Human Emotion Recognition Using Deep Learning Techniques

A Dissertation submitted towards the partial fulfilment of
the requirement for the award of degree of

Master of Technology
in
Signal Processing & Digital Design

Submitted by
Arun
2K15/SPD/05

Under the supervision of

Sh. Rajesh Birok
(Associate Professor, Department of ECE)

Department of Electronics & Communication Engineering
Delhi Technological University
(Formerly Delhi College of Engineering)
Delhi-110042
2015-2017

DELHI TECHNOLOGICAL UNIVERSITY

Established by Govt. Of Delhi vide Act 6 of 2009

(Formerly Delhi College of Engineering)

SHAHBAD DAULATPUR, BAWANA ROAD, DELHI-110042

CERTIFICATE

This is to certify that the dissertation title "**Human Emotion Recognition Using Deep Learning Technique**" submitted by **Mr. ARUN, Roll. No. 2K15/SPD/05**, in partial fulfilment for the award of degree of Master of Technology in "**Signal Processing and Digital Design (SPDD)",** run by Department of Electronics & Communication Engineering in Delhi Technological University during the year 2015-2017, is a bonafide record of student's own work carried out by him under my supervision and guidance in the academic session 2016-17. To the best of my belief and knowledge the matter embodied in dissertation has not been submitted for the award of any other degree or certificate in this or any other university or institute.

Dr. S. Indu
Head of Department
Electronics and Communication Dept.
Delhi Technological University
Delhi-110042

Sh. RAJESH BIROK
Supervisor
Associate Professor (ECE)
Delhi Technological University
Delhi-110042

DECLARATION

I hereby declare that all the information in this document has been obtained and presented in accordance with academic rules and ethical conduct. This report is my own work to the best of my belief and knowledge. I have fully cited all material by others which I have used in my work. It is being submitted for the degree of Master of Technology in Signal Processing & Digital Design at the Delhi Technological University. To the best of my belief and knowledge it has not been submitted before for any degree or examination in any other university.

Arun

M. Tech. (SPDD)

2K15/SPD/05

ACKNOWLEDGEMENT

I owe my gratitude to all the people who have helped me in this dissertation work and who have made my postgraduate college experience one of the most special periods of my life.

Firstly, I would like to express my deepest gratitude to my supervisor **Sh. Rajesh Birok**, Associate Professor (ECE) for his invaluable support, guidance, motivation and encouragement throughout the period during which this work was carried out.

I also wish to express my heart full thanks to my classmates as well as staff at Department of Electronics & Communication Engineering of Delhi Technological University for their goodwill and support that helped me a lot in successful completion of this project.

Finally, I want to thank my parents, family and friends for always believing in my abilities and showering their invaluable love and support.

ARUN

M. Tech. (SPDD)

2K15/SPD/05

ABSTRACT

Human emotion recognition plays an important role in the interpersonal relationship. The automatic recognition of emotions has been an active research topic from early eras. Therefore, there are several advances made in this field. Emotions are reflected from speech, hand and gestures of the body and through facial expressions. Hence extracting and understanding of emotion has a high importance of the interaction between human and machine communication.

The clinical, emotionless computer or robot is a staple of science fiction, but science fact is starting to change: computers are getting much better at understanding emotions. Automated customer service "bots" will be better able to know if a customer is getting the help they need. Robot caregivers involved with telemedicine may be able to detect pain or depression even if the patient doesn't explicitly talk about it. Insurance companies are even experimenting with call voice analytics that can detect that someone is telling lies to their claims handers.

This project will use deep learning techniques to detect human emotions from faces, since face is the prime source for recognizing human emotions. In particular, we used convolutional neural network(CNN) as the deep learning technique. Network was designed in Python language with the help of deep learning library by Google called TensorFlow without the CUDA framework.

INDEX

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Emotions play a key role in human life. Research on human emotion recognition over past few decades has resulted in several important real-world applications. Human emotion analysis has consistently been an active topic as the Physiognomy has been waxing and waning over the time.

In the 19th century, Charles Darwin's work related to automatic facial expression recognition is the fundamental for today's theories and applications. Darwin also cataloged the facial deformations related to the different class of expressions. For example: *"the contraction of the muscles round the eyes when in grief", "the firm closure of the mouth when in reflection", "the depression of the corners of the mouth when in low spirits".*

Most of the current theory and applications related to emotion recognition are built around the work of Ekman. According to Ekman human emotions can be broadly classified into 6 categories: Anger, Happy, Sad, Surprise, Fear and Disgust. Ekman developed a facial action coding system (FACS) in which he defined the various movements of facial muscles as Action Units (AU). With the help of these action units various class of emotion can be easily classified. For example, in the happy state of any person, intensity of action units related to 'cheeks raising' and 'lip corner movement' will increase.

There are mainly two models to encode the facial muscle movements:

1. Ekman's Facial Action Coding System (FACS)
2. MPEG's Facial Action Potential (FAP)

For images, FACS shows very robust performance but for videos it is a challenging task. Emotional state of a person can be recognized by determining the facial expressions using FACS. For real-time applications, AUs should be recognized from profile views too.

FAPs defines the deformation of face from its neutral state. The value of FAP shows the magnitude of the deformation with respect to neutral face. There is a total of 68 FAPs grouped into 10 categories.

Before the advent of deep learning techniques, these two models were used mostly for emotion recognition. But for real world applications, the performance of the systems built upon these models were not much robust. For example, in real world applications, it is difficult to obtain the frontal view of the face thus the system should be robust enough to classify emotion from profile view also. Another common problem is of occlusion which can limit the performance of the system.
Example images of FACS and FAPs are shown below. A common workflow to detect the emotion with the help of these two models will also be described briefly.

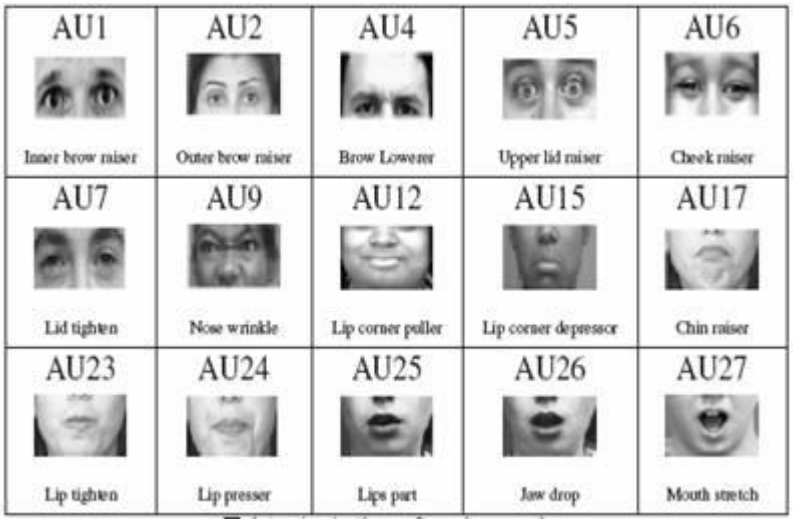


Figure 1.1: Examples of AU from FACS

AUs can be either additive or non-additive. Additive AUs are independent and do not affect the appearance of other AUs while AUs are called non-additive if they modify the appearance of each other's. Facial Action Coding System greatly simplified representation of facial expression. Now with the help of FACS, each facial expression can be represented as a combination of one or more additive or non-additive AUs.

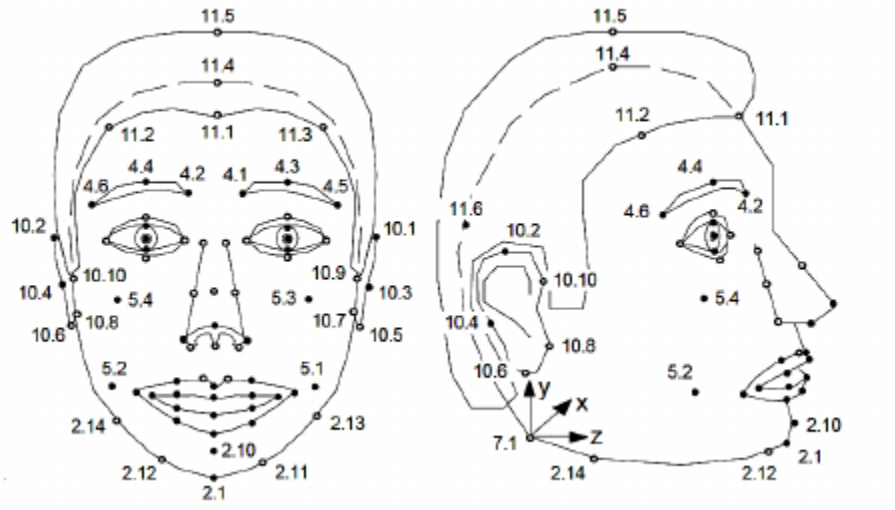


Figure 1.2: MPEG-4 facial feature points

The first step in the emotion recognition is to extract face from the input image. Most commonly used method is the Haar cascade classifier. Now features can be extracted from these face regions with the help of different methods like: Gabor transform, Active appearance models, Hidden Markov models etc. FACS and FAPs also acts like a feature set. There are different methods which can extract Action Units (for FACS) and Facial Action Potentials (For MPEG-4 FAP). At last, machine learning classifiers like support vector machine, multi-layer perceptron, k nearest neighbor can be used to classify the features. Workflow of this process is shown in figure on the next page.

