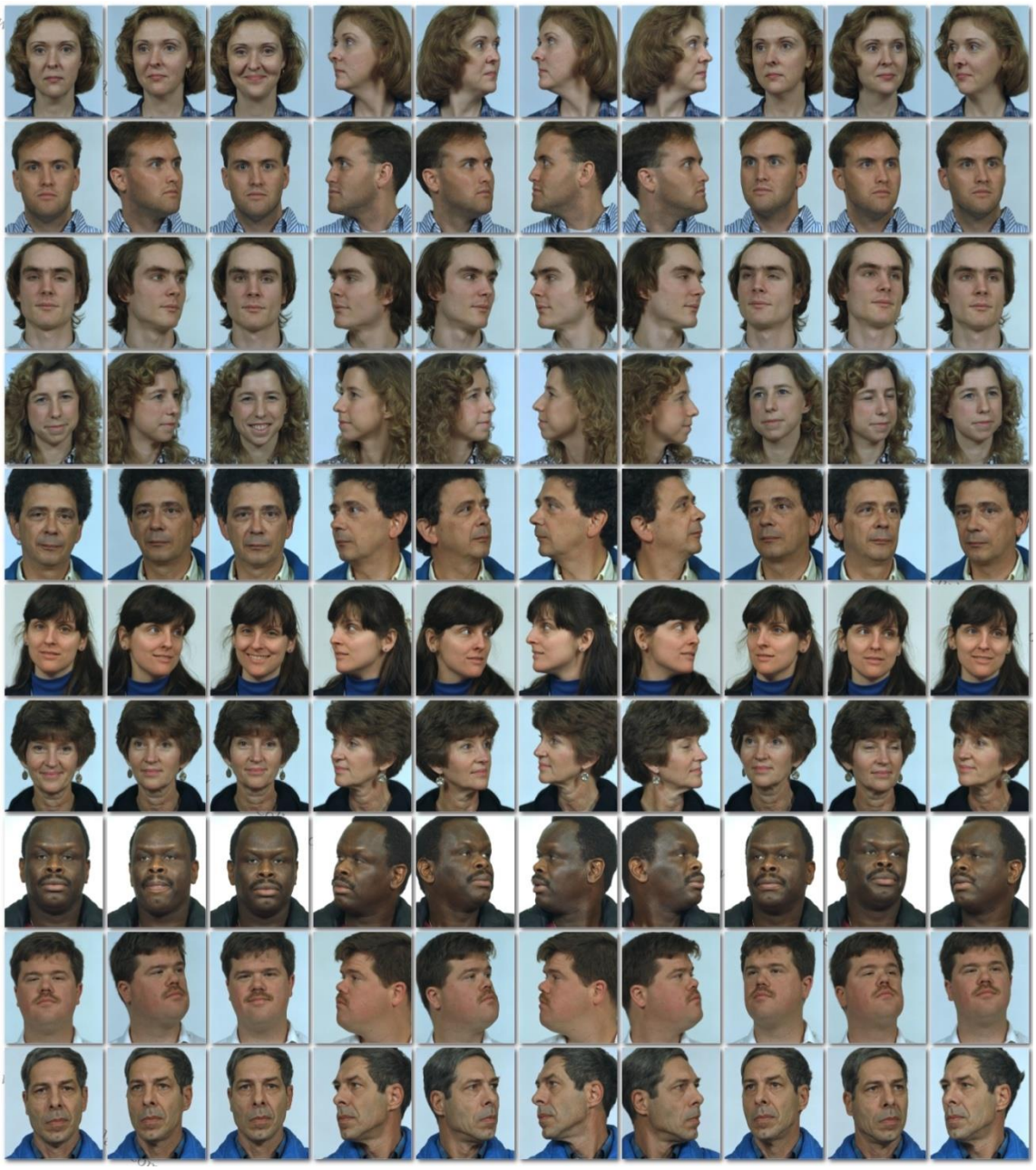


Fig. 5.2 FERET Database Images

5.1.3. WEBCAM Database

This database has 10 subjects and each subject has 8 images with various poses from -60º to 60º approximately and these pictures are affected by irregular illumination. Size of the images is 100 x 100.

