Heaven's Light is Our Guide

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Rajshahi University of Engineering & Technology, Bangladesh

Facial Expression Recognition using Convolutional Neural Network

Authors
Sourav Roy
133034
&
Mohammad Abir Hossain
133077

Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology

Supervised by
Firoz Mahmud
Assistant Professor
Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology

ACKNOWLEDGEMENT

All praises to the Almighty the creator and controller of everything and by Whose grace we were able to complete the thesis work successfully.

We express the deepest sense of gratitude, sincere appreciation and respect to our thesis supervisor **Firoz Mahmud**, Assistant Professor, Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology, Rajshahi for his sincere and scholastic guidance, constructive suggestion and constant encouragement throughout the experimental period as well as in the preparation of the thesis, without which it could not come to light.

We owe our deepest gratitude to **Dr. Md. Rabiul Islam**, Professor and Head, Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology, Rajshahi for his helps in various ways from the department. We are also grateful to all other teachers for their motivation, suggestion, guidance, inspiration at different times.

We would also like to thank our friends for their support at different times that helped us a lot in finding solution of any problem.

Finally we would like to thank our family members for their constant love, continuous support and encouragement.

November 2018 — Sourav Roy

RUET, Rajshahi — Mohammad Abir Hossain

Heaven's Light is Our Guide

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING Rajshahi University of Engineering & Technology, Bangladesh

CERTIFICATE

*This is to certify that the thesis report entitled "**Facial Expression Recognition Using Convolutional Neural Network**" submitted by **Sourav Roy**, Roll No. **133034** and **Mohammad Abir Hossain**, Roll No. **133077** in partial fulfillment of the requirement for the award of degree of Bachelor of Science in Computer Science & Engineering to Rajshahi University of Engineering & Technology, Bangladesh is a record of the candidates' own work carried out by them under my supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.*

Supervisor	External Examiner
--	--
Firoz Mahmud	
Assistant Professor	
Department of Computer Science &Engineering	Department of Computer Science &Engineering
Rajshahi University of Engineering &Technology	Rajshahi University of Engineering &Technology
Rajshahi-6204	Rajshahi-6204

ABSTRACT

Human facial expression recognition is one of the most effective ways to recognize human emotions. A method is proposed to recognize human emotions through facial expression recognition of front facing images. Convolutional Neural Network architecture has been used in this thesis. Different layers have been designed for feature extraction and classification. To evaluate the performance of the proposed method four well known facial expression datasets are used namely FER 2013, JAFFE, CK+. Correct recognition rate of 62.75%, 90.63% and 88.26%, was found on FER 2013, JAFFE and CK+ dataset respectively which indicates the effectiveness of the proposed method.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1
Introduction

1.1 Motivations

Human beings mainly interact with each other through speech, but sometimes they also interact through body gestures in order to emphasize certain portion of the speech as well as to show emotions. And the most important way to show emotions is facial expression. Thus facial expressions become one of the important parts of communication, where nothing is said verbally by humans. But there are a lot of things to understand about the messages which contain the use of non-verbal communication. According to Professor Albert Mehrabian more than 90 of the human communication is nonverbal [1]. So to understand human emotion becomes a key area of research which ranges from more friendly human-computer interactions to better targeted advertising campaigns.

One of the reasons we chose to focus on the area of facial expressions is because certain facial expressions have universal meaning, and these emotions have been documented for tens and even hundreds of years. Thus, nowadays, most databases containing facial emotions use the same key classification of the human emotions as it was originally presented in a paper by Ekman et al in 1971 - "Constants across cultures in the face and emotion" [2].That paper identified the following six key emotions: anger, disgust, fear, happiness, sadness and surprise. These are the same emotions that are being used by current researchers to identify facial expression in computer vision, or in competitions such as Kaggle’s Facial Expression Recognition Challenge, along with the addition of a seventh, neutral emotion, for classification.

1.2 Related Works

In recent years, researchers have made considerable progress in developing automatic expression classifiers [3, 4, 5]. Some expression recognition systems classify the face into a set of prototypical emotions such as happiness, sadness and anger [6]. Others attempt to recognize the individual muscle movements that the face can produce [7] in order to provide an objective description of the face. The best known psychological framework for describing nearly the entirety of facial movements is the Facial Action Coding System (FACS) [8]. FACS is a system to classify human facial movements by their appearance on the face using Action Units (AU). An AU is one of 46 atomic elements of visible facial movement or its associated deformation; an expression typically results from the accumulation of several AUs [7, 8].Moreover, there have been several developments in the techniques used for facial expression recognition: Bayesian Networks, Neural Networks and the multilevel Hidden Markov Model (HMM) [9, 10]. Some of them contain drawbacks of recognition rate or timing. Usually, to achieve accurate recognition two or more techniques can be combined; then, features are extracted as needed. The success of each technique is dependent on pre-processing of the images because of illumination and feature extraction.

1.3 Applications

There has been a growing interest in improving all aspects of the interaction between humans and computers in recent years. This emerging field has been a research interest for scientists from several different scholastic tracks, i.e., computer science, engineering, psychology, and neuroscience. These studies focus not only on improving computer interfaces, but also on improving the actions the computer takes based on feedback from the user. The rapid advance of technology in recent years has made computers cheaper and more powerful, and has made the use of microphones and PC-cameras affordable and easily available. The microphones and cameras enable the computer to "see" and "hear," and to use this information to act. A good example of this is the "SmartKiosk" [Garg et al., 2000a]. It is argued that to truly achieve effective human-computer intelligent interaction (HCII), there is a need for the computer to be able to interact naturally with the user, similar to the way human-human interaction takes place. Human beings possess and express emotions in everyday interactions with others. Emotions are often reflected on the face, in hand and body gestures, and in the voice, to express our feelings or likings. Facial expressions and vocal emotions are commonly used in everyday human-to-human communication, as one smiles to show greeting, frowns when confused, or raises one's voice when enraged. People do a great deal of inference from perceived facial expressions: "You look tired," or "You seem happy." The fact that we understand emotions and know how to react to other people's expressions greatly enriches the interaction. There is a growing amount of evidence showing that emotional skills are part of what is called "intelligence" [11]. Computers today, on the other hand, are still quite "emotionally challenged." They neither recognize the user's emotions nor possess emotions of their own. Psychologists and engineers alike have tried to analyze facial expressions in an attempt to understand and categorize these expressions. This knowledge can be for example used to teach computers to recognize human emotions from video images acquired from built-in cameras. In some applications, it may not be necessary for computers to recognize emotions. For example, the computer inside an automatic teller machine or an airplane probably does not need to recognize emotions. However, in applications where computers take on a social role such as an “instructor,” “helper,” or even “companion,” it may enhance their functionality to be able to recognize users’ emotions. In her book, Picard [12] suggested several applications where it is beneficial for computers to recognize human emotions. For example, knowing the user’s emotions, the computer can become a more effective tutor. Synthetic speech with emotions in the voice would sound more pleasing than a monotonous voice. Computer “agents” could learn the user’s preferences through the users’ emotions. Another application is to help the human users monitor their stress level. In clinical settings, recognizing a person’s inability to express certain facial expressions may help diagnose early psychological disorders.

1.4 Basic Facial Expressions

Basically, human facial expressions can be categorized into seven primary classes such as neutral, happy, angry, fear, disgust, sad and surprised [2]. Figure 1.1 shows sample images of a subject with seven basic facial expressions from JAFFE dataset [13]. Although different datasets use different number of facial expressions by including or ignoring some of these seven expressions, these are the basic seven expressions most widely used for dealing with facial expressions and included in most datasets.

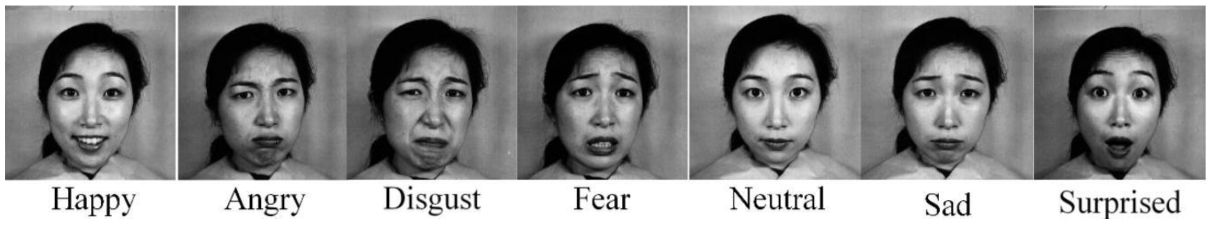

Figure 1.1: Basic Human Facial Expressions from JAFFE Dataset [13]

1.5 Thesis Organisation

Chapter 1: Introduction

In this chapter the basic introductory discussion has been pictured. The discussion covered the reason why human facial expression is so important, the application of human facial expression and the existing methods to classify facial expressions and also the organisation of the book.

Chapter 2: Literature Review

In this chapter the methods which have been used for recognizing human facial expression for years are discussed. Some of the existing works are also enlisted in this chapter.

Chapter 3: Convolutional Neural Network

This chapter broadly discuss about the classification method namely Convolutional Neural Network which has been used in our thesis work. The various terminology of CNN has been discussed in this chapter and the mathematical derivations are also included.

Chapter 4: Implementation

In this chapter the architectures of convolutional Neural Network which are used in our work are discussed. How different architecture works for different datasets are pictured in this chapter.

Chapter 5: Result and Analysis

The experimental result of different datasets has been depicted in this chapter. Why the performance differs from one dataset to another is also discussed briefly in this chapter.

Chapter 6: Conclusion

In this chapter the overall thesis work has been summarized. Also the limitation of our proposed method and the future scope of work has been described in this chapter

1.6 Conclusion

The reasons for choosing this work as the thesis work and the importance of facial expression recognition is discussed in this chapter. Where the facial expression recognition can be used in our real life some such examples and sectors are also discussed in brief. Finally, the structure of this thesis book is mentioned in the last section for the overview of the book at a glance.

Chapter 2

Literature Review

2.1 Overview

In this chapter different methods used for facial expression recognition have been discussed which have been used for years. They are Naïve Bayes Classifier, Support Vector Machine, Gabor filter, Extreme Learning Machine, Local Binary Pattern, K- Nearest Neighbor etc.

2.2 Naive Bayes classifier

A simple technique for construction classifier is naive Bayes model which assigns class label to problem instance, represented as vectors of features values, where class label is drawn from finite set. It is family of algorithm based on the common principle, and all naive Bayes classifiers assume that the particular feature value is independent of the any other feature values of the class variables. For example, consider an apple, red, round, and diameter is 3”. The colors, roundness, and diameter features are independently contributed for a naive Bayes classifier. In practical application, parameter estimation for naive Bayes model uses the maximum method.