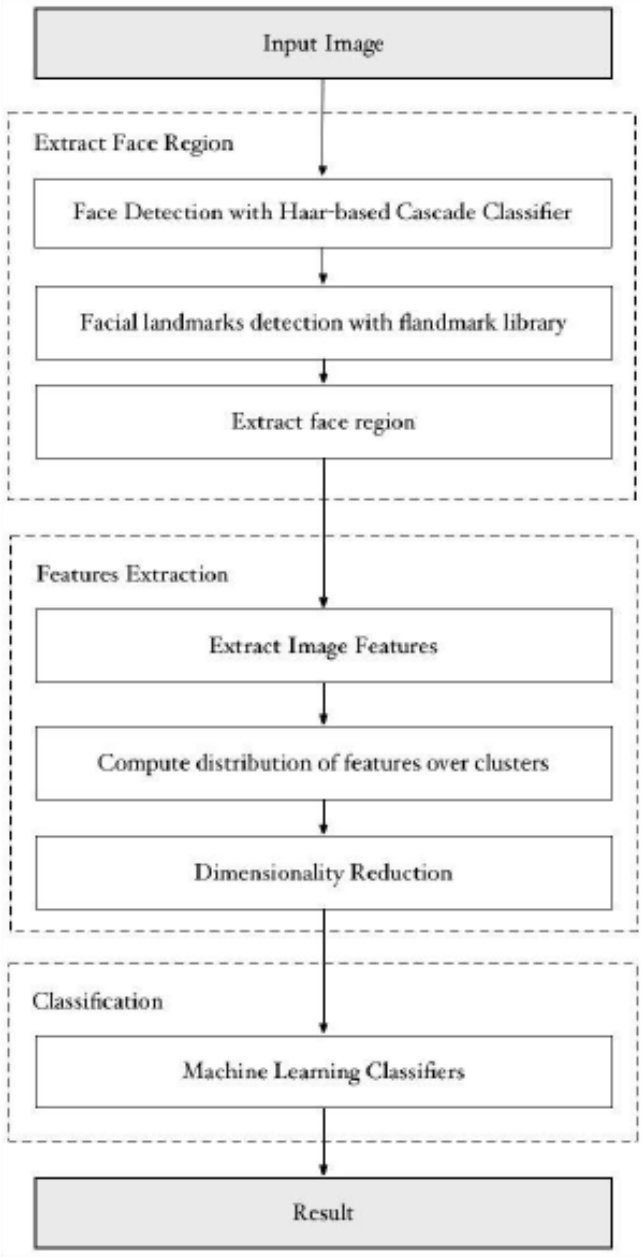


Figure 1.3: Flowchart for emotion recognition system (Using machine learning techniques)

There are some limitations on the performance of the systems based on this approach like:

1. Works well for posed expressions but gives poor results for spontaneous expressions.
2. Occlusions are not handled much efficiently
3. Don't give satisfactory results for real world applications

Deep learning techniques especially the CNN (convolutional neural networks) can counter these issues very efficiently because of the fact that CNNs are very good at the classification of images. Convolutional neural networks can find even the complex patterns in images like the facial expressions. In the recent years, many state of the art networks were presented by researchers for different image classification problems like the famous IMAGENET challenge. A brief introduction to the evolution of deep learning techniques is presented below.

1.1 Deep Learning

Deep learning is a branch of machine learning which deals with the deep neural network architectures. Deep learning is used mainly to solve various problems in computer vision, NLP, and bioinformatics, among other fields. With the advancement in computational resources, deep learning has emerged as an efficient solution to many real-world problems as mentioned above.

Deep learning is the implementation of deep neural networks. Deep neural networks have more than a single hidden layer of neurons. However, this is not a general definition for DNN rather it is a very simplistic view of deep learning. Architecture of DNN varies considerably according to the different tasks and goals. Many state of the art DNN models are implemented in recent years and is used as the core of many giant tech companies like Google, Facebook, Amazon etc.

Below is presented a brief introduction of the evolution of deep learning models from the very basic element i.e. perceptron.

1.1.1 Perceptrons

Perceptron, a basic neural network building block, is the earliest supervised learning algorithm.

Consider the case of binary classification problem, where only two classes exist denoted by 0 and 1. The task is to assign a class to the new input data point, either 0 or 1. In the simplest way, it can be done by calculating the distance of input data point to some specific neighbors and assign the input to the class having majority of nearest distance. A better approach is to first draw a line to separate the data and then assign the class to input data according to the region in which it lies.

In this case, each data point is represented as a vector $x = (x1, x2)$. And the separating function will be like '0' to the area below the line and '1' to area above the line. Mathematically it can be represented by defining the separating function as a vector of weights **w** and a vertical offset (or bias) *b*. Then, the function will take the weighted sum of the inputs and weights:

$$f(x) = w.x + b$$

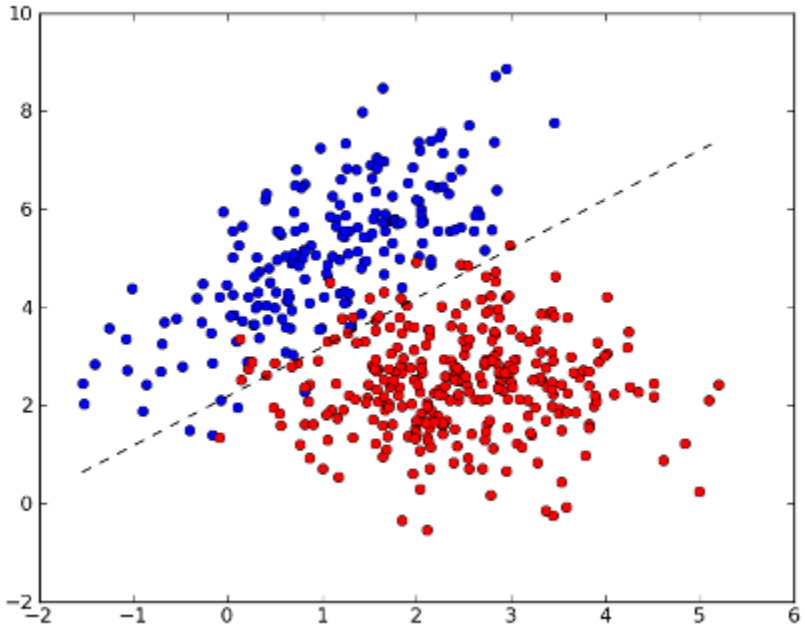


Figure 1.4: Example of a linear classifier

An activation function is used to produce the labeling according to the input. For example, a threshold cutoff activation function (e.g., 1 if greater than some value) can be used as below:

$$h(x) = \begin{cases} 1 : if\ f(x) = w.x + b > 0 \\ 0 : \qquad\qquad otherwise \end{cases}$$

1.1.2 Training of Perceptron

Perceptron can be trained by feeding multiple training examples and calculating the output for each of them. After each sample, error is calculated between the desired output and actual output and the weights *w* are adjusted accordingly to minimize the *output error*. Error functions may differ according to the need of application but training of the perceptron will always be done in the same way as described above.

1.1.3 Drawbacks of single perceptron

Major drawback to use single perceptron for deep learning is, it can separate only linearly separable data. Perceptron can't handle the nonlinearity in the input data, it can't draw complex decision boundaries. This can be seen clearly with an example below, in which perceptron is unable to draw a single classifier to separate even the very simple non-linear XOR problem.

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$Y = X_1 \oplus X_2$

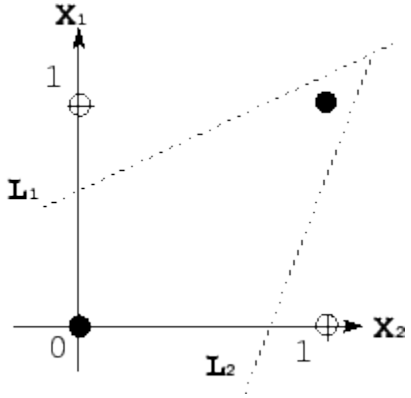


Figure 1.5: Fail attempt of a single perceptron to classify nonlinear problems

This problem can be resolved by using multi-layer perceptron also called feed-forward neural network. MLP is a bunch of perceptrons arranged in some specific manner.

1.1.4 Feedforward Neural Networks for Deep Learning

As mentioned above, MLP is just a collection of perceptrons, connected in some specific way and operating on different activation functions.

Feedforward neural network has the following properties:

- Basic component of any MLP is an input, output, and one or more *hidden* layers. In the example figure below, feed-forward network has a 3-unit input layer, 4-unit hidden layer and an output layer with 2 units (the terms units and neurons can be used interchangeably).
- The neuron of input layer serves as the input for hidden layer while the neurons of the hidden layer serves as the input for output layer.
- Weights are defined for every interconnection of neurons.
- The neurons in each layer *s* is typically connected to *every* neuron of the previous layer *s - 1* (although they can be disconnected by setting their weight to 0).
- The data is processed by clamping the input vector to the input layer and setting the values of desired output to the output layer. Values of input layer are then propagated forward to the hidden units after the weighted sum (hence the term forward propagation), which in turn produces the output with the help of activation function.

- The output of the network is calculated at hidden layer in the same way as hidden layer.

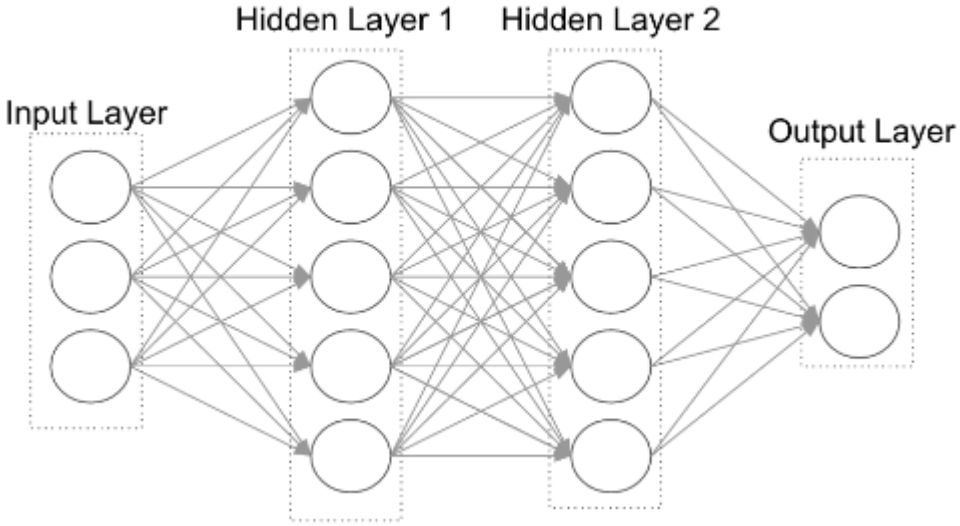


Figure 1.6: Feed forward neural network (Multi-layer perceptron)

1.1.5 Problems with Linearity

If each of our perceptrons is allowed to use only linear activation function, then, the final output of the network will *still* be some linear function of the inputs. Thus, if neurons are restricted to use linear activation function, then the feed forward network will be of no significant use, no matter how many number of layer it consists.
This is the reason why non-linear activation functions are used for most of the feed forward networks. Most commonly used non-linear activation functions are:

- logistic
- tanh
- binary or rectifier

1.1.6 Hidden Layer

The hidden layer is of great importance in any neural network. According to the universal approximation theorem, a neural network with a single hidden layer having finite number of neurons can approximate an arbitrarily random function. In other words, a single hidden layer is powerful enough to learn *any* function. Thus, multiple hidden layers (i.e., deeper nets) performs much better than the perceptron. Hidden layers store the internal abstract representation of the training data, same as a human brain (greatly simplified analogy) has an internal representation of the real world.