Fig. 5.3 Webcam Database Images

5.1.4. POSE Database

This database is a subset of FERET Database which was created to test code in matlab using PCA and pose estimator which has 10 subjects and each subject has 5 images with 0º, 30º, -30º, 60º and -60º. Size of the images is 112 x 92.The below database is free from irregular illuminations or occlusions.

Fig. 5.4 Pose Database Images

These images are handpicked from database and resized. With PCA and pose estimator we got accuracy of 100% but when the data set increases, it might affect the accuracy and execution time.

Thought it's not a part of CNN we tried to implement alternative codes on matlab in order to reduce the time complexity while dealing with poses.

Not only mapping a pose invariant face of the given subject but also we display the estimated pose of the subject.

5.1.5. Own Database

We never know the robustness of the architecture until and unless we train and test our own database. As a part of it we collected images of 10 different classmates with various poses varying from -90º to 90º. There is no illumination or occlusion problem for the data set. Size of the images is 112 x 92.With this data set we achieved maximum accuracy of 80% while testing.

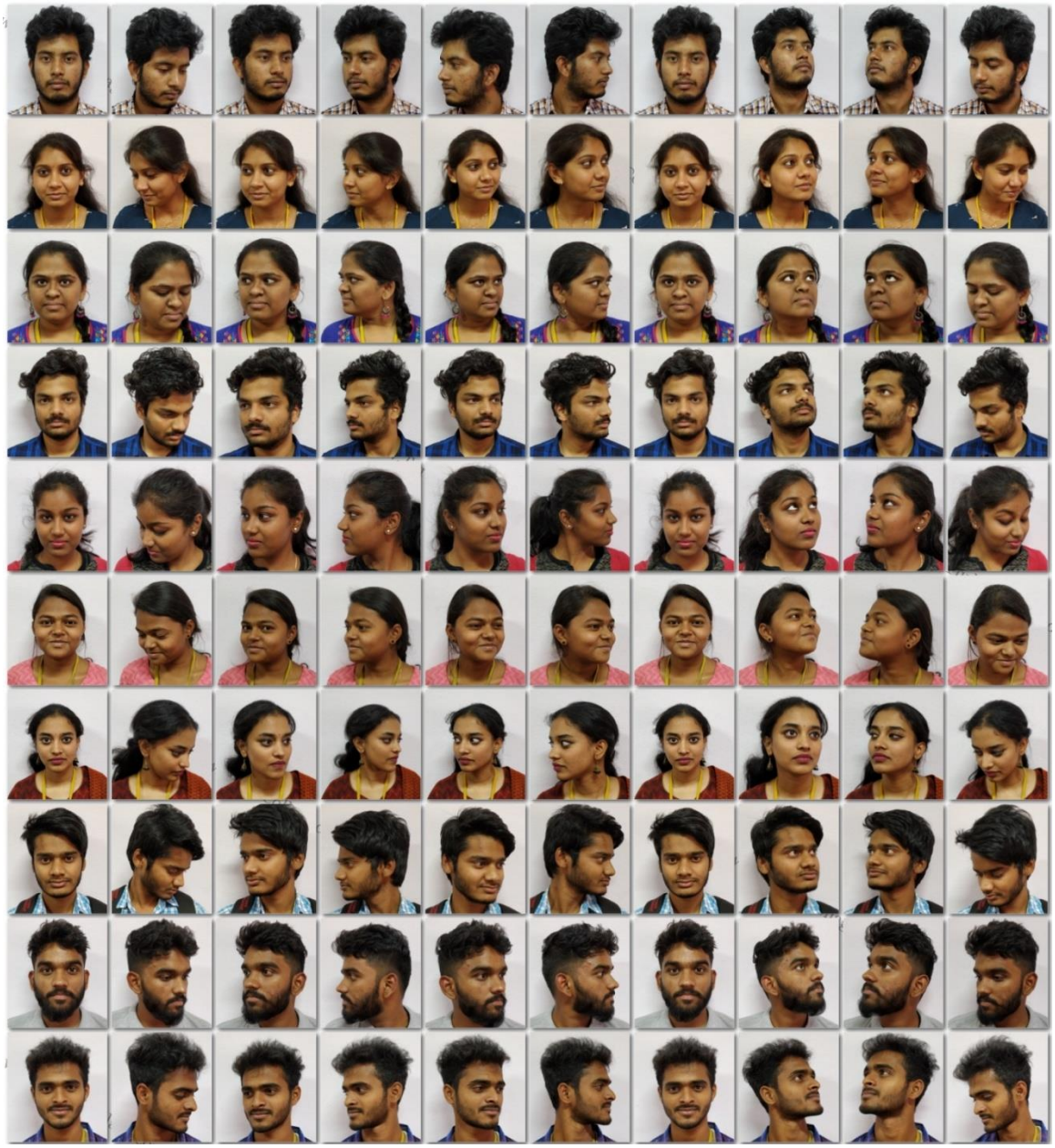


Fig. 5.5 Own Database Images

5.2. Training Results

Training is the most important part in CNN where all the layers, forward propagation and backward propagation come into picture. So, we took training recognition and training error at every iteration and plotted line graph for easy analysis.