Only a small amount of training data are required to estimate the parameter necessary for classification which is the advantage of naive Bayes model. A family of simple probabilistic classifiers based on applying Bayes theorem with strong independent assumption between the features is done in machine learning. With more advanced methods including support vector machine,[14] this domain is found to be competitive, with appropriate preprocessing technique. An automatic medical diagnosis is one of its applications.

$$\{P(outcome\ given\ that\ we\ know\ some\ evidence)\} = \frac{P(evidence\ given\ that\ we\ know\ the\ outcome) times}{prob(outcome), scaled\ by\ the\ P(evidence)}$$

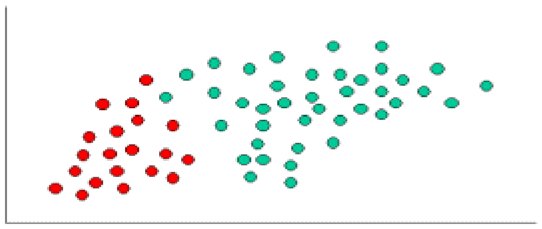

Figure 2.1: Representatoin of Naive Bayes classifier

2.3 Gabor filter

In image processing, a Gabor filter, named after Dennis Gabor, is a linear filter used for texture analysis, which means that it basically analyzes whether there are any specific frequency content in the image in specific directions in a localized region around the point or region of analysis. Frequency and orientation representations of Gabor filters are claimed by many contemporary vision scientists to be similar to those of the human visual system, though there is no empirical evidence and no functional rationale to support the idea. They have been found to be particularly appropriate for texture representation and discrimination. In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal wave. Gabor filter is the implementation of the Gabor transform which is a short term Fourier transformation with Gaussian window for analysis in the spatial domain. The distortion information of content adaptive image steganography incorporates the texture information of the image. On embedding there causes texture anomaly in an image. This texture anomaly can be characterized by the Gabor output obtained by 2D Gabor filtering. The two dimensional Gabor filter represents the texture information because of its spatial selectivity and orientation [15].

For obtaining the Gabor residuals u(x,y), convolution of an image I(x,y) is done with a two dimensional Gabor function g(x,y) as represented in equation (2.1) and in equation (2.2).

$$u(x,y) = \iint_{\delta} I(\alpha,\beta) g(x-\alpha, y-\beta) d\alpha d\beta \tag{2.1}$$

Where, the set of image points are (x,y) ∈ δ, δ and the integral variables are α and β

g(x,y) is the Gabor function.

$$g_{\lambda,\theta\varphi,\sigma,\gamma}(x,y) = \exp\left(-\frac{(x^2+y^2y^2)}{2\sigma^2}\right) \quad \cos(2\pi\frac{x}{\lambda}+\varphi) \tag{2.2}$$

Where

$$x = a\cos\theta + b\sin\theta$$

$$y = -a\ \cos\theta + b\sin\theta$$

And

λ= Wavelength of Gabor function cosine factor.

θ= Orientation of Gabor function normal to the parallel stripes.

φ= Phase offset of the of Gabor function cosine factor.

σ= Standard deviation sigma of Gaussian factor.

γ= Ellipticityof the Gaussian factor.

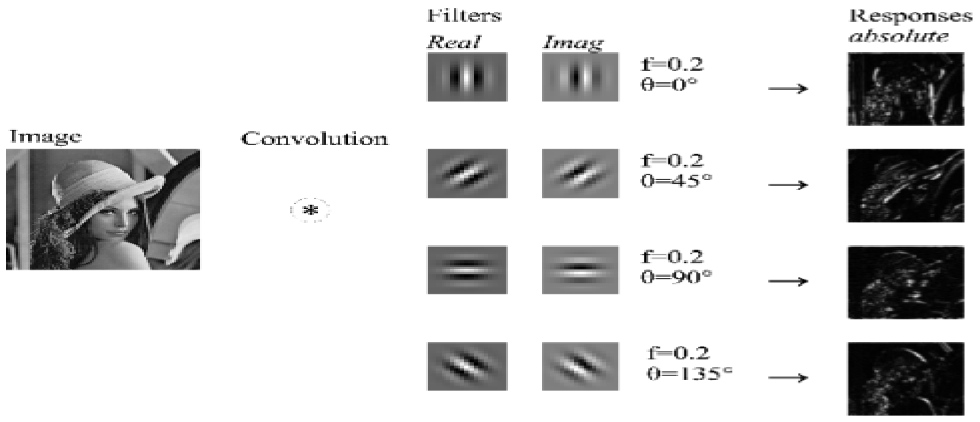

Figure 2.2: Feature extraction using a bank of Gabor filters

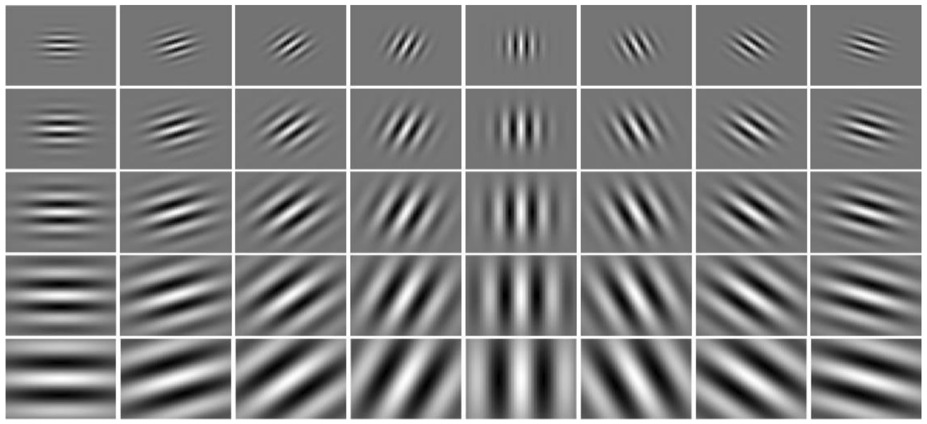

Figure 2.3: Gabor feature extraction

2.4 Extreme learning machine

Huang et al. [16] proposed Extreme Learning Machine, which in an improved feedforward neural network that can randomly generate weights and thresholds, and only the number of neurons in the hidden layer are arranged. It doesn't need to set the other parameters, and it can obtain optimal solution after learning. Jia, Bo, et al. [17] describe a technique to make matrices directly as input 2DELM can be used instead of commonly used vector. 2DELM take the matrices as input features without vectorization. Uçar et al. [18] was proposed a novel algorithm for facial expression recognition by integrating curvelet transformation and online sequential extreme learning machine (OSELM) with radial basis function (RBF) hidden node, where spherical clustering was employed to the features set to automatically determine optimal hidden node number and radial basis function (RBF) hidden node parameters of OSELM. Claude et al. [19] presented a high-level overview of automatic expression recognition by highlighting the main system components and some research challenges. To solve the problem of large amount of data and efficiency problem in the processing of expression recognition, a method of facial expression recognition based on ELM is proposed. General procedure is divided into three parts. First, the images are preprocessed, including face detection and segmentation. Secondly, Gabor filter is used for feature extraction and at last ELM is used for facial expression recognition using the extracted features.

Given a single hidden layer of ELM, suppose that the output function of the i-th hidden node is $h_{i(x)=}g(a_i, b_i, x)$ where a_i and b_i are the parameters of the i-th hidden node. The output function of the ELM for SLFNs with L hidden node is

$f_l(x) = \sum_{i=1}^{L} \beta_i \, h_i(\text{x})$ whereβ_i is the output weight of the i-th hidden node.

$h(\text{x}) = [\text{G}(h_i(x) ,\ldots\ldots\ldots h_L(x))]$ is the hidden layer output mapping of ELM

General speaking, ELM is a kind of regularization neural networks but with non-tuned hidden layer mappings.

2.5 KNN

The k-nearest neighbor algorithm has been proved to be effective for expression recognition because of closest training examples in the feature space. KNN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. According to the needs of special vision, it can adjust the spatial and frequency properties to face expression characteristic wanted, so Gabor filter wavelet is suitable for people face analysis and treatment of expression[20][21].

In K-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors. K-NN comes under instance-based learning, or lazy learning, where the function is only approximated locally and all evaluation is deferred until classification. The KNN algorithm is among the simplest of all machine learning algorithms in the terms of classification and regression, it can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones.

K-mean algorithm idea is to gather each element to its most close heart-shaped (mean) class.
Basic steps:[22]
a) The object is roughly divided into K initial classes;
b) The object individually is assigned to its closest mean class. The new object class and the loss object class mean is re-calculated;
c) Repeat step 2 until no various elements are in and out. At the beginning, the object isn't assigned to K pre-specified classes. It can also specify K seed points, and then proceed to Step 2. To some extent, the final clustering result dependents on the initial division or seed points selection. However, its class size is little different and it is very sensitive to the dirty data.

Here, Euclidean Distance $=\sqrt[2]{\sum_{i=1}^{n}(x_i - y_i)^2}$

Where

$$x = [x_1, x_2, x_3, \dots\dots, x_n] \text{And}$$

$$y = [y_1, y_2, y_3, \dots\dots, y_n]$$

2.6 SVM

Support vector machine (SVM) are the commonly used tool for effective classification of facial expressions. In machine learning, SVMs, are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

SVM is a beneficial technique for data classification. Consider a training set of instance-label pairs (x_i, y_i), i=1,…..,iwhere $x_i \in R^n$ and y$\in\{1, -1\}$. The SVM requires the solution of the following improvement problem:

$$min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_{i\,,} \tag{2.3}$$

Training vectors are mapped into a higher dimensional space by the function ϕ.

Furthermore $k(x_i, x_j) \equiv \emptyset(x_i)^T \phi(x_j)$ is called the kernel function. The choice of kernel function is also significant when designing SVM- based classifiers. There are two basic kernels that are usually used:

Linear function:

$$k(x_i, x_j) = x_i^T x_j \tag{2.4}$$

Radial basis function (RBF):

$$k(x_i x_j) = (\gamma x_i^T x_j + r)\Lambda d \quad \gamma > 0 \tag{2.5}$$

The prevailing approach for forming a multiclass SVM is to decrease the single multiclass problem into multiple binary classification problems. There are two methods (i) one-versus-all that is applied by a winner-takes-all method, in which the classifier with the highest output function determines the class or (ii) one-versus-one that is applied by a max-wins voting method, in which every classifier determines the instance to one of the two classes, then the vote for the assigned classis increased by one vote, and finally the class with the most votes determines the instance classification. The one-versus-all approach was applied to turn the binary SVM classifiers into a multi-label classifier

2.7 Local Binary Pattern

Many techniques are used for the extraction, which is the most important step to carry out face recognition. Local Binary Pattern (LBP) is also one of the methods used for the feature extraction. A relatively new approach came forward through Ojala et al. in 1996. Through LBP it becomes achievable for the illustration of shape and texture. It is done by the separation of image into many small regions commencing the feature extraction. These features comprises of binary patterns so as to express the surroundings of pixel in region.