1.1.7 The Problem with Large Networks

In general, the performance of a neural network increases by increasing the number of hidden layers. Higher layers build abstraction over previous layers thus generating more complex features. These complex features help the neural network to efficiently classify the input data.

However, the number of hidden layers can't be increased arbitrarily. Increasing the number of hidden layers causes two issues:

1. **Vanishing gradients:** as the more hidden layers are added, backpropagation becomes less efficient in passing information to the lower layers. Error at the output, according to which the weights have to be adjusted gets much diluted farther from the output. In other words, as information (error in case of neural network) is passed back, the gradients begin to vanish and become small relative to the weights of the networks.

2. **Overfitting:** this is the central problem in machine learning. Overfitting tries to fit the training data as efficiently as possible. This reduces the training error greatly but it gives very poor results over test data. In other words, overfitting can give very complex boundary to fit the training data very well but it can't be generalized for unseen data.

3. **Complex Optimization**: More number of layers intensify the problem as it results in the increased number of parameters. As the number of hidden layers increases, there will be more paths thus the optimization is complex.

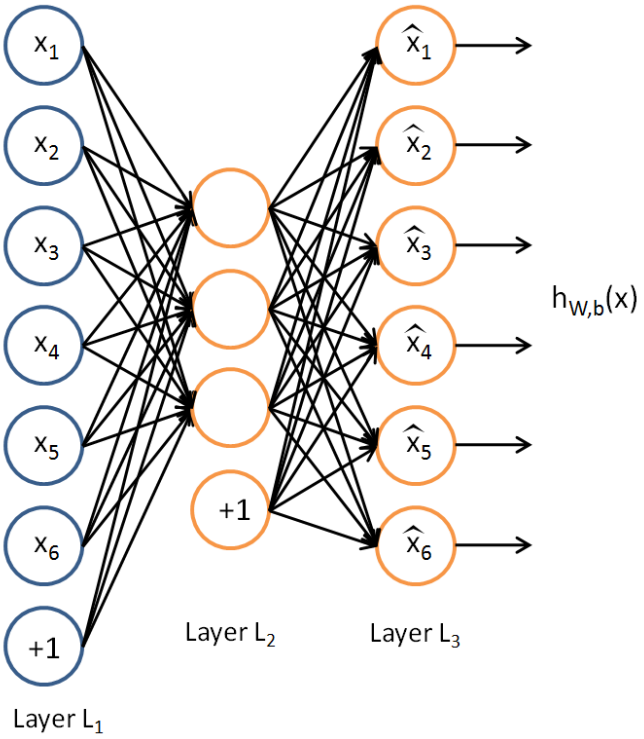
To address these issues, below are introduced some deep learning algorithms.

1.1.8 Autoencoders

An autoencoder is a neural network that takes typically unlabeled data as input and after encoding them, tries to reconstruct them as accurately as possible. As a result of this the net must decide which of the data features are the most important essentially acting as a feature extractor.

Autoencoder aims to *learn a compressed, distributed representation (encoding) of a dataset.* Conceptually, here both the input and output data remains same, the network is trained to "recreate" the input. In other words, the output is going to be same as

the input but compressed in some sense. This is a confusing approach, example below will explain the concept.



.

Figure 1.7: Autoencoder

Concept of autoencoders

Consider the training data having input grayscale images of size 28x28. Each neuron at the input layer takes the value of pixels in the image (i.e., the input layer will have 784 neurons). Then for autoencoders, the output layer will also have 784 neurons and the target value for these neurons will be the pixel value of the images.

The main objective of autoencoders is to learn the internal structure and features of the data. Autoencoders will not learn a "mapping" between the training data and its labels. This is the reason why hidden layer is also called *feature detector*. Usually, the number of neurons in hidden layer is smaller than that in the input/output layers. This ensures that the autoencoders learn only the most important features and gives

the output data that is compressed. Dimensionality reduction is a functionality of the autoencoders.

Autoencoders are trained with backpropagation algorithm using a metric called loss. Thus, autoencoders gives a compact representation of input data.

1.1.9 Restricted Boltzmann Machines

RBM is a shallow two-layer net. It is mathematical equivalent of two-way translator. In the forward pass the RBM takes an input and translates them into a number that encodes the input, in the backward pass it takes the set of number and translates them back to reconstruct the inputs.

RBMs are composed of three layers:

1. hidden layer
2. visible layer
3. bias layer

In RBM, the connection between the hidden layer and visible layer is undirected means the values can be propagated in both the direction i.e. from visible to hidden and from hidden to visible. The connections are fully connected also means each neuron in any layer is connected to each neuron in next layer. If the neurons in any layer is allowed to connect with neurons in any other layer then the resulting network will be Boltzmann machine rather than a restricted Boltzmann machine

The standard RBM consists of binary hidden and visible units: that is, the unit activation is 0 or 1 under a Bernoulli distribution, but there are variants with other non-linearities. RBM uses KL divergence to compare the actual to recreation.

RBM is a part of feature extractor neural network. They automatically find patterns in a data by reconstructing the input. These nets are also called autoencoders in some sense because in a way they encode their own structure. Restricted Boltzmann machines are a special case of Boltzmann machines and Markov random fields. Their graphical model corresponds to that of factor analysis

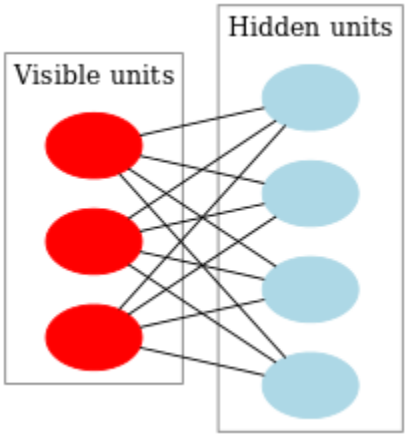


Figure 1.8: Restricted Boltzmann Machines

1.1.10 Deep Networks

As mentioned above, Autoencoders and RBMs can be used as a feature extractor. But these features can't be used directly for further processing. Thus, we need a method by which these extracted features can be used indirectly.

Here Deep networks comes to the rescue. Deep networks can be formed by stacking these structures. Most attractive feature of deep networks is: the layers of these

networks can be trained greedily, one by one. They also overcome the problems faced by classic backpropagation like vanishing gradients and overfitting.

In terms of network architecture, a deep network is identical to MLP but when it comes to training, they are entirely different. In fact, the difference in training is the key factor that enables deep network to outperform their shallow counterpart. Key advantages of deep networks is:

- Less training time
- Increased accuracy
- Small labeled dataset

1.1.10.1 Stacked Autoencoders

This network is a stack of multiple encoders.

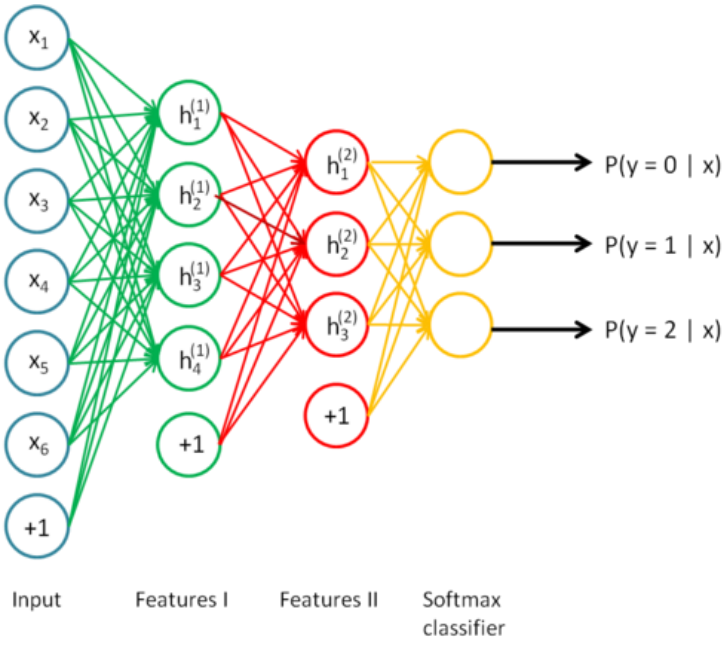


Figure 1.9: Stacked Autoencoders

Input is applied to the first layer of the first autoencoder. This input works as the input for whole system. Input to any hidden layer is the output of the previous layer. The procedure for layer-wise greedy training is as below:

1. With the help of backpropagation algorithm, the first autoencoder (*t = 1* or the red connections) is trained individually with an additional output layer by taking all the available training data.

2. The second autoencoder *t=2* (green connections) is trained by clamping input sample to the input layer of *t=1,* which is propagated forward to the output layer *t=2*. Since the hidden layer *t=1* works as the input layer for *t=2*, the output layer of *t=1* is no longer required and can be removed from the network.

3. The above procedure is repeated for all the layers (i.e., replace the output layer of previous encoder with another encoder, and train with back propagation).