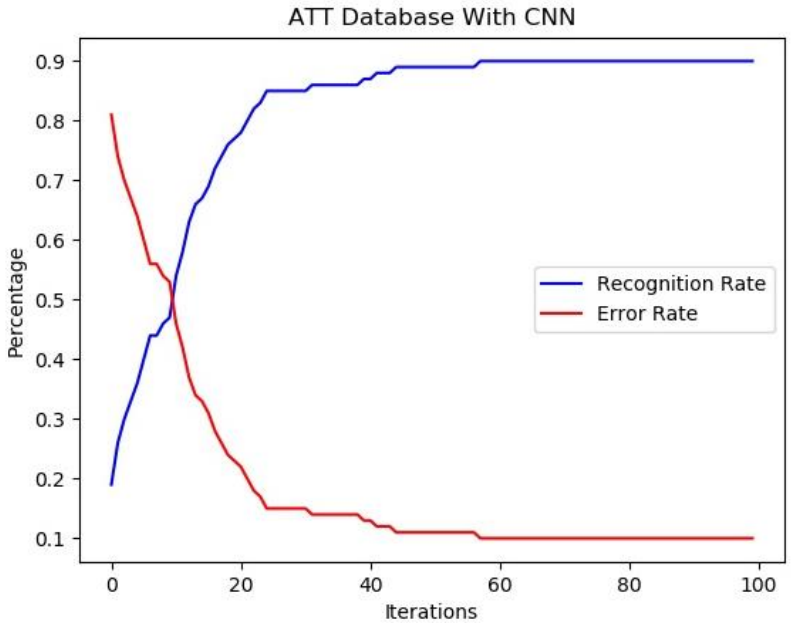


Fig. 5.6 ATT Training Result

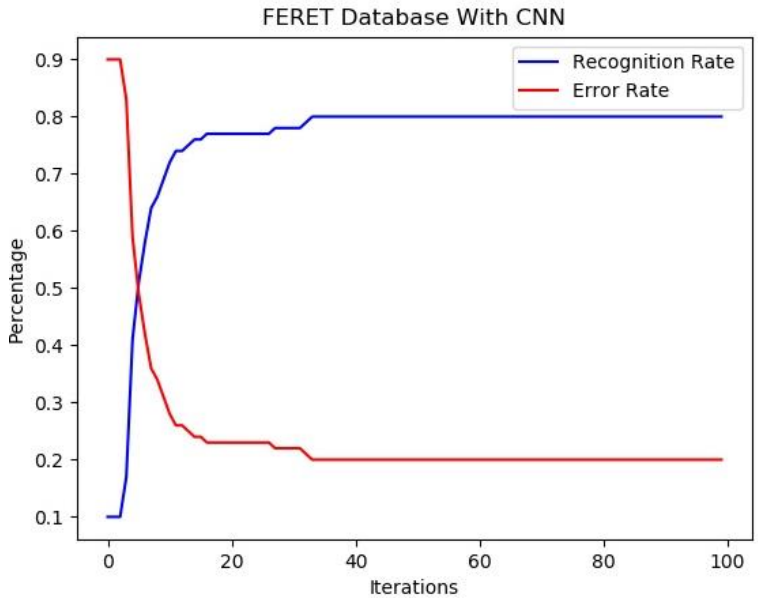


Fig. 5.7 FERET Training Result

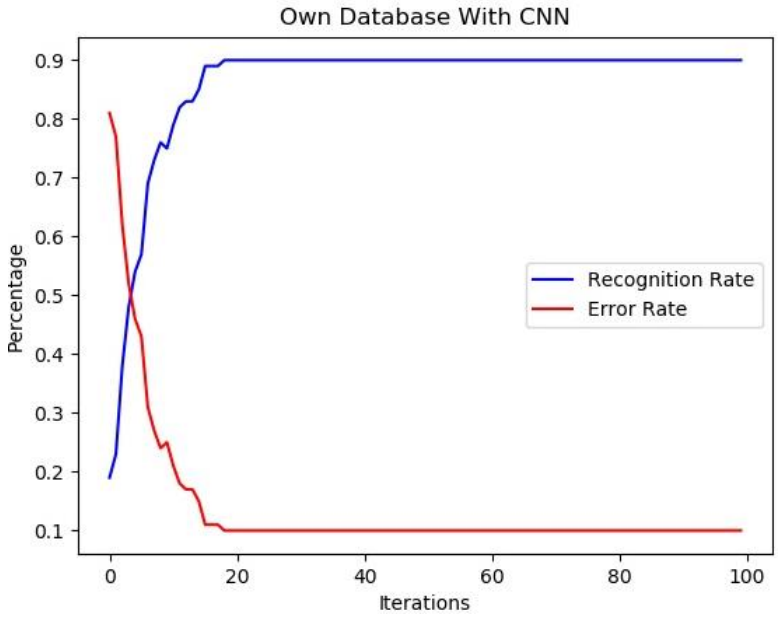


Fig. 5.8 OWN Database Training Result

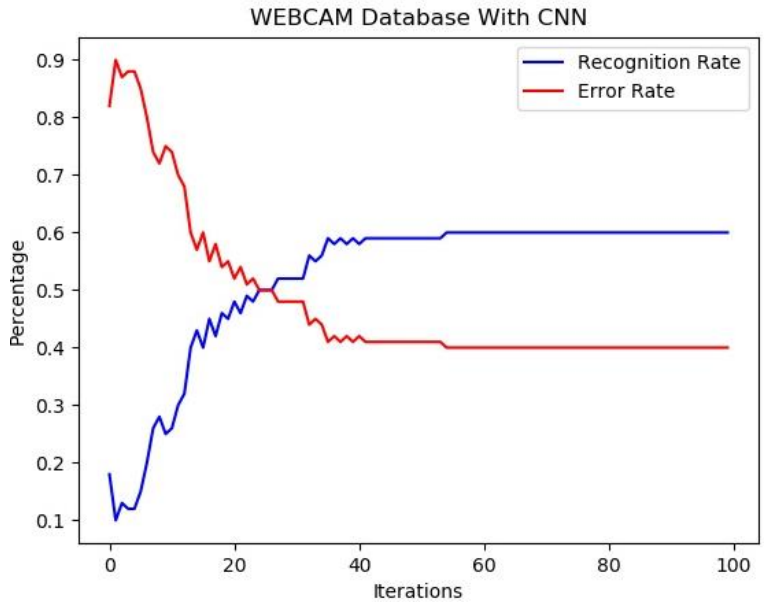


Fig. 5.9 WEBCAM Database Training Result

It is clearly visible that training of Webcam Database is low compared to other databases because we designed our architecture specifically for poses. But, webcam database is highly affected by illumination variation as you can see in the above database images. For FERET and OWN database there is a rapid growth in the training because there are only pose variations.