The images obtained can be distinguished by evaluating the corresponding distance between their histograms. As per observations from many studies it is concluded that face recognition shows extremely high quality results in terms of both speed and discrimination concert. The LBP operator works among eight neighbors of pixel, by means of center pixel's value as threshold. When the neighbor pixel is having gray value higher than center pixel than the assigned value of that pixel, gets one. The same case happens when the values are same. Then the LBP code is created by concatenation of zeroes to a binary code as shown in Figure 1 or eight ones for the center pixel. In Figure 2 original and cropped images are shown

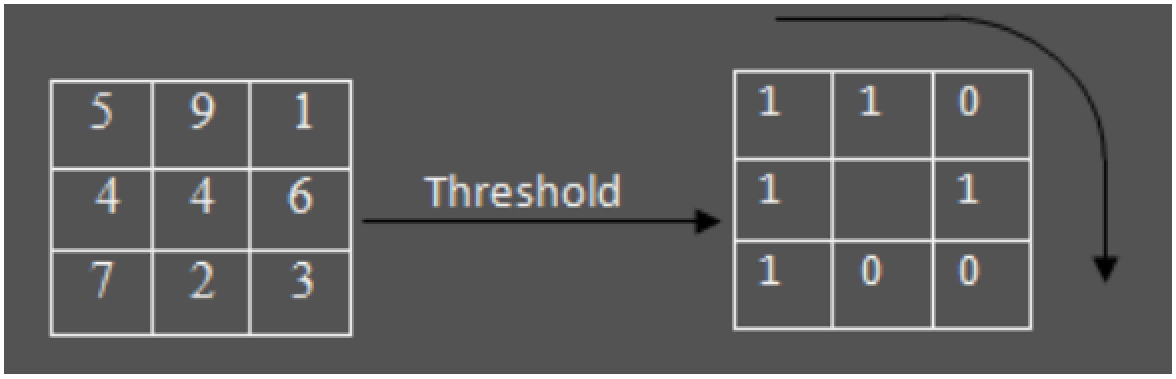

Figure 2.4: The LBP Operator

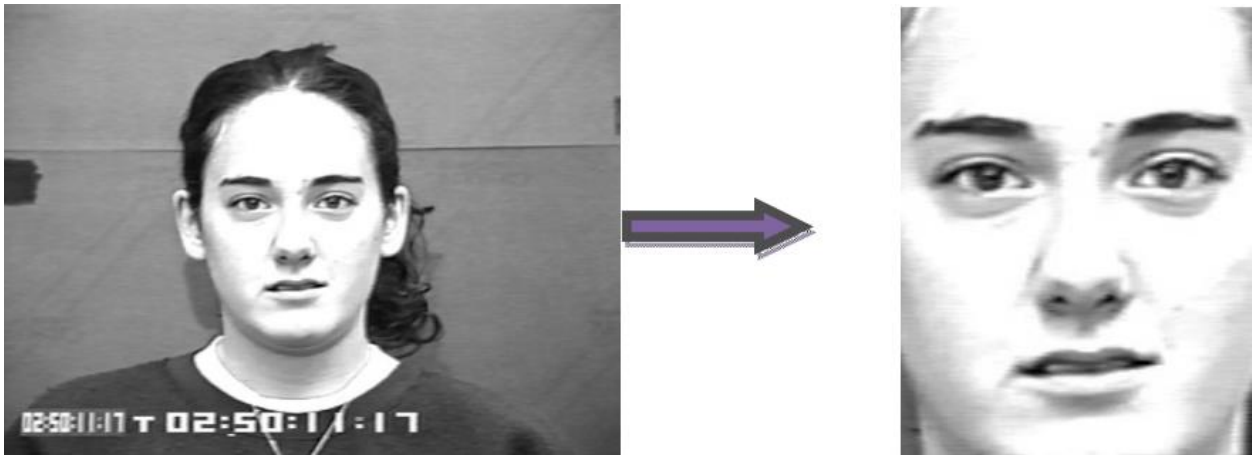

Figure 2.5: The original face and cropped face image

2.8 Previous Works

In this section, work related to previous techniques is discussed. Many researchers in their work have attempted to recognize the facial expression of an individual to the samples in a particular database of faces.

In the work of S. Kumar, et.al. The aim of facial expression recognition (FER) algorithms is to extract discriminative features of a face. However, discriminative features for FER can only be obtained from the informative regions of a face. Also, each of the facial sub regions have different impacts on different facial expressions. Local binary pattern (LBP) based FER techniques extract texture features from all the regions of a face, and subsequently the features are stacked sequentially. This process generates the correlated features among different expressions, and hence affects the accuracy. The authors' approach entails extracting discriminative features from the informative regions of a face. In this view, he proposed an informative region extraction model, which models the importance of facial regions based on the projection of the expressive face images onto the neural face images. This feature extraction method reduces miss-classification among different classes of expressions. Experimental results on standard datasets show the efficiency of the proposed method.[23]

J. V. Patil, et.al. described facial expression as an essential part in communication. It is a challenging task in computer vision as well as in pattern recognition. Facial expression recognition has applications in many fields such as HCI, video games, virtual reality, and analyzing customer satisfaction etc. His proposed system focuses on emotion recognition using facial expressions which are captured by using Intel's Real Sense SR300 camera. This camera detects landmarks on the depth image of a face automatically using Software Development Kit (SDK) of Real Sense camera. Geometric feature based approach is used for feature extraction. The distance between landmarks is used as features and for selecting an optimal set of features brute force method is used. The Proposed system has used Multilayer Perceptron (MLP) neural network algorithm using back propagation method for classification. The proposed system recognizes three facial expressions namely neutral, happy, and surprised. The recognition rate achieved is 93.33%.[24]

J. d. A. Fernandes, et.al. in his article presented two facial geometric-based approaches for facial expression recognition using support vector machines. The first method performed an experimental research to identify the relevant geometric features for human point of view and achieved 85% of recognition rate. The second experiment employed the Correlation Feature Selection and achieved 96.11% of recognition rate. All experiments were carried out with CohnKanade database and the results obtained are compatible with the state-of-the-art in this in this research area [25]

L. A. Jeni, et.al. worked on pose invariance in his article. He rendered 3D emotional database with different poses using BU 4DFE database, fitted 3D CLM, transformed the 3D shape to frontal pose and evaluated the outputs of our classifier. Results show that the high quality classification is robust against pose variations. The superior performance suggests that shape, which is typically neglected or used only as side information in facial expression categorization, could make a good benchmark for future studies.[26]

S. Jain, et.al. proposed that Conditional Random Fields (CRFs) can be used as a discriminative approach for simultaneous sequence segmentation and frame labeling. Latent-Dynamic Conditional Random Fields (LDCRFs) incorporates hidden state variables within CRFs which model sub-structure motion patterns and dynamics between labels. Motivated by the success of LDCRFs in gesture recognition, he proposed a framework for automatic facial expression recognition from continuous video sequence by modeling temporal variations within shapes using LDCRFs. The proposed approach outperforms CRFs for recognizing facial expressions. Using Principal Component Analysis (PCA) we study the reparability of various expression classes in lower dimension projected spaces. By comparing the performance of CRFs and LDCRFs against that of Support Vector Machines (SVMs). He demonstrated that temporal variations within shapes are crucial in classifying expressions especially for those with a small range of facial motion like anger and sadness.[27]

H. Li, et.al. in his work emphasized on Cross-age face recognition as it has remained a popular research topic because the sophisticated facial change across age disables regular face recognition systems. Widely applied in age-related tasks, the hidden factor analysis (HFA) model decomposes face feature into independent age and identity factors. However, the hypothesis that the identity and age factors are independent is not in accordance with the fact that aging has different appearance changes on different people's faces. To address this problem, this letter presents a novel method for cross-age face recognition, called age-identity modified HFA, which exploits a new latent factor modeled as a linear combination with the age factor and the identity factor. Hence, the cross-age identity information can be extracted and separated preferably. A maximum likelihood strategy is proposed to judge which gallery face has the same identity with the probe image, while we do not need to know what the probe identity is. Extensive experiments are performed on the benchmark aging datasets MORPH and FG-Net, and the recognition rate of our method outperforms HFA by 10.4% and 1.15%, respectively.[28]

S. An and Q. Ruan, et.al. calculated the difference of gray value standard between neighboring pixels and the center pixel as a threshold to binary instead of the traditional LBP operation which only comparison of size between neighboring pixels and the center pixel. After he get the LTBP feature, fuse the LTBP and HOG (Histogram of Oriented Gradient) features to get multi-feature fusion for 3D facial expression recognition. His algorithm of 3D facial expression recognition comprises three steps: (1) extracting two sets of feature vectors and establishing the correlation criterion function between the two sets of feature vectors; (2) solving the two sets canonical projective vectors and extracting their canonical correlation features by the framework of canonical correlation analysis algorithm; (3) doing feature fusion for classification by using proposed strategy. He had performed comprehensive experiments on the BU-3DFE database which is presently the largest available 3D face database. He achieved verification rates of more than 90% for the 3D facial expression recognition.[29]

J. Kumari, et.al. in her work stated that Facial Expression is the easy way of telling/showing inner feelings. The Facial Expression Recognition system has many applications including human behavior understanding, detection of mental disorders, synthetic human expressions and many more. This paper presents a quick survey of facial expression recognition as well as a comparative study of various features on JAFFE and CK datasets. It mainly focuses on appearance based techniques. Recently, HOG has been widely used for feature extraction in image. It is found in our experiment that HOG feature gives comparable good recognition rate in facial expression recognition. Fusion of LBP with LGC and Fusion of HOG with other features like LDP and wavelets also improved their respective recognition rates.[30]

The work of T. W. Shen et.al. proposes that the Facial expressions recognition has gained a growing attention from industry and also academics, because it could be widely used in many field such as Human Computer Interface (HCI) and medical assessment. In this paper, we evaluate the strength of the Light Field Camera for facial expression recognition. The light filed camera can capture the directions of the incoming light rays which is not possible with a conventional 2D camera.. Firstly, a new facial expression dataset is collected by the light field camera. The depth map is estimated and applied on Histogram Oriented Gradient (HOG) to encode these facial components as features. Then, a linear SVM is trained to perform the facial expression classification. Performance of the proposed approach is evaluated using the new dataset with estimated depth map. Experimental results show that significant improvements on accuracy are achieved as compared to the traditional approach.[31]