4. Steps 1-3 initializes the weights properly and called as *pre-training*. However, the input data is not associated in any way to the output labels. For example, in case of a handwritten digit recognition system, we can't get the digit type of the input image from the hidden layer of the last autoencoder. In that case, one or more fully connected layers can be added as a solution to get the final mapping between the input image and output label. This network now can be viewed as a MLP and can be trained with the help of backpropagation algorithm (this step is also called *fine-tuning*).

Stacked auto encoders provides an effective pre-training method to initialize the weights of a network. The result is a complex, multi-layer perceptron can be trained (or *fine-tune*).

1.1.10.2 Deep Belief Networks

DBN can be viewed as a stack of Restricted Boltzmann machines (RBM).

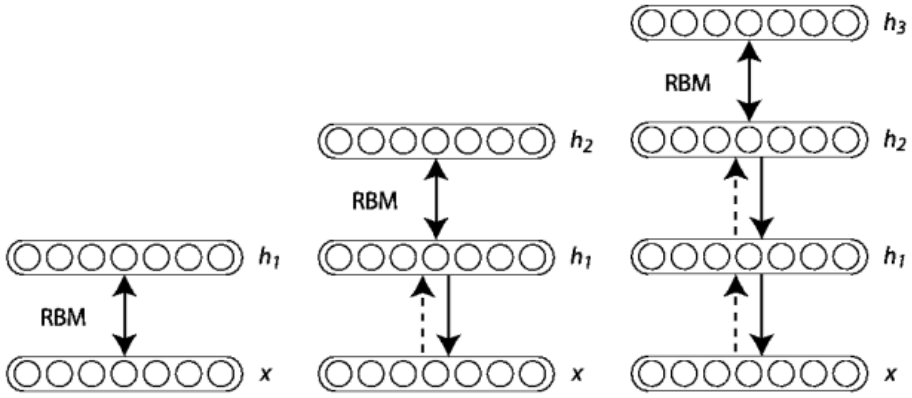


Figure 1.10: Deep Belief Networks

Each RBM layer learns the entire input. Input to the first layer of first RBM works as the input for whole system. Hidden layer of RBM *t* works as a visible layer for RBM *t+1*. The procedure for layer-wise greedy training is as below:

1. First RBM *t=1* is trained with all the training examples using contrastive divergence.

2. Second RBM *t=2* is trained by making the visible layer of *t=1* as input, this data is propagated forward to the hidden layer of *t=1*. Now, with the help of this data, contrastive divergence can be initiated for the training of the RBM *t=2*.

3. Previous procedure is repeated for all the layers.

4. Same as in case of the stacked autoencoders, the network can be extended after pre-training. This can be done by connecting fully connected layers to the final hidden layer of RBM. Again, this forms a MLP which can then be *fine-tuned* with the help of backpropagation.

This procedure is same as to that of stacked autoencoders, but here autoencoders are replaced with RBMs and contrastive divergence algorithm is used instead of backpropagation algorithm.

1.1.10.3 Convolutional Networks

Convolutional neural network is a type of feedforward network with some complex structures as layers in between the structure. Convolutional neural networks are of special interest for performing tasks related to images. CNNs are very good at finding complex patterns in images.

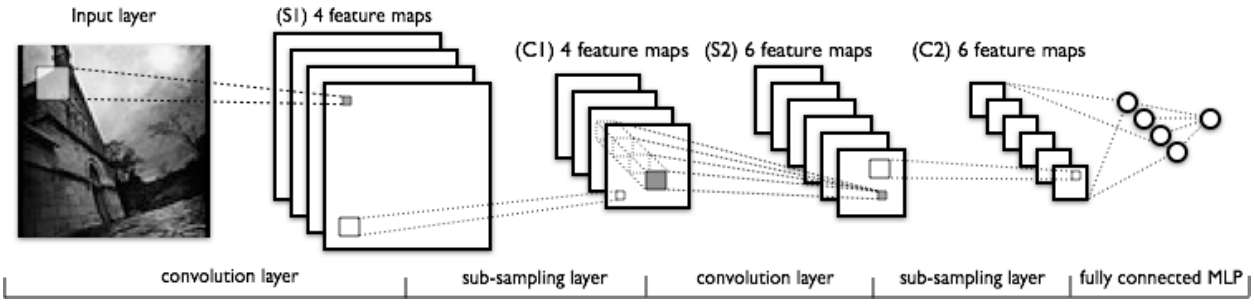


Figure 1.11: CNN architecture

- **Convolutional layers** consist of a number of filters. These filters are applied to input image the result of which is called *feature map* (FM). Each filter generates a feature map. Thus, total number of FM will be equal to the number of filters in convolutional layer. There may be multiple convolutional layers in a CNN. In that case, filters of the second convolutional layer are applied to the feature maps

of the previous layer with different weights. This way each feature map in the input is connected to the feature map in the output. The concept of weight sharing allows to detect features in the input image regardless of the location. Multiple number of filters helps to extract different types of patterns.

- In convolutional neural networks, there are many number of layers. Thus, the processing time increases exponentiality as the size of the input data increases. Size of the input data or image can be reduced with the help of ***subsampling layers***. For example, if the input image is of size 32x32 and the subsampling layer consists the region of 2x2 then the output will be of size 16x16. Here subsampling will replace the square of 4 pixels with 1 pixel. There are many different methods for subsampling. Commonly used methods are:
    - Max pooling
    - Average Pooling
    - Stochastic Pooling

Most popular is Max pooling as it extracts the strongest features and also prevents the diffusion

of information, as in the case of average pooling.

- After the convolutional and subsampling layers, ***fully connected layers*** are there to represent the labeled output. There may be multiple fully connected layers.
- CNN uses backpropagation algorithm for training. A very efficient technique is used during the training of the CNNs which reduces the training time considerably, is called ***dropout.*** It makes other features unreliable to break co-adaption. Several dropped out architectures are trained in a single run to choose the one with optimal performance.

1.2 Applications of Human Emotion Recognition

Emotion recognition is a trending topic among researchers. As emotions play an important part in our lives, detecting emotions and taking actions accordingly can improve the performance of many field of application. Some most common application of emotion recognition system can be:

1. Healthcare and medicine

Emotion recognition can play an important role in medical treatments. For example, physiatrists can use emotion recognition system to find how the patient is feeling about the treatment and can use different techniques accordingly. Another example where emotion recognition can play a significant role in healthcare is to deal with persons suffering from autism. As these people struggles in social communication, emotion recognition system can help significantly.

2. E - learning

In today's digital world, internet plays a key role in learning new skills. Online courses bridge the gap between users and distant universities. Now according to the state of the user, the presentation style can be changed accordingly to make online tutor more interactive and effective.

3. Monitoring and alerting systems

Driver monitoring system can be deployed in automobiles to warn drivers if they are feeling angry or sad, to pull over for some time and either clam down or take some rest.

Another example of monitoring system is in ATMs, where ATMs can be blocked to dispense money if user is feeling scared (in case of forceful action).

4. Entertainment

Emotion recognition system can play a significant role in entertainment. For example, YouTube recommender system will be more effective if it can also recommend multimedia content according to the emotional state of the user.

5. Marketing

In supermarkets, by detecting the response of the user for any particular product will help the consumer product company to change their products accordingly. In consumer advertisements also, detecting the emotional response will help to make them more effective.

1.3 Thesis Outline

The thesis work is totally divided into five chapters. Outline of each chapter is presented below.

Chapter 1 is the introduction part of the work. This highlights the methods that can be used to model the framework to detect the human emotion. FACS and FAP are discussed in the chapter which decodes the human faces and can be used to detect emotions. Different deep learning techniques are also introduced in the chapter.

Chapter 2 concludes some of the most popular and state of the art work related to emotion recognition. Literature review is summarized in 19 research papers.

Chapter 3 elaborates our proposed method. Convolutional neural networks are discussed in detail. Various techniques to improve the performance like pooling and

dropout are also discussed. In the end, some well-known convolutional networks are referred like LeNet, AlexNet.

Chapter 4 shows the result of our proposed method. All the five evaluation metrics are also discussed *i.e.* confusion matrix, precision, recall, F1-score and accuracy. Results of our deep learning approach are compared also with classical machine learning algorithms.

Chapter 5 concludes our work. It will be clear from the result section that deep learning approach that is used here *i.e.* CNN outperforms the classical machine learning algorithms like MLP, SVM and kNN. This chapter also highlights for the scope of future work for improvement.

Chapter 2

Literature Review

The foundational studies on facial expressions that have formed the basis of today's research can be traced back to the 17th century. In the 19th century, Charles Darwin's work related to automatic facial expression recognition is the fundamental for today's theories and applications. In 1872, Darwin wrote a treatise that established the general principles of expression and the means of expressions in both humans and animals. Since the 1970s, psychologist Paul Ekman and his colleagues has done magnificent work in the study of facial expressions and human emotions. Since it is almost impossible to mention all of the research work related to emotion recognition, here are presented 17 most important and state of the art work related to this field.

1. Bourel et al., 2001 [1]

Feature Extraction

Local spatio-temporal vectors obtained from the Extended Kanade-Lucas-Tomasi tracker

Classifier

Data fusion with modular classifiers. Local classifiers were ranked according to kNN method.