5.3. Testing Results

After training the CNN model we test the model using images of the same database to test its efficiency of recognition. Here, we compare the testing results of various databases.

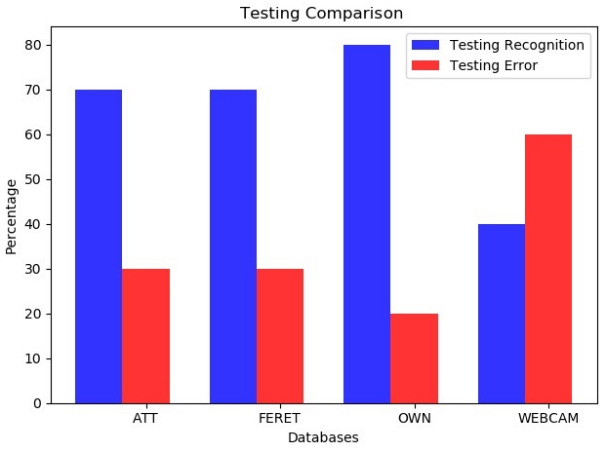


Fig. 5.10 Testing Comparison

Here, we compare the efficiency of the architecture for various databases.

Table 5.1 Overall Comparison

Database	Subjects	Images	Size	Iterations	Time (Min)	Training Accuracy	Testing Accuracy
ATT	5	10	112 x 92	50	50	56 %	20 %
			112 x 92	100	100	77 %	40 %
	10	10	112 x 92	50	100	65 %	50 %
			112 x 92	100	200	90 %	70 %
FERET	5	10	112 x 92	50	50	66 %	40 %
			112 x 92	100	100	78 %	60 %
	10	10	112 x 92	50	100	73 %	50 %
			112 x 92	100	200	82 %	70 %
WEB CAM	5	10	100 x 100	50	45	33 %	20 %
			100 x 100	100	90	42 %	40 %
	10	10	100 x 100	50	90	50 %	30 %
			100 x 100	100	180	55 %	40 %
OWN	5	10	112 x 92	50	50	70 %	40 %
			112 x 92	100	100	84 %	60 %
	10	10	112 x 92	50	100	86 %	60 %
			112 x 92	100	200	91 %	**80 %**

We can see that iterations are directly proportional to testing and training accuracy. We achieved maximum accuracy with our own database for 100 iterations and 10 subjects.

5.4. Existing Methods Comparison

CNNs are currently trending in computer vision world; we took some of the current architectures and created a comparison table below:

Table 5.2 Existing Methods

Method	Mean Accuracy
ROI + Gist [36]	26.1
DPM [30]	30.4
Object Bank [24]	37.6
RBow [31]	37.9
BoP [21]	46.1
miSVM [25]	46.4
D-Parts [40]	51.4
IFV [21]	60.8
MLrep [9]	64.0
CNN-SVM	58.4
CNNaug-SVM	69.0
CNN (AlexConvNet)+Multiscale Pooling [16]	68.9
Our CNN Architecture [8]	**70.0**

5.5. Alternative Methods

CNN and other neural networks are often time consuming but they provide accurate results. Thus, we can go for alternative methods like Pose Invariant PCA or Pose Estimator which are quite fast.

5.5.1. Generation of Frontal Face Using PCA

Here we use standard PCA with various pose dataset as shown in Fig. 5.4 and then generate frontal face using the index of the person.

Fig. 5.11 Output of PCA

5.5.2. Pose Estimation Using Patch Based Recognition

In this approach we break the test image into a non-overlapping regular grid of patches. Each is treated separately and provides independent information about the true pose. At the core of this algorithm is a predefined library of object instances. The library can be considered as a palette from which image patches can be taken. We exploit the relationship between the patches in the test image and the patches in the library to estimate the face pose.

Fig. 5.12 Output of Pose Estimator

5.6. Hardware Comparison

Time complexity of algorithm is the most important part in execution. Unfortunately many neural network and machine learning algorithms have more time complexity. So, we generally go for high end processors and GPUs.