S. M., et.al. in his article stated human detection in images as a fast growing and challenging area of research in computer vision with its main application in video surveillance, robotics, intelligent vehicle, image retrieval, defense, entertainment, behavior analysis, tracking, forensic science, medical science and intelligent transportation. This paper presents a robust multi-posture human detection system in images based on local feature descriptors such as HOG and BO (Block Orientation). The proposed system employs LLE method to achieve dimensionality reduction on the Hog feature descriptors and thus reduce time complexity. Performance of the proposed method is evaluated using feature and classifier based schemes with different datasets. By using classifier based schemes, fast additive SVM outperforms other SVM classifiers. The combined feature vector can retain precision of HOG as well as improve the detection rate. The experiment results on INRIA person, SDL dataset, and TUDB russels dataset demonstrate that combined feature vector along with LLE and fast additive SVM significantly improves the performance.[32]

Whereas Yunsheng Jiang, et.al. in his paper presents an effective combination models with certain combination features for human detection. In the past several years, many existing features/models have achieved impressive progress, but their performances are still limited by the biases rooted in their self-structures, that is, a particular kind of feature/model may work well for some types of human bodies, but not for all the types. To tackle this difficult problem, combined certain complementary features/models together with effective organization/fusion methods. Specifically, the HOG features, color features and bar-shape features are combined together with a cell-based histogram structure to form the so-called HOG-III features. Moreover, the detections from different models are fused together with the new proposed weighted NMS algorithm, which enhances the probable “true” activations as well as suppresses the overlapped detections. The experiments on PASCAL VOC datasets demonstrate that, both the HOG-III features and the weighted-NMS fusion algorithm are effective (obvious improvement for detection performance) and efficient (relatively less computation cost).[33]

2.9 Conclusion

In this chapter, we have discussed about the theoretical background of the facial expression recognition system. We have also discussed about some generalized process of human facial expression recognition.

Chapter 3

Convolutional Neural Network

3.1 Introduction

Convolutional Neural Networks are very similar to ordinary Neural Networks.They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. Softmax) on the last (fully-connected) layer and all the tricks we developed for learning regular Neural Networks still apply. So the basic change is ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.[34]

3.2 Architecture Overview

In Regular Neural Nets, Neural Networks receive an input (a single vector), and transform it through a series of *hidden layers*. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections. The last fully-connected layer is called the “output layer” and in classification settings it represents the class scores.

Regular Neural Nets don’t scale well to full images. In Fer2013, images are only of size 48x48 (48 wide, 48 high,), so a single fully-connected neuron in a first hidden layer of a regular Neural Network would have 48*48 = 2304 weights. This amount still seems manageable, but clearly this fully-connected structure does not scale to larger images. For example, an image of more respectable size, e.g. 200x200x3, would lead to neurons that have 200*200*3 = 120,000 weights. Moreover, we would almost certainly want to have several such neurons, so the parameters would add up quickly! Clearly, this full connectivity is wasteful and the huge number of parameters would quickly lead to overfitting. [34]

Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth**.** (The word *depth*here refers to the third dimension of an activation volume, not to the depth of a full Neural Network, which can refer to the total number of layers in a network.). As we will soon see, the neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner. Here is a visualization:

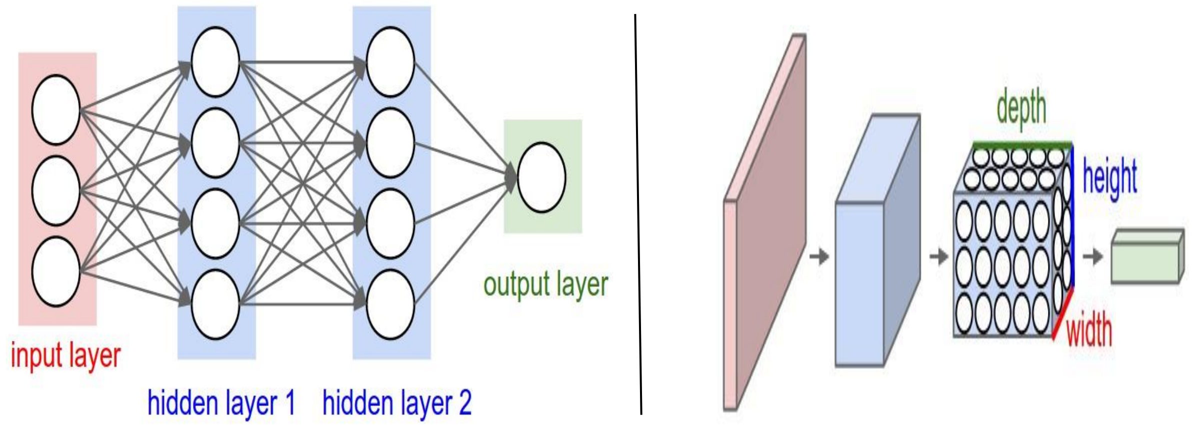

Figure 3.1: Visualization of a Neural Network. Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels). [34]

3.3 Layers used to build ConvNets

As we described above, a simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. We use three main types of layers to build ConvNet architectures: Convolutional Layer**,** Pooling Layer**,** and Fully-Connected Layer (exactly as seen in regular Neural Networks). We will stack these layers to form a full ConvNet architecture. [34]

Example Architecture:A simple ConvNet for FER-2013 classification could have the architecture [INPUT - CONV - RELU - POOL - FC]. In more detail:

- INPUT [48x48] will hold the raw pixel values of the image, in this case an image of width 48 and height 48.
- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [48x48x12] if we decided to use 12 filters.

- RELU layer will apply an element wise activation function, such as the (max(0,x)) thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).
- POOL layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [24x24x12].
- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x7], where each of the 7 numbers correspond to a class score, such as among the 7 categories of FER-2013. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

In this way, ConvNets transform the original image layer by layer from the original pixel values to the final class scores. Some layers contain parameters and other don't. In particular, the CONV/FC layers perform transformations that are a function of not only the activations in the input volume, but also of the parameters (the weights and biases of the neurons). On the other hand, the RELU/POOL layers will implement a fixed function. The parameters in the CONV/FC layers will be trained with gradient descent so that the class scores that the ConvNet computes are consistent with the labels in the training set for each image[34]. In summary:

- A ConvNet architecture is in the simplest case a list of Layers that transform the image volume into an output volume (e.g. holding the class scores)
- There are a few distinct types of Layers (e.g. CONV/FC/RELU/POOL are by far the most popular)
- Each Layer accepts an input 3D volume and transforms it to an output 3D volume through a differentiable function
- Each Layer may or may not have parameters (e.g. CONV/FC do, RELU/POOL don't)
- Each Layer may or may not have additional hyperparameters (e.g. CONV/FC/POOL do, RELU doesn't)

3.3.1 Convolutional Layer

The CONV layer is the core building block of a Convolutional Network that does most of the computational heavy lifting. The CONV layer's parameters consist of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. For example, a typical filter on a first layer of a ConvNet might have size 5x5x3 (i.e. 5 pixels width and height, and 3 because images have depth 3, the color channels). During the forward pass, we slide (more precisely, convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, or eventually entire honeycomb or wheel-like patterns on higher layers of the network. Now, we will have an entire set of filters in each CONV layer (e.g. 12 filters), and each of them will produce a separate 2-dimensional activation map. We will stack these activation maps along the depth dimension and produce the output volume.

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 hyperparameter called the **receptive field** of the neuron (equivalently this is the filter size). The extent of the connectivity along the depth axis is always equal to the depth of the input volume. It is important to emphasize again this asymmetry in how we treat the spatial dimensions (width and height) and the depth dimension: The connections are local in space (along width and height), but always full along the entire depth of the input volume.[34]

Example 1. For example, suppose that the input volume has size [48x48x3], (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). Notice that the extent of the connectivity along the depth axis must be 3, since this is the depth of the input volume.

Example 2. Suppose an input volume had size [16x16x20]. Then using an example receptive field size of 3x3, every neuron in the Conv Layer would now have a total of 3*3*20 = 180 connections to the input volume. Notice that, again, the connectivity is local in space (e.g. 3x3), but full along the input depth (20).

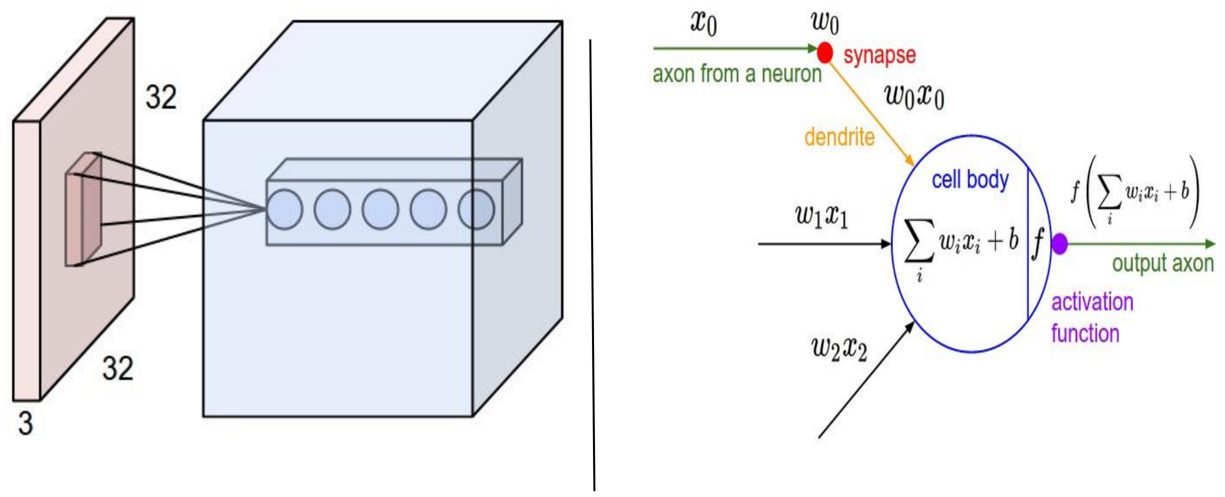

Figure **3.**2: An example input volume in red and an example volume of neurons in the first Convolutional layer. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth (i.e. all color channels). Note, there are multiple neurons (5 in this example) along the depth, all looking at the same region in the input - see discussion of depth columns in text below. **Right:** The neurons from the Neural Network chapter remain unchanged: They still compute a dot product of their weights with the input followed by a non-linearity, but their connectivity is now restricted to be local spatially.[34]

Spatial arrangement

We have explained the connectivity of each neuron in the Conv Layer to the input volume, but we haven't yet discussed how many neurons there are in the output volume or how they are arranged. Three hyperparameters control the size of the output volume: the depth**,** stride and zero-padding. We discuss these next:

1. First, the depth of the output volume is a hyperparameter: it corresponds to the number of filters we would like to use, each learning to look for something different in the input. For example, if the first Convolutional Layer takes as input the raw image, then different neurons along the depth dimension may activate in presence of various oriented edges, or blobs of color. We will refer to a set of neurons that are all looking at the same region of the input as a depth column (some people also prefer the term *fibre*).
2. Second, 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 (or uncommonly 3 or more, though this is rare in practice) then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially.