Database

CK

Sample size

There was a total of 100 video sequences from 30 subjects (25 sequences for 4 expressions)

Performance

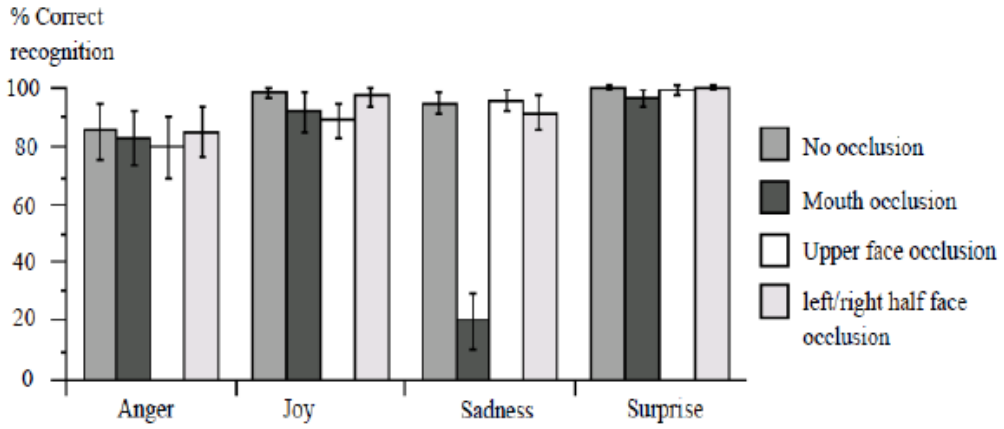


Figure 2.1: Bourel et al's system performance

Important Points:

Fusion methods were used to get the final output from classifier. Effects of occlusion was also considered.

2. Pardas and Bonafonte, 2002 [2]

Feature Extraction

Active Contour algorithm and motion estimation were used to extract MPEG-4 FAPs

Classifier

Hidden Markov Model

Database

CK

Sample Size

Whole database was used

Performance

Accuracy of 84% overall (with 6 prototypic expressions)

Important Points

A method was proposed to automatically extract MPEG-4 FAPs. They also proved that FAPs can also be used efficiently to extract the emotions.

3. Cohen et al., 2003 [3]

Feature Extraction

Motion Units (MUs) vector tracking using piecewise Bezier volume deformation tracker

Classifier

Hidden Markov Models, Multi-level HMM, Naive Bayes, Tree Augmented Naive Bayes

Database

CK and self-made database

Sample Size

53 subjects from CK database

5 subjects from own database

Performance

	Cohn-Kanade DB	Own DB
Person Dependent Tests	Insufficient data to perform person dependent tests	NB (Gaussian): 79.36% NB (Cauchy): 80.05% TAN: 83.31% HMM: 78.49% ML-HMM: 82.46%
Person Independent Tests	NB (Gaussian): 67.03% NB (Cauchy): 68.14% TAN: 73.22% Insufficient data to conduct tests using HMM and ML-HMM	NB (Gaussian): 60.23% NB (Cauchy): 64.77% TAN: 66.53% HMM: 55.71% ML-HMM: 58.63%

Table 2.1: Cohen et al.'s system performance

Important Points

It was real-time system able to classify emotions from video. Use of Hidden Markov Models were recommended for automatic segmentation of a video into different expression segments.

4. Bartlett et al., 2003 [4]

Feature Extraction

Features were extracted using Gabor wavelet transform.

Classifier

Support Vector Machine with and without Adaptive boosting

Database

CK

Sample Size

313 sequences from 90 subjects. First and last frame were used as training images

Performance

SVM (Linear kernel)	Automatic face detection	84.4%
	Manual alignment	85.3%
SVM (RBF kernel)	Automatic face detection	87.5%
	Manual alignment	87.6%

Table 2.2: Bartlett et al.'s system performance

Important Points

It was a real-time and fully automatic system with high level of accuracy.

It was successfully deployed on Sony's Aibo pet robot, ATR's RoboVie and CU Animator

5. Michel and Kaliouby, 2003 [5]

Feature Extraction

Feature displacement vector was used. It is the Euclidean distance between neutral and peak emotion

Classifier

Support Vector Machine

Database

CK

Sample Size

10 examples for training and 15 examples were used for testing, for each class emotion

Performance

With RBF Kernel: 87.9%.

Person independent: 71.8%

Person dependent (train and test data supplied by expert): 87.5%

Important Points

It was a real-time system and did not require preprocessing

6. Pantic and Rothkrantz, 2004 [6]

Feature Extraction

Profile and frontal face points were used as features.

Classifier

Rule based classifier

Database

MMI

Sample Size

25 subjects

Performance

Accuracy of 86% is achieved

Important Points

This was not a real-time system but it can recognize facial expressions in both the profile and frontal views. A method was proposed for automatic action unit coding in profile images

7. Buciu and Pitas, 2004 [7]

Feature Extraction

Non-Negative Matrix Factorization (NMF) and Local Non-Negative Matrix factorization (LNMF) were used to represent the images

Classifier

Nearest neighbor classifier was used with Cosine Similarity Measure and Maximum Correlation Classifier

Database

CK and JAFFE

Sample Size

164 samples from CK database

150 samples from JAFFE database

Performance

CK: the highest accuracy of 81.4% is achieved by LNMF with MCC

JAFFE: 55% to 68% (using all the methods)

Important Points

PCA was also used to compare with the LNMF and NMF. NMF showed worst results while LNMF outperformed both the PCA and NMF.

Cosine similarity measure gives better performance than maximum correlation classifier.

8. Pantic and Patras, 2005 [8]

Feature Extraction

20 facial fiducial points were tracked as feature

Classifier

Temporal Rules

Database

CK and MMI

Sample Size

90 images for CK database

45 images for MMI database

Performance

90% accuracy achieved as overall average

Important Points

Showed robust performance under occlusion. It was invariant to facial occlusions like glasses and hairs

9. Zheng et al., 2006 [9]

Feature Extraction

A Labeled Graph (LG) is created using 34 landmark points with the help of Gabor transform. A semantic expression vector is also built for each training face. The correlation between LG vector and semantic vector is were found using KCCA.

Classifier

Correlation used to estimate semantic expression vector is used for classification.

Database

JAFFE and Ekman's PA

Sample Size

JAFFE contains183 images

Ekman's PA contains 96 images

Neutral expressions were excluded from both the databases

Performance

Semantic Info	JAFFE database	LOIO	85.79%
		LOSO	74.32%
	Ekman's database		81.25%
class label info	JAFFE database	LOIO	98.36%
		LOSO	77.05%
	Ekman's database		78.13%

Table 2.3: Zheng et al.'s system performance

Important Points

Kernel Canonical Correlation Analysis method was used to detect facial expressions and to tackle singularity problem in Gram matrix

10. Anderson and McOwen, 2006 [10]

Feature Extraction

Spatial ratio template tracker based motion signature and MCGM based optical flow features of face

Classifier

Support Vector Machine and Multi-Layer Perceptron

Database

Carnegie Mellon University, Pittsburg Action Unit coded database and a non-expressive database

Sample Size

CMU: 253 samples of 6 basic expressions

Non-expressive: 4800 frames of 10 subjects

Performance

Motion averaging using: co-articulation regions: 63.64%, 7x7 blocks: 77.92%, ratio template algorithms, with MLP: 81.82%, with SVM: 80.52%

Important Points

It was a fully automated real-time system. Gave robust performance for cluttered scene also. Motion-averaging is also used to condense data

11. Aleksic and Katsaggelos, 2006 [11]

Feature Extraction

MPEG-4 facial action potentials, eyebrow and outer-lip. PCA was also used for dimensionality reduction

Classifier

Hidden Markov Models and Multi Stream-Hidden Markov Models

Database

CK

Sample Size

284 recordings of 90 subjects

Performance

Using HMM: eye-brow FAPs only: 58.8%, outer lip FAPs only: 87.32%, Joint FAPs: 88.73%

MS-HMM: 93.66% (weights of outer lip are greater than eyebrows).

Important Points

Showed that MS-HMM can improve the performance. They also suggested a method to assign stream weights.

12. Pantic and Patras, 2006 [12]

Feature Extraction

15 facial points were tracked with particle filter to generate mid-level parameter

Classifier

Rule based classifier

Database

MMI

Sample Size

1500 samples of both static and profile views

Performance

96 test profiles: 86.6%

Important Points

Facial expressions were automatically segmented in input video

Temporal segments were also recognized for 27 AUs

Action units were recognized automatically from images

13. Sebe et al., 2007 [13]

Feature Extraction

Piecewise Bezier volume deformation tracker based MUs

Classifier

Bayesian nets, Support Vector Machine and Decision Trees. Results were improved with the help of voting methods like bagging and boosting.

Database

CK. Created their own dataset also containing spontaneous emotions

Sample Size

Created DB: 28 subjects showing mostly neutral, joy, surprise and delight.

CK: 53 subjects

Performance

With different classifiers: CK: 72.46% to 93.06%, Created DB: 86.77% to 95.57%.

Using kNN with k = 3, best result of 93.57%

Important Points

Most attractive feature was to detect the spontaneous emotions.

14. Kotsia and Pitas, 2007 [14]

Feature Extraction

Candide method with Geometric displacement

Classifier

Multiclass SVM

Database

CK

Sample Size

Complete database is used for training

Performance

Facial expression recognition: 99.7%

Action unit detection based Facial expression recognition: 95.1%

Important Points

Detects either the six prototypic expressions or a set of chosen action units.

Recognition rate was very high

15. Wang and Yin, 2007 [15]

Feature Extraction

Topographic context (TC) expression descriptors

Classifier

Quadratic Discriminant Classifier, Linear Discriminant Analysis, Support Vector Classifier and Naïve Bayes.

Database

CK and MMI

Sample Size

CK: 864 images (4 images per subject for each expression for 53 subjects)

MMI: 180 images (6 images per subject for each expression for 5 subjects)

Performance

Person dependent test	MMI database	QDC	92.78 %
		LDA	93.33%
		NB	85.56%
	CK database	QDC	82.52%
		LDA	87.27%
		NB	93.29%
Person independent test	CK database	QDC	81.96%
		LDA	82.68%
		NB	76.12%
		SVC	77.68%

Table 2.4: Wang and Yin's system performance

Important Points

Proposed a topographic model-ing approach in which the gray scale image is treated as a 3D surface.

Analyzed the robustness against the distortion of detected face region and the different intensities of facial expressions.

16. Dornaika and Davoine, 2008 [16]

Feature Extraction

Features were tracked with the help of candid face model.

Classifier

Online Appearance Models for head pose and then stochastic approaches for emotions

Database

Used their own dataset

Sample Size

Video sequences were used instead of images.