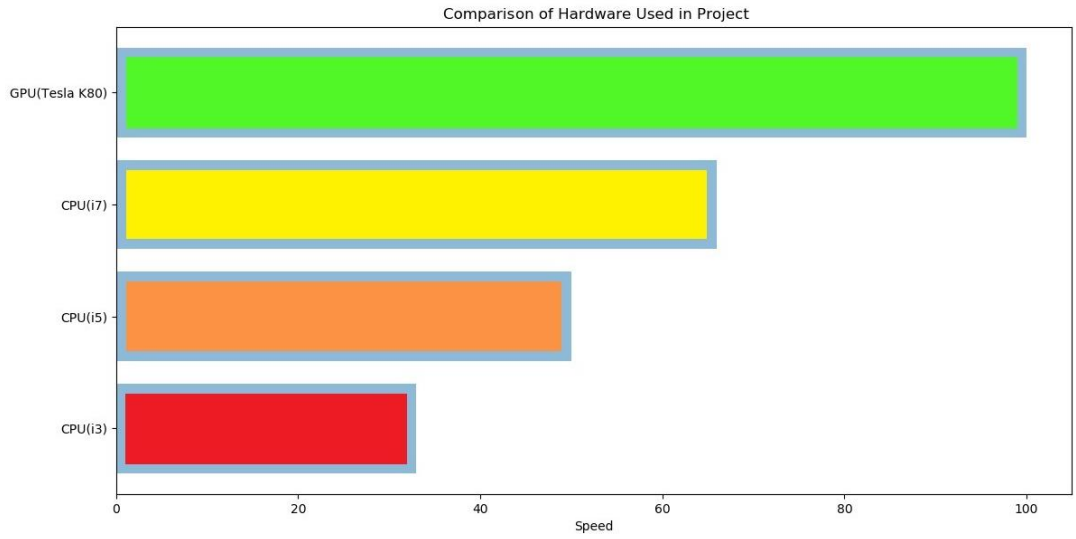


Fig. 5.13 Hardware comparison

CONCLUSION

In this thesis, various approaches to classify the poses of different face images are proposed and investigated. The experimental results show that CNN has the capability to capture the semantic contents from the image spectra. The proposed approaches provide a novel framework for pre-processing steps prior to image classification. The proposed techniques are evaluated through extensive simulations on large standard databases of different categories of pose invariant facial images. The enormous scope of spectra of pose invariant facial images is analyzed.

Although great progress in PIFR has been achieved, there is still much room for improvement, and the performance of existing approaches needs to be further evaluated on real-world databases. To meet the requirement of practical face recognition applications, we propose the following design criteria as a guide for future development.

- **Fully automatic:** The PIFR algorithms should work autonomously, i.e., require no manual facial landmark annotations or pose estimation, etc.
- **Full range of pose variation:** The PIFR algorithms should cover the full range of pose variations that might appear in the face image, including the yaw, pitch, and combined yaw and pitch. In particular, recognition of profile faces is very difficult and largely under-investigated. For pose normalization-based methods, the difficulty lies in the larger error of shape and pose estimation for profile faces.
- **Recognition from a single image:** The PIFR algorithms should be able to recognize a single non frontal face utilizing a single gallery image per person. This is the most challenging but also the most common setting for real-world applications.
- **Robust to combined facial variations:** As explained in Chapter 1, the pose variation is often combined with illumination, expression, and image quality variations. A practical PIFR algorithm should also be robust to combined facial appearance variations.

- •**Matching between two arbitrary poses:** The most common setting for existing PIFR algorithms is to identify non-frontal probe faces from frontal gallery images. However, it is desirable to be able to match two face images with arbitrarily different poses, for both identification and verification tasks. One extreme example is to match a left-profile face to a right-profile face. However, facial symmetry does not exactly hold true for high-resolution images where fine facial textures are clear.
- **Reasonable requirement of labeled training data:** Although a large amount of labeled multi pose training data helps to promote the performance of pose-robust feature extraction based PIFR algorithms, it is not necessarily available because labeled multi-pose data are difficult to collect in many practical applications. Possible solutions may incorporate making use of 3D shape priors of the human head and combining unsupervised learning algorithms.
- **Efficient**: The PIFR algorithms should be efficient enough to realize the requirement of practical applications, e.g., video surveillance and digital entertainment. Therefore, approaches that are free from complicated optimization operations in the testing time are preferable.