3. As we will soon see, 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 (most commonly as we'll see soon we will use it to exactly preserve the spatial size of the input volume so the input and output width and height are the same).

We can compute the spatial size of the output volume as a function of the input volume size (W), the receptive field size of the Conv Layer neurons (F), the stride with which they are applied (S), and the amount of zero padding used (P) on the border. The correct formula for calculating how many neurons "fit" is given by

$$\frac{W-F+2P}{S}+1 \tag{3.1}$$

For example for a 7x7 input and a 3x3 filter with stride 1 and pad 0 we would get a 5x5 output. With stride 2 we would get a 3x3 output. Lets also see one more graphical example:

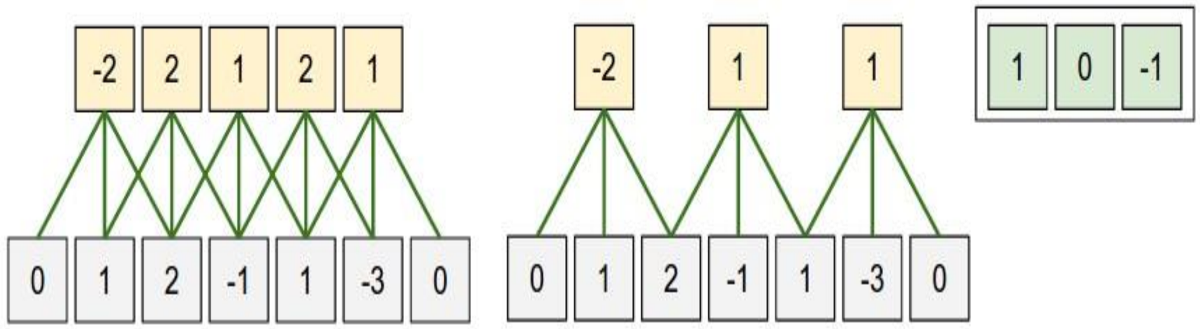

Figure 3.3: Illustration of spatial arrangement.

In this example there is only one spatial dimension (x-axis), one neuron with a receptive field size of F = 3, the input size is W = 5, and there is zero padding of P = 1. **Left:** The neuron strided across the input in stride of S = 1, giving output of size (5 - 3 + 2)/1+1 = 5. **Right:** The neuron uses stride of S = 2, giving output of size (5 - 3 + 2)/2+1 = 3. Notice that stride S = 3 could not be used since it wouldn't fit neatly across the volume. In terms of the equation, this can be determined since (5 - 3 + 2) = 4 is not divisible by 3.
The neuron weights are in this example [1,0,-1] (shown on very right), and its bias is zero. These weights are shared across all yellow neurons.

Use of zero-padding

In the example above on left, note that the input dimension was 5 and the output dimension was equal: also 5. This worked out so because our receptive fields were 3 and we used zero padding of 1. If there was no zero-padding used, then the output volume would have had spatial dimension of only 3, because that it is how many neurons would have "fit" across the original input. In general, setting zero padding to be $P = (F - 1)/2$when the stride is S=1ensures that the input volume and output volume will have the same size spatially. It is very common to use zero-padding in this way.

Constraints on strides

The spatial arrangement hyperparameters have mutual constraints. For example, when the input has size $W = 10$, no zero-padding is used $P = 0$, and the filter size is $F = 3$, then it would be impossible to use stride $S = 2$, since$\frac{W-F+2P}{S} + 1 = (10 - 3 + 0)/2 + 1 = 4.5$, i.e. not an integer, indicating that the neurons don't "fit" neatly and symmetrically across the input. Therefore, this setting of the hyperparameters is considered to be invalid, and a ConvNet library could throw an exception or zero pad the rest to make it fit, or crop the input to make it fit, or something. As we will see in the ConvNet architectures section, sizing the ConvNets appropriately so that all the dimensions "work out" can be a real headache, which the use of zero-padding and some design guidelines will significantly alleviate.

Real-world example. The Krizhevsky et al. architecture that won the ImageNet challenge in 2012[35] accepted images of size [227x227x3]. On the first Convolutional Layer, it used neurons with receptive field size $F = 11$, stride $S = 4$and no zero padding $P = 0$. Since (227 - 11)/4 + 1 = 55, and since the Conv layer had a depth of $K = 96$, the Conv layer output volume had size [55x55x96]. Each of the 55*55*96 neurons in this volume was connected to a region of size [11x11x3] in the input volume. Moreover, all 96 neurons in each depth column are connected to the same [11x11x3] region of the input, but of course with different weights. As a fun aside, if you read the actual paper it claims that the input images were 224x224, which is surely incorrect because (224 - 11)/4 + 1 is quite clearly not an integer. This has confused many people in the history of ConvNets and little is known about what happened. Probably Alex used zero-padding of 3 extra pixels that he does not mention in the paper.

Parameter Sharing

Parameter sharing scheme is used in Convolutional Layers to control the number of parameters. Using the real-world example above, we see that there are 55*55*96 = 290,400 neurons in the first Conv Layer, and each has 11*11*3 = 363 weights and 1 bias. Together, this adds up to 290400 * 364 = 105,705,600 parameters on the first layer of the ConvNet alone. Clearly, this number is very high.

It turns out that we can dramatically reduce the number of parameters by making one reasonable assumption: That if one feature is useful to compute at some spatial position (x,y), then it should also be useful to compute at a different position (x2,y2). In other words, denoting a single 2-dimensional slice of depth as a depth slice (e.g. a volume of size [55x55x96] has 96 depth slices, each of size [55x55]), we are going to constrain the neurons in each depth slice to use the same weights and bias. With this parameter sharing scheme, the first Conv Layer in our example would now have only 96 unique set of weights (one for each depth slice), for a total of 96*11*11*3 = 34,848 unique weights, or 34,944 parameters (+96 biases). Alternatively, all 55*55 neurons in each depth slice will now be using the same parameters. In practice during backpropagation, every neuron in the volume will compute the gradient for its weights, but these gradients will be added up across each depth slice and only update a single set of weights per slice.if all neurons in a single depth slice are using the same weight vector, then the forward pass of the CONV layer can in each depth slice be computed as a **convolution** of the neuron's weights with the input volume (Hence the name: Convolutional Layer). This is why it is common to refer to the sets of weights as a **filter** (or a **kernel**), that is convolved with the input.

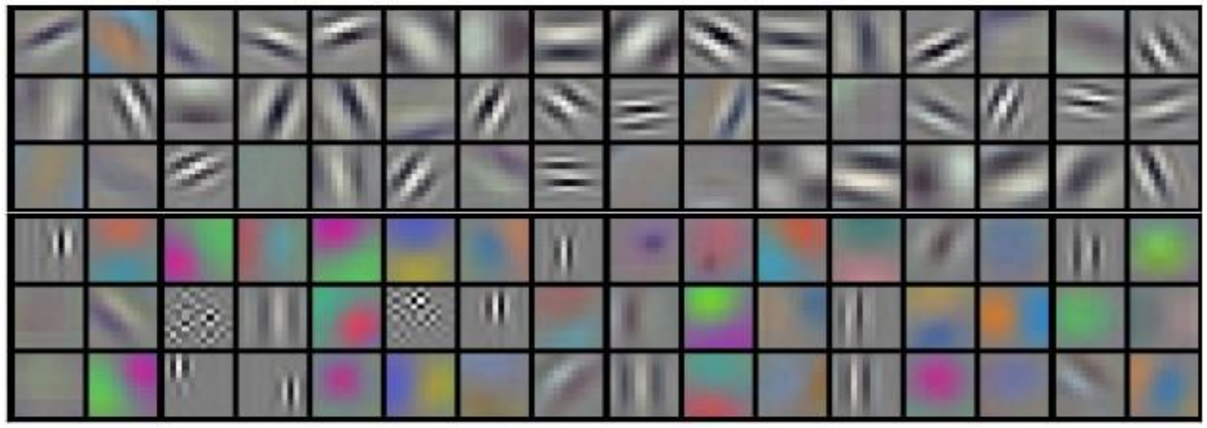

Figure 3.4: Example filters learned by Krizhevsky et al.[35]

Each of the 96 filters shown here is of size [11x11x3], and each one is shared by the 55*55 neurons in one depth slice. Notice that the parameter sharing assumption is relatively reasonable:

If detecting a horizontal edge is important at some location in the image, it should intuitively be useful at some other location as well due to the translationally-invariant structure of images. There is therefore no need to relearn to detect a horizontal edge at every one of the 55*55 distinct locations in the Conv layer output volume.

Sometimes the parameter sharing assumption may not make sense. This is especially the case when the input images to a ConvNet have some specific centered structure, where we should expect, for example, that completely different features should be learned on one side of the image than another. One practical example is when the input are faces that have been centered in the image. You might expect that different eye-specific or hair-specific features could (and should) be learned in different spatial locations. In that case it is common to relax the parameter sharing scheme, and instead simply call the layer a Locally-Connected Layer**.**[34]

Summary

The CONV layer can be summarised as follows:

- Accepts a volume of size $\boldsymbol{W1} \times \boldsymbol{H1} \times \boldsymbol{D1}$
- Requires four hyperparameters:
  - Number of filters K
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W2 \times H2 \times D2$ where:
  - $W2 = \frac{W1-F+2P}{S} + 1$
  - $H2 = \frac{H1-F+2P}{S} + 1$ (i.e. width and height are computed equally by symmetry)
  - $D2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D1$ weights per filter, for a total of $(F \cdot F \cdot D1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W2 \times H2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

A common setting of the hyperparameters is $F = 3, S = 1, P = 1$.

3.3.2 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 amount 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 downsamples 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 (little 2x2 region in some depth slice). The depth dimension remains unchanged. More generally, the pooling layer:

- Accepts a volume of size $W1 \times H1 \times D1$
- Requires two hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W2 \times H2 \times D2$ where:
  - $W2 = \frac{W1-F}{S} + 1$
  - $H2 = \frac{H1-F}{S} + 1$
  - $D2 = D1$
- 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.

It is worth noting that there are only two commonly seen variations of the max pooling layer found in practice: A pooling layer with F=3,S=2 (also called overlapping pooling), and more commonly F=2,S=2. Pooling sizes with larger receptive fields are too destructive.