Performance

Graphs are mentioned in the reference

Important Points

This was one of the first framework to simultaneous face tracking and emotion recognition.

Posed expression were also there in the videos.

17. Kotsia et al., 2008 [17]

Feature Extraction

Feature were extracted with the help of these three approaches: Discriminant Non-Negative Matrix factorization, Gabor transform, and Geometric displacement vectors.

Classifier

Multiclass Support Vector Machine and Multi-Layer Perceptron

Database

CK and JAFFE

Sample Size

Complete database was used in training

Performance

For JAFFE dataset:

Using Gabor transform, the accuracy was 88.1% while with DNMF it was 85.2%

For CK dataset:

Using Gabor transform the accuracy was 91.6%, with DNMF 86.7% and with SVM: 91.4%

Important Points

Effect of occlusion on prototypic expressions was discussed and the system was robust as it identified the emotions in spite of the occlusions.

Chapter 3

Proposed Methodology

In chapter 1, various deep learning techniques were discussed like autoencoders, restricted Boltzmann machines, deep belief networks and convolutional neural networks. We will use convolutional neural networks for the human emotion recognition problem as CNNs are very efficient at recognizing patterns from images.

3.1 Basic concept of CNN

Before the advancement in computational resources deep learning was not much popular among researchers. Till then, multi-layer perceptron (MLP) was the backbone for automatic pattern recognition. But for more complex patterns MLP have some limitations, like:

- More layers intensify the problem for complex patterns
- Error gets much diluted farther from the output
- Many different paths are available in MLP thus optimization is complex
- Fine control over architecture is needed

Now, a simplification over this architecture can be applied if the weights are shared instead of making them independent. The structure of weights to a particular neuron is exactly the same, their values are also exactly the same. In fact, these are not separate neurons, but the same one applied at different locations. This structure works as a filter which filters out a specific pattern from image. Applying many of these filters over the image will extract the specified patterns. This process is called convolution. In general, for convolution repeat the process below:

- Slide a weight matrix over a feature map

- Take element wise product
- Add the products
- Resulting value is put in one location in the next feature map

Convolution reduces weights. For example, in MLP the connected layer from a 6x6 layer to another 6x6 layer will require 6^4 fully connected weights. With a convolutional architecture, we need only 3x3 weights. Each neuron (kernel) processes patches, not entire image. Convolutional kernel is a matched filter. It looks for a pixel pattern that matches its own structure. The pattern can appear in any place.

Thus, Convolutional networks also known as convolutional neural networks or CNNs, are a specialized kind of neural network for processing data that has a known, grid-like topology.

3.2 CNN architecture

CNN is a modification of the basic multilayer perceptron. It has mainly three types of layers: convolutional layers, pooling layers and fully connected layers. Below is an example figure for architecture of CNN.

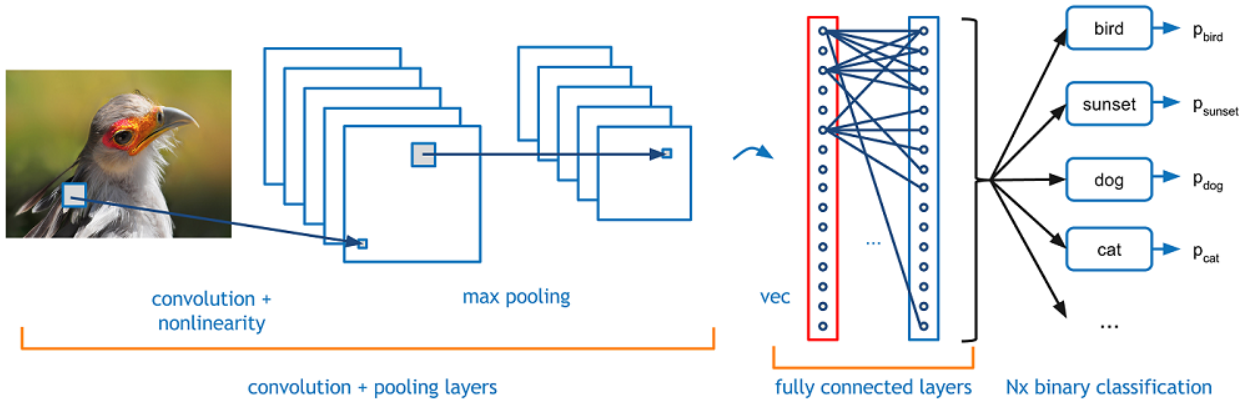


Figure 3.1: Architecture of CNN

3.2.1 Convolutional layers

These are the first layers in any convolutional network. There may be multiple number of convolutional layers. Input to these layers are the images itself. The pixel values of images are feed directly to the convolutional layers. Functionality of these layers can be viewed as a flashlight over an image searching for specific pattern. Here filters are used as a flashlight. These filters slides over the whole image in search of a specific pattern. Different types of filters can be applied in the convolutional layer to find different patterns. The area of projection of filters over the image is known as receptive field. Commonly used filter size is 3x3 and 5x5. Filters produces feature map by sum of multiplication of their values with images. These values are calculated over the whole image. This process is known as convolution. This is the reason why these layers are called convolutional layers.

3.2.2 Pooling layers

Pooling is used to reduce the size of the feature map. As the CNNs consist of a lot of layers to process data, the computation time of the process will also be reduced then. Thus, it is not a good choice to process all the features generated by the convolutional layers. We need only the most relevant features, for this purpose pooling is used in most of the architectures to reduce the size of the feature map. Most commonly used pooling method is max pooling.

General pooling. Besides the max pooling other pooling methods are also there such as average pooling and L-2 norm pooling. Average pooling is important from historical perspective only. Max pooling is the most commonly used method. Average pooling has the main drawback that it dilutes the information while max pooling extracts the strongest features.

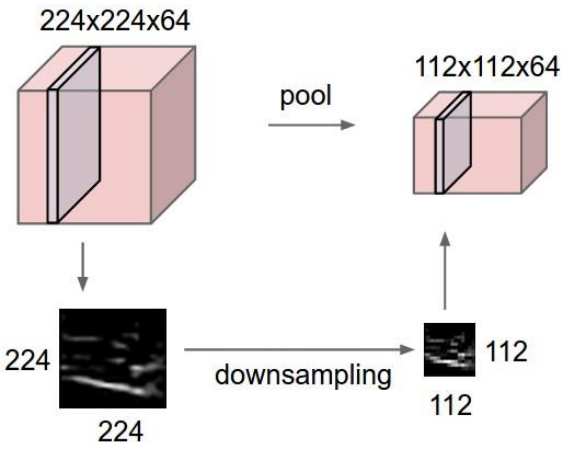


Figure 3.2: Pooling operation

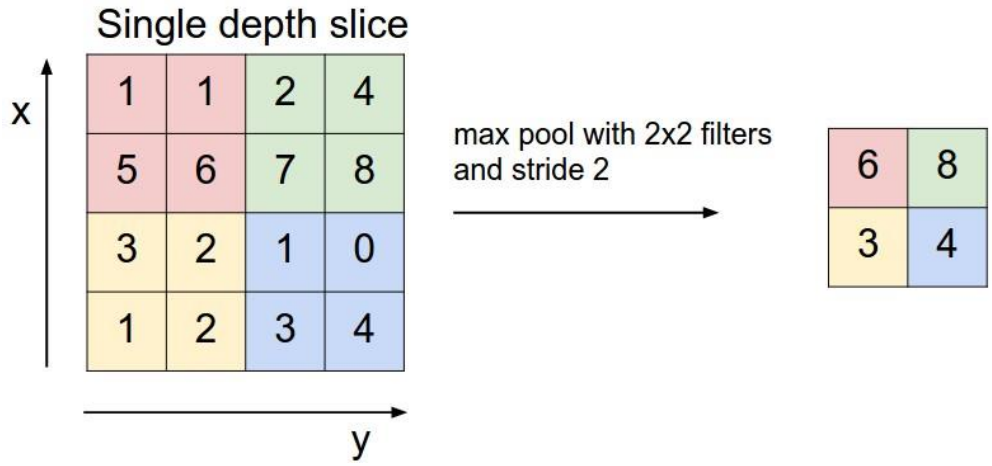


Figure 3.3: Example of Max pooling

Backpropagation. During the backward pass of the training, in max(x,y) step backpropagation routes the gradient with the largest value in the forward pass to the input. Hence, the index of the maximum gradient must be tracked in the forward pass of pooling layer to efficiently route the gradient in backpropagation.

Getting rid of pooling. In the recent studies, it is suggested that pooling can be discarded. In the CNN architecture, convolutional layers can be repeated instead of pooling layers. Now to reduce the size of the input volume, strides of larger size can

be used in convolutional layers once in a while. For the training of generative models like variationally encoders and generative adversarial networks, discarding pooling layers has been found beneficial.

3.2.3 Fully Connected layer

Fully connected layers are similar to basic neural networks as each neuron in fully connected layer is connected with each neuron in the previous layer. Activation of these layers can be easily computed with a simple matrix multiplication after adding the offset component.

Convolutional and fully connected layers are almost similar, the main difference between these two layers is the connectivity of neurons. In fully connected layers, each neuron is connected to each other in the previous layer while in the convolutional layers, neurons are connected to only a small part of the input and that much neurons only share weights. Both layer uses dot product to compute the output. Thus, convolutional layers can be converted into fully connected layers.

- For convolutional layer in the network, same forward function is implemented with a fully connected layer. Most of the entries in the weight matrix will be zero in a block as neurons in the convolutional layers are connected only to a small part of the input. Many of these blocks will be same as convolutional layers uses the concept of weight sharing.

- Fully connected layer can also be converted into a convolutional layer. For example, a fully connected layer having the value of the parameter K as 4096, with 7x7x512 size of input volume can be converted into a convolutional layer having the value of the parameters F, P, S and K as 7, 0, 1 and 4096

respectively. Thus, we need to set the size of the filter same as the input volume size. In this case, output will be of size 1×1×4096 because only a single depth column has to fit across the input volume. This will give the same results as the initial fully connected layer.