FUTURE SCOPE

- Besides obtaining a robust technique like CNN to classify pose variations in facial images, we are also trying to incorporate volumetric regression so that we can generate different poses with single image and then train them using the proposed architecture.
- Since we know that the problems and challenges faced in facial recognition are inevitable, we were specifically dealing only with poses in this project. But in the near future we can expect many more advancements in this field considering the rapid growth in the field of Machine Learning.
- CNN can also be used for large-scale video classification.
- Besides CNN, Deep learning techniques like Recurrent Neural Networks can be used to implement phase-sensitive and recognition-boosted speech separation.

REFERENCES

1. G. Arulampalam and A. Bouzerdoum. A generalized feedforward neural network architecture for classification and regression. Neural Networks, 16:561–568, 2003.
2. S. O. Ba and J. M. Odobez. A probabilistic framework for joint head tracking and pose estimation. In Proceedings of the 17th International Conference on Pattern Recognition, volume 4, pages 264–267, August 2004.
3. Face Image Analysis with Convolutional Neural Networks Stefan Duffner Master of Science Thesis in Electrical Engineering Face Recognition with Preprocessing and Neural Networks David Habrman 2007.
4. Pose-invariant Face Recognition by Matching on Multi-resolution MRFs linked by Supercoupling Transform Shervin Rahimzadeh Arashloo, Josef Kittler and William J. Christmas.
5. A Comprehensive Survey on Pose-Invariant Face Recognition Changxing Ding Dacheng Tao Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology University of Technology,2016.
6. Disentangled Representation Learning GAN for Pose-Invariant Face Recognition Luan Tran, Xi Yin, Xiaoming Liu Department of Computer Science.
7. Face Frontalization Using Appearance Flow based Convolutional Neural Network Zhihong Zhang, Xu Chen, Beizhan, Edwin R. Hancock, *Fellow, IEEE*.
8. Illumination and Pose Invariant Face Recognition:A Technical Review K. R. Singh.
9. Logistic Regression and Convolutional Neural Networks Performance Analysis based on Size of Dataset Kartik Chopra, C. Srimathi VIT University.
10. Pose-Invariant Face Alignment via CNN-Based Dense 3D Model Fitting Amin Jourabloo New York 2017.
11. Supplementary Material for Large Pose 3D Face Reconstruction from a Single Image via Direct Volumetric CNN Regression Aaron S. Jackson.

APPENDIX

Python Code for CNN Architecture

```
from getdataset import GetDataSet
from neuralnetwork import NeuralNetwork
from layers import *
import numpy

class cnn:
    def run(self):
        getdataob = GetDataSet()
        X_train,Y_train,X_test,Y_test = getdataob.createDataSet(9)
        self.n_classes = len(numpy.unique(Y_train))
        # Create CNN Feature Extractor
        nn= NeuralNetwork(
            layers=[
                #CNN Feature Extractor
                Conv(
                    n_filters=4,
                    filter_shape=(3, 3),
                    weight_scale=0.1,
                    weight_decay = 0.01
                ),
                Activation('relu'),
                Pool(
                    pool_shape=(2, 2),
                    mode='max'
                ),
                Conv(
                    n_filters=8,
                    filter_shape=(3, 3),
                    weight_scale=0.1,
                    weight_decay=0.01
                ),
                Activation('relu'),
                Pool(
                    pool_shape=(2, 2),
                    mode='max'
                ),
                Flatten(), # Gives Feature Vectors
                # Classifier
                Linear(
                    n_out= self.n_classes,
                    weight_scale=0.1,
                    weight_decay=0.002
                ),
                LogRegression(),
            ],
            )
        # Initialize(Setup) The Layers of CNN
        nn._setup(X_train,Y_train)
        #Fit the Training Set to CNN to learn the task specific filters for Feature Extraction
        nn.fit(X_train,Y_train,learning_rate=0.01,max_iter=100,batch_size=40)
        print("\nModel Trained\n\n")
        print("\nTesting Prediction : \n")
        Y_pred = nn.predict(X_test)
        print("Actual  : \n",Y_test)
        print("Predicted : \n",Y_pred)
        error = nn._error(Y_pred,Y_test)
        print("Testing Error : ",error)
        file = open("weights.txt", "r+")
        file.write("\nTesting Error : "+str(error))
ob = cnn()
ob.run()
```