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.[34]

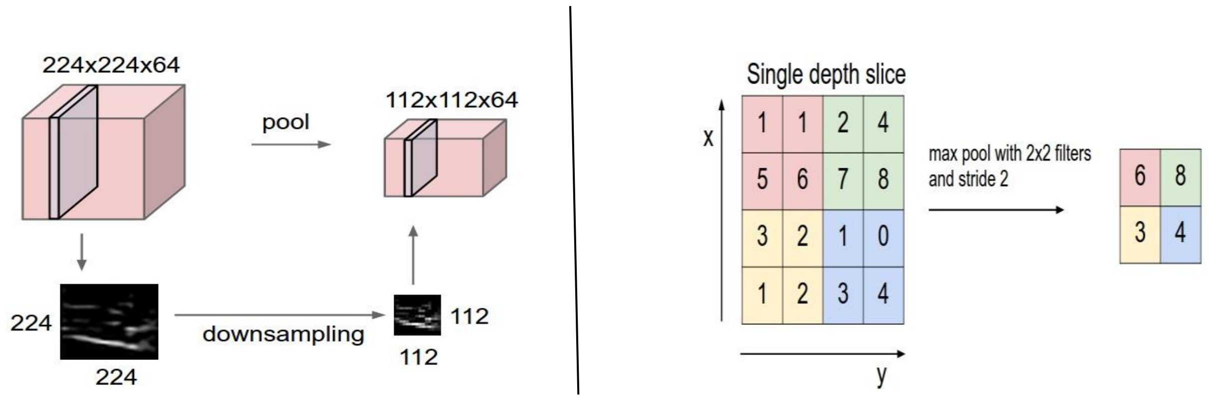

Figure 3.5 : Downsampling by using pooling layer independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to max pooling, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).[34]

3.3.3 Normalization Layer

Many types of normalization layers have been proposed for use in ConvNet architectures, sometimes with the intentions of implementing inhibition schemes observed in the biological brain. However, these layers have since fallen out of favor because in practice their contribution has been shown to be minimal, if any.[34]

3.3.4 Fully- Connected Layer

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset. [34]

3.3.5 Converting FC Layers to CONV Layers

It is worth noting that the only difference between FC and CONV layers is that the neurons in the CONV layer are connected only to a local region in the input, and that many of the neurons in a CONV volume share parameters. However, the neurons in both layers still compute dot products, so their functional form is identical. Therefore, it turns out that it's possible to convert between FC and CONV layers:

- For any CONV layer there is an FC layer that implements the same forward function. The weight matrix would be a large matrix that is mostly zero except for at certain blocks (due to local connectivity) where the weights in many of the blocks are equal (due to parameter sharing).
- Conversely, any FC layer can be converted to a CONV layer. For example, an FC layer with $K = 4096$ that is looking at some input volume of size $7 \times 7 \times 512$ can be equivalently expressed as a CONV layer with $F = 7, P = 0, S = 1, K = 4096$. In other words, we are setting the filter size to be exactly the size of the input volume, and hence the output will simply be $1\,x\,1x\,4096$ since only a single depth column "fits" across the input volume, giving identical result as the initial FC layer.

FC->CONV conversion

Of these two conversions, the ability to convert an FC layer to a CONV layer is particularly useful in practice. Consider a ConvNet architecture that takes a 224x224x3 image, and then uses a series of CONV layers and POOL layers to reduce the image to an activations volume of size 7x7x512 (in an *AlexNet* architecture that we'll see later, this is done by use of 5 pooling layers that downsample the input spatially by a factor of two each time, making the final spatial size 224/2/2/2/2/2 = 7). From there, an AlexNet uses two FC layers of size 4096 and finally the last FC layers with 1000 neurons that compute the class scores. We can convert each of these three FC layers to CONV layers as described above:

- Replace the first FC layer that looks at [7x7x512] volume with a CONV layer that uses filter size $F = 7$, giving output volume [1x1x4096].
- Replace the second FC layer with a CONV layer that uses filter size $F = 1$, giving output volume [1x1x4096]
- Replace the last FC layer similarly, with $F = 1$, giving final output [1x1x1000]

Each of these conversions could in practice involve manipulating (e.g. reshaping) the weight matrix W in each FC layer into CONV layer filters. It turns out that this conversion allows us to "slide" the original ConvNet very efficiently across many spatial positions in a larger image, in a single forward pass.[34]

For example, if 224x224 image gives a volume of size [7x7x512] - i.e. a reduction by 32, then forwarding an image of size 384x384 through the converted architecture would give the equivalent volume in size [12x12x512], since 384/32 = 12. Following through with the next 3CONV layers that we just converted from FC layers would now give the final volume of size [6x6x1000], since (12 - 7)/1 + 1 = 6. Note that instead of a single vector of class scores of size [1x1x1000], we're now getting an entire 6x6 array of class scores across the 384x384 image.

Naturally, forwarding the converted ConvNet a single time is much more efficient than iterating the original ConvNet over all those 36 locations, since the 36 evaluations share computation. This trick is often used in practice to get better performance, where for example, it is common to resize an image to make it bigger, use a converted ConvNet to evaluate the class scores at many spatial positions and then average the class scores.[34]

3.4 ConvNet Architecture

We have seen that Convolutional Networks are commonly made up of only three layer types: CONV, POOL (we assume Max pool unless stated otherwise) and FC (short for fully-connected). We will also explicitly write the RELU activation function as a layer, which applies element wise non-linearity. In this section we discuss how these are commonly stacked together to form entire ConvNets.

3.4.1 Layer Patterns

The most common form of a ConvNet architecture stacks a few CONV-RELU layers, follows them with POOL layers, and repeats this pattern until the image has been merged spatially to a small size. At some point, it is common to transition to fully-connected layers. The last fully-connected layer holds the output, such as the class scores. In other words, the most common ConvNet architecture follows the pattern:

`INPUT -> [[CONV -> RELU]*N -> POOL?]*M -> [FC -> RELU]*K -> FC`

where the `*` indicates repetition, and the `POOL?` indicates an optional pooling layer. Moreover, `N >= 0` (and usually `N <= 3`), `M >= 0`, `K >= 0` (and usually `K < 3`). For example, here are some common ConvNet architectures that follow this pattern:

- `INPUT -> FC`, implements a linear classifier. Here `N = M = K = 0`.
- `INPUT -> CONV -> RELU -> FC`

- INPUT -> [CONV -> RELU -> POOL]*2 -> FC -> RELU -> FC. Here we see that there is a single CONV layer between every POOL layer.
- INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL]*3 -> [FC -> RELU]*2 -> FC Here we see two CONV layers stacked before every POOL layer. This is generally a good idea for larger and deeper networks, because multiple stacked CONV layers can develop more complex features of the input volume before the destructive pooling operation.

We assume that we stack three 3x3 CONV layers on top of each other (with non-linearities in between, of course). In this arrangement, each neuron on the first CONV layer has a 3x3 view of the input volume. A neuron on the second CONV layer has a 3x3 view of the first CONV layer, and hence by extension a 5x5 view of the input volume. Similarly, a neuron on the third CONV layer has a 3x3 view of the 2nd CONV layer, and hence a 7x7 view of the input volume. Suppose that instead of these three layers of 3x3 CONV, we only wanted to use a single CONV layer with 7x7 receptive fields. These neurons would have a receptive field size of the input volume that is identical in spatial extent (7x7), but with several disadvantages. First, the neurons would be computing a linear function over the input, while the three stacks of CONV layers contain non-linearities that make their features more expressive. Second, if we suppose that all the volumes have C channels, then it can be seen that the single 7x7 CONV layer would contain $C \times (7 \times 7 \times C) \ = \ 49\ C^2$parameters, while the three 3x3 CONV layers would only contain $3 \times (C \times (3 \times 3 \times C)\ 27\ C^2$ parameters. Intuitively, stacking CONV layers with tiny filters as opposed to having one CONV layer with big filters allows us to express more powerful features of the input, and with fewer parameters. As a practical disadvantage, we might need more memory to hold all the intermediate CONV layer results if we plan to do backpropagation.

3.4.2 Layer Sizing Patterns

Until now we've omitted mentions of common hyperparameters used in each of the layers in a ConvNet. We will first state the common rules of thumb for sizing the architectures and then follow the rules with a discussion of the notation:

The input layer (that contains the image) should be divisible by 2 many times. Common numbers include 32 (e.g. CIFAR-10), 48, 64, 96 (e.g. STL-10), or 224 (e.g. common ImageNet ConvNets), 384, and 512.

The conv layers should be using small filters (e.g. 3x3 or at most 5x5), using a stride of $S = 1$, and crucially, padding the input volume with zeros in such way that the conv layer does not alter the spatial dimensions of the input. That is, when $F = 3$, then using $P = 1$ will retain the original size of the input. When $F = 5$, $P = 2$. For a general F, it can be seen that $P = (F - 1) / 2$ preserves the input size. If you must use bigger filter sizes (such as 7x7 or so), it is only common to see this on the very first conv layer that is looking at the input image.

The pool layers are in charge of downsampling the spatial dimensions of the input. The most common setting is to use max-pooling with 2x2 receptive fields (i.e.$F = 2$), and with a stride of 2 (i.e. $S = 2$). Note that this discards exactly 75% of the activations in an input volume (due to downsampling by 2 in both width and height). Another slightly less common setting is to use 3x3 receptive fields with a stride of 2, but this makes. It is very uncommon to see receptive field sizes for max pooling that are larger than 3 because the pooling is then too lossy and aggressive. This usually leads to worse performance.
The scheme presented above is pleasing because all the CONV layers preserve the spatial size of their input, while the POOL layers alone are in charge of down-sampling the volumes spatially. In an alternative scheme where we use strides greater than 1 or don't zero-pad the input in CONV layers, we would have to very carefully keep track of the input volumes throughout the CNN architecture and make sure that all strides and filters "work out", and that the ConvNet architecture is nicely and symmetrically wired.
Smaller strides work better in practice. Additionally, as already mentioned stride 1 allows us to leave all spatial down-sampling to the POOL layers, with the CONV layers only transforming the input volume depth-wise.
In addition to the aforementioned benefit of keeping the spatial sizes constant after CONV, doing this actually improves performance. If the CONV layers were to not zero-pad the inputs and only perform valid convolutions, then the size of the volumes would reduce by a small amount after each CONV, and the information at the borders would be "washed away" too quickly.[34]

3.5 Conclusion

In this chapter the classifier which have been used in our thesis work, that is, Convolutional Neural Network has been broadly discussed. The structure of a CNN architecture is discussed with different examples. The mathematical derivation of different layers have also been described in this chapter.

Chapter 4
Implementation

4.1 Introduction

Convolutional neural network (CNN) is a trending classification technique nowadays. Deep CNN can both extract the features of images and classify the extracted features. So we have developed CNNs with variable depths to evaluate the performance of these models for facial expression recognition. We worked on three different Dataset namely, FER-2013, JAFEE and CK+ and got different testing accuracies for them as the datasets have different volume of images. We slightly changed our proposed architecture for each dataset to achieve better performances.