3.3 Layer Sizing Pattern

In any convolutional network, there are a number of hyperparameters, the value of which must be taken carefully. Value of these can't be selected arbitrarily. There are certain rule of thumb for sizing the architecture. These are mentioned below:

The **input layer** (containing the image) needs to be divisible by 2 multiple times. Common numbers are: 32, 64, 96 or 224, 384, and 512.

The **convolutional layers** should use filters of small size (e.g. 3x3 or at most 5x5). If the value of stride S is 1 then zero padding should be used in such a way so that the spatial structure of the input volume is not altered. Thus, for the value of F as 3 and for padding P is 1, we will get the original size of the input. For the value of F as 5, the value of P should be 2. In general, for any value of F, $P = (F - 1)/2$ the original size of the input will be retained. If the bigger sizes of the filters can't be avoided then they should be restricted to the very first convolutional layer only i.e. looking on the input image.

The **pooling layers** are mainly responsible for down sampling the spatial size of the input. The most commonly used value for max-pooling is F =2 (2x2 receptive fields), and for stride S the value is 2. Exactly 75% of the input volume is discarded by this setting as we down sampled the space by 2 along both the height and the width. Another slightly less commonly used value for F is 3 (receptive fields of size 3x3)

with the value of stride is 2. Receptive fields of size greater than 3 is not used because the pooling becomes too lossy and aggressive then.

Reducing sizing headaches, in the above-mentioned scheme the size of the input volume is not changed as we used zero padding for the input data. In many different approaches if the zero padding is not used and the value of the stride also is not 1, then the size of the input volume is changed and they should be tracked throughout the CNN architecture.

Significance of stride with value 1: Smaller strides give better results. Having the value of stride 1 only POOL layers are responsible for down sampling, where in convolutional layers, the input volume is only transformed depth-wise.

Necessity of padding: Padding actually improves the performance. If the zero-padding is not allowed to perform and only the convolution operation is allowed to perform over input volume then the size of the input volume will be reduced by some amount after each convolution, and the information at borders will be disappeared rapidly.

Memory constraints based compromise*:* With the rule of thumb presented above, sometimes memory can build up very quickly. Smaller filter sizes and strides results in more number of activations thus will require a large amount of memory as compared to the case where the filter size and the stride have larger values. As the performance of the GPUs are totally dependent on the system memory. Thus, if there is not enough memory available in the system the performance of the GPUs must be compromised. In general, compromise is made only at the first convolutional layer.

3.4 Proposed CNN architecture

We used 2 convolutional, 2 pooling and 2 fully connected layers as arranged in the figure shown below:

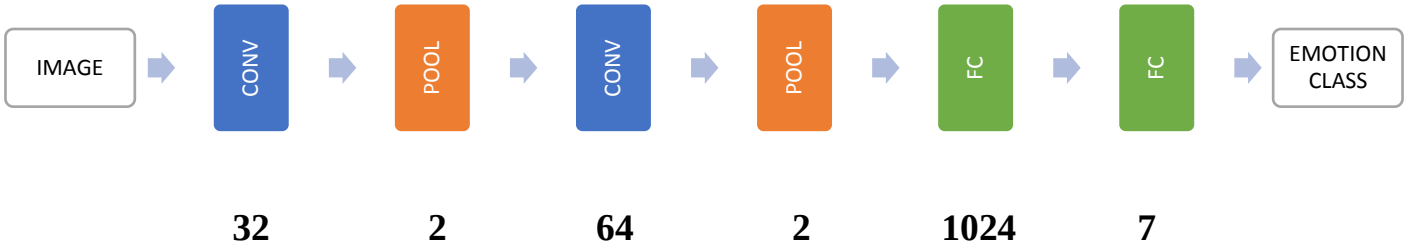


Figure 3.4: Proposed architecture of CNN

3.5 State of the art examples

There are several state of the art architectures of Convolutional Neural Networks. The most common are:

- **LeNet**. It was first introduced by Yann LeCun for the application of OCR and other character recognition from documents. The architecture of LeNet is simple yet powerful enough to produce interesting results. Most attractive feature of LeNet is that it can run even on CPU efficiently.

- **AlexNet**. AlexNet was developed by the winners of ILSVRC 2012 challenge. AlexNet is the network which inspired researchers to use Convolutional Neural Network in the field of Computer Vision at first. The architecture of AlexNet is very similar to LeNet but it is deeper, bigger and have more number of convolutional layers stacked upon one another.

- **ZF Net**. ZFNet was developed by the winners of ILSVRC 2013 challenge. Hyperparameters of AlexNet model were tweaked to modify it and get the resulting ZFNet model. In particular, the size of the convolutional layers were increased in the middle and size of the stride and filter was made smaller on the first layer.

- **GoogLeNet**. GoogLeNet was developed by the winners of ILSVRC 2014 challenge. The main attraction of this design was an *inception module* that reduced the number of parameters greatly. Additionally, Average Pooling was used instead of Fully Connected layers at the top of the Convolutional Network, eliminating a large number of insignificant parameters.

Chapter 4

Results

4.1 Dataset

The dataset used here is known as FER2013 dataset. It was produced by Kaggle for their facial expression recognition challenge in 2013. This is a labeled dataset consists of 35585 pre-cropped grayscale face images having size of 48x48. Each image is labeled with one of the emotion class: happy, anger, fear, surprise, disgust, sad and neutral.

This dataset is divided into three parts as below:

- Training set consisting 28709 images
- Two hold-out sets for post training validation consisting 3589 images per set

Sample images from FER2013 dataset is shown below:

Figure 4.1: Sample images from FER2013 dataset

4.2 Software requirements

Deep learning algorithms requires a lot of computational resources. Use of GPUs for the training of deep learning algorithms especially CNN is a trend now as it greatly reduces the training time. Most commonly used framework for GPU enabled computing is CUDA developed by NVIDIA.

We have not used GPU enabled computing. The system used for the training have following specification:

- Processor: Intel(R) Core(TM) i5-2450M @ 2.50 GHz
- Cores: 2.50 GHz x 4
- RAM: 4 GB
- Operating System: Ubuntu 16.04 LTS (64-Bit)
- Programming Language: Python 2.7
- Deep Learning libraries: TensorFlow 0.12.1

It took us around 110 minutes to train the CNN on this system (for 10 epochs). Cross-validation and testing was quick and took around 15 minutes (for 10 epochs).

4.3 Evaluation metrics

For machine learning and deep learning applications, accuracy is not an efficient metric for evaluation of the performance. There are certain other metrics are used which are described below:

4.3.1 Confusion matrix

It is a matrix that describes the performance of any classification system. For confusion matrix C, any element $C_{i,j}$ will represent the number of observations known to be in group i but classified or predicted to be in class j. The confusion

matrix is a square matrix that show the count value of the true positive, false positive, true negative and false negative.

Consider the case of simple binary classification where only two classes exist: positive class denoted by P and negative class denoted by N. Confusion matrix for this case can be shown as below:

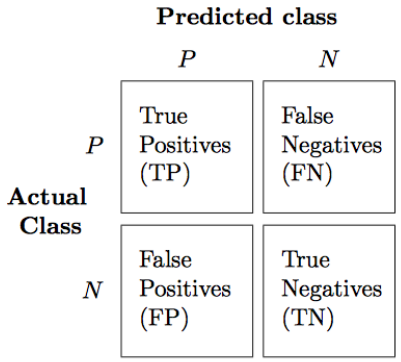


Figure 4.2: Confusion matrix

Note:

For multiclass classification problem, as like the case here, the value of TP, TN, FP and FN can be extracted from the confusion matrix as below:

- For any class, total number of examples will be the sum of the corresponding row
  (*i.e.* TP + FN)
- For any class, total number of FN will be the sum of values in the corresponding row
  (excluding TP) while FP will be the sum of values in the corresponding column
  (excluding TP)
- For any class, total number of TN will be the sum of rows and columns (excluding the row and column corresponding to that class)

4.3.2 Precision

It denotes the fraction of prediction which actually have positive class out of the total positive predicted classifications.

$$PRE = \frac{True\ Positives}{Predicted\ Positives} = \frac{TP}{TP + FP}$$

4.3.3 Recall

It denotes that of all the samples having positive class, what fraction correctly classified as positive class.

$$REC = \frac{True\ Positives}{Actual\ Positives} = \frac{TP}{TP + FN}$$

4.3.4 F1-score

For any classifier, precision and recall should be high. But both the precision and recall can't be high at the same time. Thus, another parameter is used for the analysis called F1-score.

$$F1 = 2\frac{PRE \times REC}{PRE + REC}$$

Value of F1-score lies in the range 0 to 1.