Matlab Code for Pose Mapping Using PCA

```
loaded_Image=load_database();
random_Index=round(65*rand(1,1));
random_Image=loaded_Image(:,random_Index);
rest_of_the_images=loaded_Image;%(:,[1:random_Index-1 random_Index+1:end]);
image_Signature=10;
white_Image=uint8(ones(1,size(rest_of_the_images,2)));
mean_value=uint8(mean(rest_of_the_images,2));
mean_Removed=rest_of_the_images-uint8(single(mean_value)*single(white_Image));
L=single(mean_Removed)'*single(mean_Removed);
[V,D]=eig(L);
V=single(mean_Removed)*V;
V=V(:,end:-1:end-(image_Signature-1));
all_image_Signatire=zeros(size(rest_of_the_images,2),image_Signature);
for i=1:size(rest_of_the_images,2);
    all_image_Signatire(i,:)=single(mean_Removed(:,i))'*V;
end
subplot(131);
imshow(reshape(random_Image,112,92));
title('Looking for this Face','FontWeight','bold','Fontsize',10,'color','red');
subplot(132);
p=random_Image-mean_value;
s=single(p)'*V;
z=[];
for i=1:size(rest_of_the_images,2)
    z=[z,norm(all_image_Signatire(i,:)-s,2)];
    if(rem(i,20)==0),imshow(reshape(rest_of_the_images(:,i),112,92)),end;
    drawnow;
end
[a,i]=min(z);
subplot(132);
imshow(reshape(rest_of_the_images(:,i),112,92));
title('Recognition Completed','FontWeight','bold','Fontsize',10,'color','red');
frontal_index=rem(random_Index,13);
if(frontal_index==0)
    frontal_index=13;
end
frontal_Image=loaded_Image(:,frontal_index);
subplot(133);
imshow(reshape(frontal_Image,112,92));
title('Frontal Face','FontWeight','bold','Fontsize',10,'color','red');

```

Matlab Code for Pose Estimation

```
function estPose=PoseEstimationWithoutAlignmentDemo()
%% Initialization
% add the Code folder to the Matlab search path
addpath('Code');
addpath('Code\Mat Data');
addpath('Code\Mat Data\Grid 10x10 RBF9Dsd 45')


Fx='RBF9D'; % the size of the RBF function e.g. RBF5D, RBF9D, RBF17D
NPatch=100; % the number of the patches in the grid.  Keep it fixed at 10x10 grid i.e. NPatch=100
sd=45;      % standard deviation of the RBF functions,  can try 11.25,45 or 90.

%load the library
load('Code\Mat Data\libraryFaceDetector240Preproc2Auto.mat','library');
sdiv=0.5; % standard deviation for mapping to the library.  Keep this fixed at 0.5 to be consistent with training

%% read the image

im1=imread('1.pgm');
im1=imresize(im1,[60,60]);

%% Preprocessing
%preprocess
im2=preprocess2Im(im1);

%divide to patches
rt=divideToPatches(im2,sqrt(NPatch));


%% Pose estimation
%find where maps to the library
libMap=findLogProbMap(rt,library,sdiv);

%Estimate pose for this image
estPose=testRegressionGivenImageManyRatings(libMap,Fx,NPatch,sd);

%% Display the result
%display the image and the estimated pose
figure
imshow(im1,'InitialMagnification',200)
hold on
text(30,0,...
            ['Estimated pose: ' num2str(estPose)],...
            'HorizontalAlignment','center',...
            'BackgroundColor',[.7 .2 .5]);

```