4.2 FER-2013 Dataset

Firstly, We used a dataset provided by Kaggle website, which consists of about 37,000 well-structured 48 x 48 pixel gray-scale images of faces. The images are processed in such a way that the faces are almost cantered and each face occupies about the same amount of space in each image. Each image has to be categorized into one of the seven classes that express different facial emotions. These facial emotions have been categorized as: 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, and 6=Neutral. Figure 1 depicts one example for each facial expression category. In addition to the image class number (a number between 0 and 6), the given images are divided into three different sets which are training, validation, and test sets. There are about 29,000 training images, 4,000 validation images, and 4,000 images for testing [36]. After reading the raw pixel data, we normalized them by subtracting the mean of the training images from each image including those in the validation and test sets.

4.2.1 Proposed Architecture for FER-2013

For the purpose of our thesis, we built a CNN architecture for FER 2013 dataset. This network had six convolutional layers and two FC layer. In the first two convolutional layers, we had 64 5×5 filters, with the stride of size 1, along with batch normalization and max-pooling with a filter size of 2×2 . In the third and fourth convolutional layers, we had 128 5×5 filters, with the stride size 1, along with batch normalization and max-pooling with a filter size 2×2. In the fifth and sixth convolution layers, we used 256 5×5 filters, with batch normalization and max-pooling with filter size 2×2. In the FC layer, we had a hidden layer with 128 neurons and Softmax as the loss function. Also in all the layers, we used Rectified Linear Unit (ReLU) as the activation function. For the training process, we used all of the images in the training set with 50 epochs and a batch size of 64 with learning rate 0.001. To validate our model in each iteration, we used the validation set and to evaluate the performance of the model we used the test set.

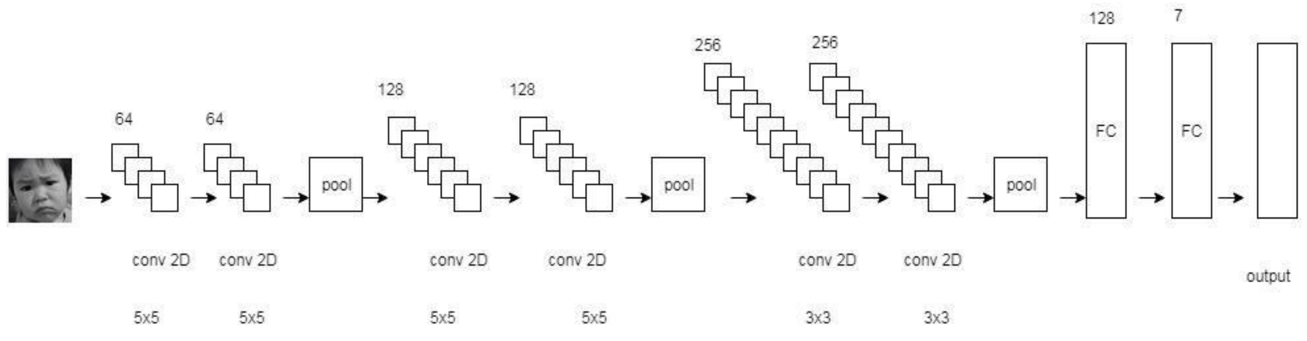

Figure 4.1: Architecture used for classification for FER-2013 dataset

The model summary is as follows:

```
_________________________________________________________________________
Layer (type)                 Output Shape             Param #
=========================================================================-------------
conv2d_1 (Conv2D)            (None, 48, 48, 64)        1664
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 48, 48, 64)        102464
_________________________________________________________________
batch_normalization_1 (Batch (None, 48, 48, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 128)       204928
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 128)       409728
_________________________________________________________________
batch_normalization_2 (Batch (None, 24, 24, 128)       512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 12, 12, 256)       295168
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 256)       590080
_________________________________________________________________
```

```
batch_normalization_3 (Batch (None, 12, 12, 256)       1024
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               1179776
_________________________________________________________________
batch_normalization_4 (Batch (None, 128)               512
_________________________________________________________________
activation_1 (Activation)    (None, 128)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 903
_________________________________________________________________
activation_2 (Activation)    (None, 7)                 0
=================================================================
Total params: 2,787,015
Trainable params: 2,785,863
Non-trainable params: 1,152
```

4.3 JAFFE Dataset

JAFFE dataset contains 213 facial expression images from 10 different Japanese volunteers. The image size is 128x128. There are 7 basic facial expressions of these volunteers. But the number of a facial expression of a volunteer differs from volunteer to volunteer. For example, a volunteer has four happy facial expression images whereas another volunteer has three happy facial expression images. The photos were taken at the Psychology Department in Kyushu University. The dataset was made available from 1998 [13]. Some of the images of the dataset are shown in Figure.

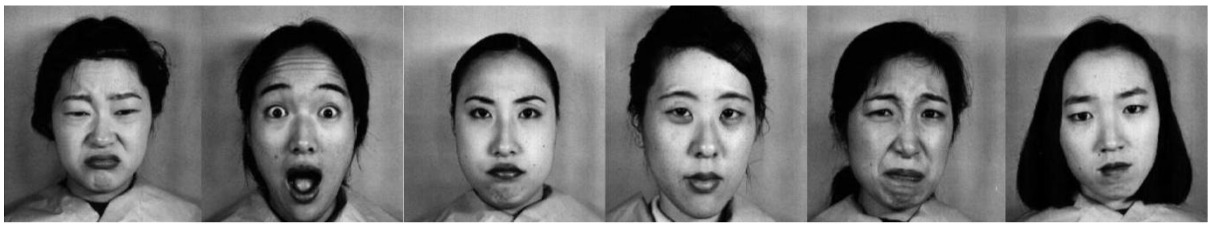

Figure 4.2: Sample Images from JAFFE Dataset [13]

4.3.1 Proposed Architecture for JAFFE Dataset

This network had three convolutional layers and two FC layer. In the first convolutional layer, we had 6 5×5 filters, with the stride of size 1, along with ReLU and max-pooling with a filter size of 2×2 . In the second convolutional layer, we had 16 5×5 filters, with the stride size 1, along ReLU and max-pooling with a filter size 2×2. In the third convolution layer, we used 120 5×5 filters, with Dropout. In the FC layer, we had a hidden layer with 84 neurons. Also in all the layers, we used Rectified Linear Unit (ReLU) as the activation function. Lastly, the second FC layer are classified into seven output classes. In this layer Softmax activation function is used to produce the output.

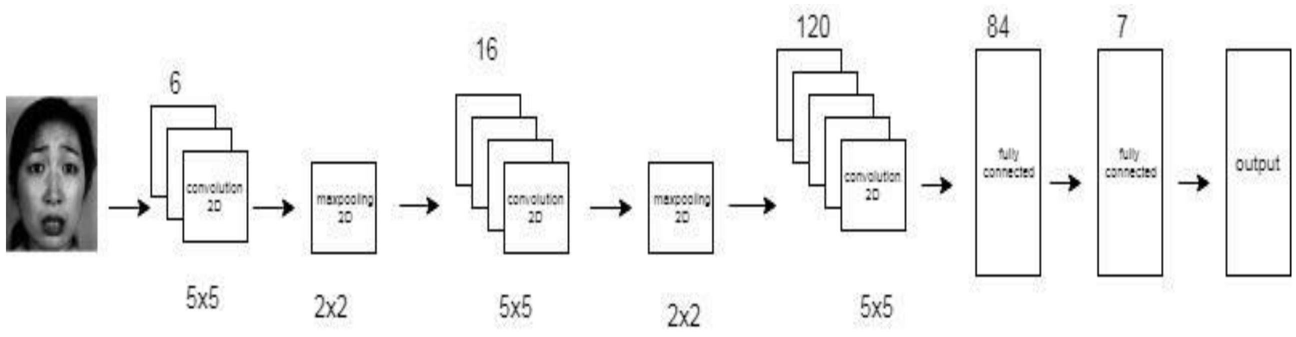

Figure 4.3: Architecture applied on JAFFE dataset

The model summary is as below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_13 (Conv2D)           (None, 128, 128, 6)       456       
_________________________________________________________________
activation_21 (Activation)   (None, 128, 128, 6)       0         
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 64, 64, 6)         0         
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 64, 64, 16)        2416      
_________________________________________________________________
activation_22 (Activation)   (None, 64, 64, 16)        0         
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 32, 32, 16)        0         
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 28, 28, 120)       48120     
_________________________________________________________________
activation_23 (Activation)   (None, 28, 28, 120)       0         
_________________________________________________________________
dropout_10 (Dropout)         (None, 28, 28, 120)       0         
_________________________________________________________________
```