4.3.5 Accuracy

It is defined as the ratio of number of correct predictions to the total number of predictions.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

4.4 Results

4.4.1 Confusion Matrix

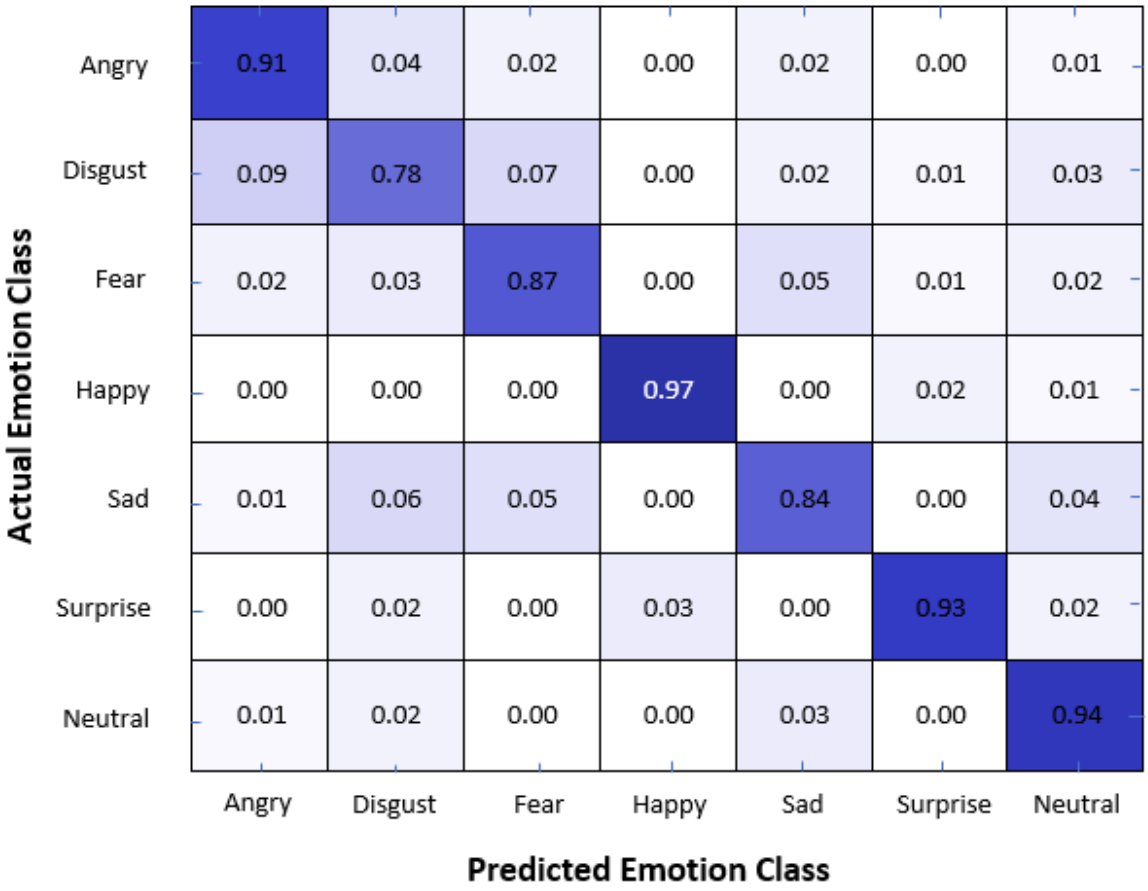


Figure 4.3: Confusion matrix of the result

4.4.2 Accuracy

As mentioned earlier, the accuracy is the ratio of the number of correct prediction to the total number of predictions.

Number of correct predictions will be the sum of values of the diagonal (representing TP) in the confusion matrix while the total number of predictions will be the sum of all the elements of the confusion matrix.

Thus, from the above-mentioned formula and from resulted confusion matrix, the value of the accuracy is ***89.14%.***

4.4.3 Precision, Recall and F1-score

Emotion	Precision	Recall	F1-score
Anger	0.875	0.910	0.892
Disgust	0.821	0.780	0.799
Fear	0.861	0.870	0.865
Happy	0.970	0.970	0.970
Sad	0.875	0.840	0.857
Surprise	0.958	0.930	0.953
Neutral	0.878	0.940	0.907
Average	***0.8911***	***0.8914***	***0.8912*** (*from averaged precision and recall*)

Table 4.1: precision, recall and F1-score of result

4.5 Comparison

For comparison of the performance of our proposed method, we used classical machine learning techniques. Same problem (*i.e.* Human emotion recognition) is solved with the help of three machine learning algorithms. These are:

- MLP
- SVM
- kNN

These classifiers can't be feed directly with the pixel values of face images. They need some kind of feature extractor at first to extract the features of the face images. These features then act as the input for these classifiers. Feature extractors that we used are:

- SIFT
- SURF
- BRISK
- Dense-SIFT

Accuracy is taken as the evaluation metric. The value of the *cluster size* is taken as *500* for all the methods. Accuracies (in %) for above mentioned methods is as shown below:

K = 500	**MLP**	**SVM**	**kNN**
SIFT	56.81	70.45	75
SURF	54.54	63.63	68.18
BRISK	36.36	59.09	52.27
Dense-SIFT	27.27	43.18	38.63

Table 4.2: Results of emotion recognition system with machine learning techniques.

The maximum accuracy achieved with the classical machine learning methods is 75%. Accuracy achieved by deep learning technique *i.e.* CNN (Convolutional Neural Network) is 89.14%. Thus, deep learning techniques outperform the classical machine learning techniques in complex classification tasks such as Human Emotion Recognition.

Chapter 5

Conclusion and Future Scope

The goal of this project was to classify the human emotions with the help of deep learning techniques. In particular, we used convolutional neural networks as they are known for their ability to find complex patterns in images very efficiently. With the help of proposed architecture, we successfully classified the input image in one of the 7 emotion classes *i.e.* anger, disgust, fear, surprise, sad, happy and neutral with ***89.14% accuracy***.

Emotion class 'happy' is classified most efficiently with F1-score of ***0.970*** while the most badly affected class is 'disgust'. The F1-score for the classification of 'disgust' class is ***0.799.*** There is a key factor which affected the classification of 'disgust' class badly. There were very less number of samples for 'disgust' class as compared to other classes. This issue can be addressed in future implementations. Another viable solution for this issue is to merge the 'disgust' class with 'anger' as there is not much difference in both the classes. It is also clear from the confusion matrix that for 'disgust' class, most of the ***False Negatives*** lies in the 'anger' category.

Another factor that imposed the constraints over performance of our system was the computational resources. As we all know deep learning techniques are computationally very expensive especially CNN. Training of deep learning algorithms with the help of GPUs is necessary to fine tune the hyperparameters of the network. As we didn't use the GPUs, the training time was large. It took us

around 110 minutes for training with 10 epochs only. So, use of GPUs is highly recommended for future implementations.

Performance can be improved by taking much larger dataset. We used the FER2013 dataset by Kaggle which consists of only 35585 pre-cropped face images. CNNs performs very well if trained with a large dataset as compared with the previous counterpart. In the dataset we used, the face region was already cropped and centralized so no preprocessing needed. But this may not be the case with real world problems. If we want to deploy a system for real world application, preprocessing is a must before feeding the face images into the CNNs.

BIBLIOGRAPHY

1. F. Bourel, C.C. Chibelushi and A. A. Low, "Recognition of Facial Expressions in the Presence of Occlusion," *Proc. of the Twelfth British Machine Vision Conference,*vol. 1, pp. 213–222, 2001.

2. M. Pardas and A. Bonafonte, "Facial animation parameters extraction and expression recognition using Hidden Markov Models," *Signal Processing: Image Communication,*vol. 17, pp.675–688, 2002.

3. Cohen, N. Sebe, A. Garg, L.S. Chen, and T.S. Huang, "Facial Expression Recognition From Video Sequences: Temporal and Static Modeling", *Computer Vision and Image Understanding,*vol. 91, pp. 160-187, 2003.

4. M.S. Bartlett, G. Littlewort, I. Fasel, and R. Movellan, "Real Time Face Detection and Facial Expression Recognition: Development and Application to Human Computer Interaction," *Proc. CVPR Workshop on Computer Vision and Pattern Recognition for Human-Computer Interaction,*vol. 5, 2003.

5. P. Michel and R. Kaliouby, "Real Time Facial Expression Recognition in Video Using Support Vector Machines," *Proc. 5th Int. Conf. Multimodal Interfaces,*Vancouver, BC, Canada, pp. 258–264, 2003.

6. M. Pantic and J.M. Rothkrantz, "Facial Action Recognition for Facial Expression Analysis from Static Face Images," *IEEE Trans. Systems, Man arid Cybernetics Part B,*vol. 34, no. 3, pp. 1449-1461, 2004.

7. Buciu and I. Pitas, "Application of Non-Negative and Local Non Negative Matrix Factorization to Facial Expression Recognition," *Proc. ICPR*, pp. 288–291, Cambridge, U.K., Aug. 23–26, 2004.

8. M. Pantic, I. Patras, "Detecting facial actions and their temporal segments in nearly frontal-view face image sequences," *Proc. IEEE conf. Systems, Man and Cybernetics,*vol. 4, pp. 3358-3363, Oct. 2005

9. W. Zheng, X. Zhou, C. Zou, and L. Zhao, "Facial Expression Recognition Using Kernel Canonical Correlation Analysis (KCCA)," *IEEE Trans. Neural Networks,*vol. 17, no. 1, pp. 233–238, Jan. 2006.

10. K. Anderson and P.W. McOwan, "A Real-Time Automated System for Recognition of Human Facial Expressions," *IEEE Trans. Systems, Man, and Cybernetics Part B,*vol. 36, no. 1, pp. 96-105, 2006.

11. P.S. Aleksic, A.K. Katsaggelos, "Automatic Facial Expression Recognition Using Facial Animation Parameters and MultiStream HMMs," *IEEE Trans. Information Forensics and Security,*vol. 1, no. 1, pp. 3-11, 2006.

12. M. Pantic and I. Patras, "Dynamics of Facial Expression: Recognition of Facial Actions and Their Temporal Segments Form Face Profile Image Sequences," *IEEE Trans. Systems, Man, and Cybernetics Part B,*vol. 36, no. 2, pp. 433-449, 2006.

13. N. Sebe, M.S. Lew, Y. Sun, I. Cohen, T. Gevers, and T.S. Huang, "Authentic Facial Expression Analysis," *Image and Vision Computing,*vol. 25, pp. 1856-1863, 2007.

14. Kotsia and I. Pitas, "Facial Expression Recognition in Image Sequences UsingGeometric Deformation Features and Support Vector Machines," *IEEE Trans. Image Processing,*vol. 16, no. 1, pp. 172-187, 2007.

15. J. Wang and L. Yin, "Static Topographic Modeling for Facial Expression Recognition and Analysis," *Computer Vision and Image Understanding,*vol. 108, pp. 19–34, 2007.

16. F. Dornaika and F. Davoine, "Simultaneous Facial Action Tracking and Expression Recognition in the Presence of Head Motion,"*Int. J. Computer Vision,* vol. 76, no. 3, pp. 257–281, 2008.

17. Kotsia, I. Buciu and I. Pitas, "An Analysis of Facial Expression Recognition under Partial Facial Image Occlusion," *Image and Vision Computing,* vol. 26, no. 7, pp. 1052-1067, July 2008.