```
flatten_5 (Flatten)          (None, 94080)             0
_________________________________________________________________
dense_9 (Dense)              (None, 84)                7902804
_________________________________________________________________
activation_24 (Activation)   (None, 84)                0
_________________________________________________________________
dropout_11 (Dropout)         (None, 84)                0
_________________________________________________________________
dense_10 (Dense)             (None, 7)                 595
_________________________________________________________________
activation_25 (Activation)   (None, 7)                 0
=================================================================
Total params: 7,954,391
Trainable params: 7,954,391
Non-trainable params: 0
_________________________________________________________________
```

4.4 CK+ Dataset (Extended Cohn-Kanade)

The Extended Cohn-Kanade dataset is an extension of Cohn-Kanade dataset to handle the issues with Cohn-Kanade dataset. The dataset was made available from 2010. 1219 images from the dataset were used for testing the performance of the proposed system. The dataset contains images from different race and ethnicity which gave the opportunity to use image of different type of peoples [37]. For each expression of any person there were several images in the dataset comparing to JAFFE dataset. Sample images from the dataset are shown in Figure 6.2 for demonstration purpose.

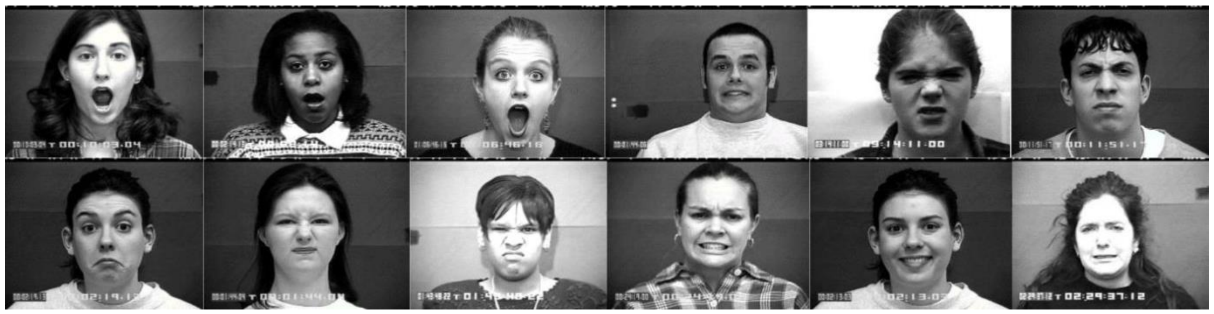

Figure 4.4: Sample Images from CK+ Dataset [37]

4.4.1 Proposed Architecture for CK+ Dataset

In this network, we use three convolutional layers and two FC layer. In the first convolutional layer, we had 64 5×5 filters, with the stride of size 1, along with ReLU and max-pooling with a filter size of 3×3 . In the second convolutional layer, we also used 64 5×5 filters, with the stride size 1, along ReLU and max-pooling with a filter size 3×3. In the third convolution layer, we used 128 4×4 filters, with Dropout. In the FC layer, we used Rectified Linear Unit (ReLU) as the activation function. Lastly, the second FC layer are classified into seven output classes. In this layer Softmax activation function is used to produce the output.

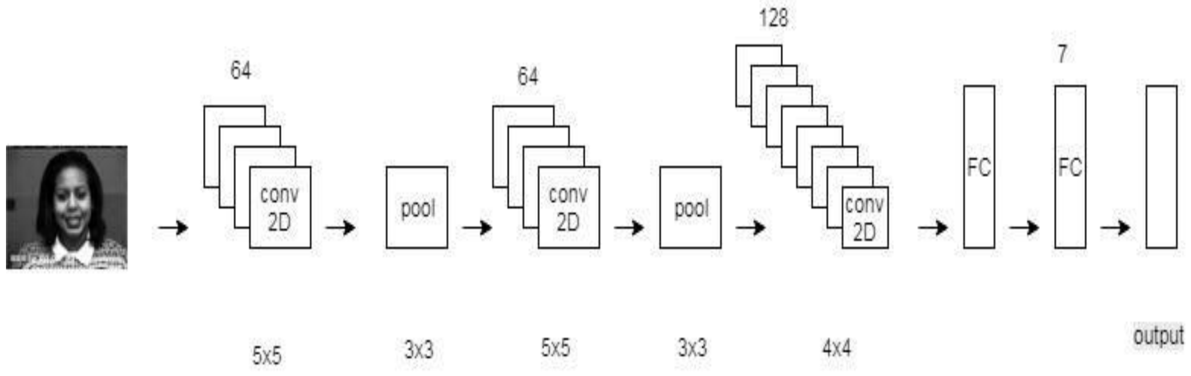

Figure 4.5: Architecture applied on CK+ dataset

4.5 Conclusion

We have implemented three different Convolutional Neural Network architecture for the three datasets. The reason behind this is that, different dataset has different number of images with different features. So we decided to developed both shallow and deep architecture considering the characteristics of the respective dataset. For FER 2013 dataset, we implemented a deep network in comparison with the networks used for other two datasets.

Chapter 5

Result and Performance Analysis

5.1 Experimental Environment

The experimental environment is configured with Intel Core i7 processor, Tesla k80 GPU and 16 GB of RAM. This environment has reduced our training time by keeping better performance of our model.

5.2 Experimental Result

The training and testing images were selected randomly without any kind of relation between them so there was no possibility of any biased recognition rate. Table 5.1 shows the training and testing percentage of images from the datasets and also the training accuracy and the recognition rate for the three datasets.

Table 5.1: Training and Testing Accuracy for Different Dataset

Dataset	**Training Data (%)**	**Testing Data (%)**	**Training Accuracy (%)**	**Testing Acccuracy (%)**
JAFFE	**85**	**15**	**94.73**	**90.63**
CK+	**80**	**20**	**98.54**	**88.26**
FER 2013	**90**	**10**	**98.84**	**62.75**

For each of the Dataset the rate of loss gradually decreased and the recognition rate increased with time. For example, The comparison of the training loss and the validation loss for JAFFE dataset has been pictured in Fig 5.1. The growing recognition rate for both training data and the test data are shown in Fig5.2.

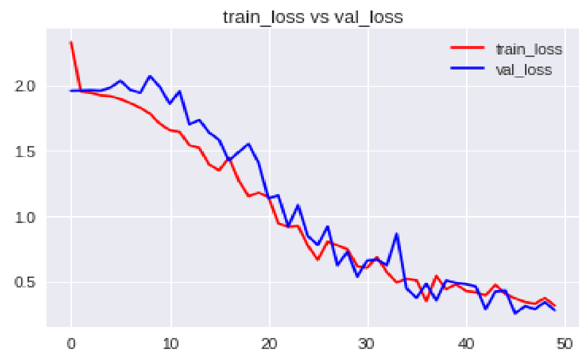

Fig 5.1: Training loss vs Validation loss for JAFFE dataset

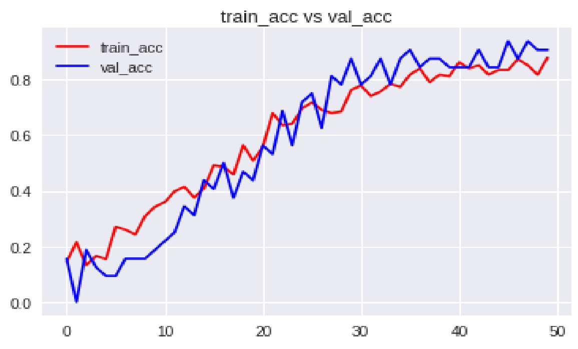

Fig 5.2: Training accuracy vs Validation accuracy for JAFFE dataset

5.3 Result Analysis

The proposed systems performed better on one dataset comparing to others. There could be several reasons behind it. But the main reason was the type and number of the images in the dataset. For example, the best performance was found in JAFEE dataset which contained only 213 images consisting of four, three or two facial expression images of any person with any specific facial expression. So, there was a high probability that the system could be trained very well about that particular expression and testing could result almost accurately. For CK+ dataset there were 1219 images of any expression of any model. So the performance was a bit lower than that of the JAFFE dataset. Lastly, FER 2013 dataset has a huge number of images, in particular 35,887 images. So the testing accuracy of FER 2013 dataset is the least among the three.

Having known about the possible reasons of variation of results now the focus will be on which type of facial expressions are getting tough for the system to recognize properly and as a result the recognition rate is decreasing. To analyze this, we need the help of confusion matrices. Now we will see confusion matrix for JAFFE dataset.

Table 5.2: Confusion Matrix of Recognition Rate on JAFFE Dataset with 7 Basic Facial Expressions

	Angry	Disgust	Fear	Happy	Neutral	Sad	Surprise
Angry	0.96	0.0	0.0	0.0	0.0	0.0	0.0
Disgust	0.0	0.92	0.0	0.0	0.0	0.0	0.0
Fear	0.12	0.0	0.99	0	0.0	0.0	0.0
Happy	0.0	0.0	0.0	0.89	0.0	0.0	0.0
Neutral	0.0	0.0	0.0	0.13	0.87	0.0	0.0
Sad	0.0	0.0	0.0	0.0	0.0	0.85	0.0
Surprise	0.0	0.16	0.0	0.0	0.0	0.0	0.84

5.4 Conclusion

The datasets used to evaluate the performance of the proposed system were briefly discussed in this chapter. The performance of the proposed system was justified by the results obtained using it. The comparison of the performance on different datasets was also illustrated. The possible reason for the variation of performance on different datasets was also discussed.

Chapter 6
Discussion and Conclusion

6.1 Introduction

The experimental results of the proposed method were quite impressive but there still could be some flaw in the proposed method. This chapter will be about the general discussion of the proposed system, discussion about some flaw of the proposed systems and where the proposed system might fail and finally there will be another section for works that could be done to develop the proposed system and make it a more robust one.

6.2 Discussion

The proposed methods consist of different convolutional neural network architectures for different datasets. These architectures can both extract features and classify the images into seven desired classes. The convolution layers mainly calculate the dot product of pixels of the input images. In these layers the linear rectifier (ReLU) activation function is used. Then the pooling layer (Maxpool) is used for downsampling. We designed both the deep and shallow networks according to the number of images in the datasets. For FER 2013 we have used a comparatively deep network as the dataset has huge number of images. And for JAFFE and CK+ dataset we've used a shallow network. After the convolution layer we normalized the data using batch normalization. Then fully connected (FC) layers have been used. The fully connected layers work as hidden layers and classify the images into desired output levels. In FER 2013 firstly a fully connected layer of 128 neurons have been used where in JAFFE fully connected layer of 84 neurons have been used. In both architectures a second fully connected layer have been used which produces the output of seven different classes which represents the seven different facial expressions. Softmax activation function has been used at the fully connected layer to produce the output. The experimental results of the proposed method on different datasets with images from different races, ethnic groups and moreover from different parts of the world would be enough to indicate the system's effectiveness with different images.

6.3 Limitations

Convolutional neural networks and deep belief nets like any neural network model are computationally expensive. But, that is more of a drawback than a weakness. . This can be overcome with better computing hardware such as GPUs and Neuromorphic chips. The weaknesses that these models however have are:

- Missing theory. A theory to explain why and how these deep architecture work is missing. This theory can be relevant in understanding how much data or how many layers are needed to achieve a certain performance.
- Reasoning. A convnet or DBN is blind to logic and reasoning due to lack of knowledge representation within itself.
- Memory. Convnets and DBNs have a problem of *catastrophic forgetting.* That is they tend to forget about a previously trained task if you try to teach them a new one. This is due to the rewriting of connection efficacies or weight by the learning algorithms.
- Unsupervised learning. The long desire of machine learning has been to have machines than can learn on themselves. Convnets and DBNs still have to be supervised, atleast on a high-level, even in a "unsupervised learning" setting. My take on that is that unsupervised learning fails due lack of reasoning and knowledge.

Solving these four issues would overcome the weaknesses of convnets and DBNs. That would be one step closer to realising a true AI vision or goal.

6.4 Future work

To further improve the performance, an unsupervised learning phase might be added before the supervised learning. This technique is particularly useful when the dataset is small. In fact, the winning team of Kaggle facial expression challenge in 2013 use a RBM (Restricted Boltzmann Machine) before the supervised learning phase and achieved an accuracy closed to 70% on the testing dataset. Facial expressions are an effective way to recognize emotions but it is not the only way to recognize emotions. As emotion recognition is the ultimate goal so voice, psychological information could be considered also along with facial expressions to develop a multimodal information fusion system to recognize emotions effectively. This could be an interesting and challenging work in the future in this field.

6.5 Conclusion

This chapter discusses about the proposed system and its ability to achieve an impressive recognition rate. There are also limitations and future. Although achieving an impressive performance the system has some drawbacks discussed in this chapter. Solving the limitations could be a future work for the researchers working in this field and following this work as a guideline. Multimodal information system could also be a challenging, complex and effective work in future to recognize human emotions

REFERENCES

[1] Albert Mehrabian. Silent Messages, University of California Los Angeles, 1971.

[2] P. Ekman and W. V. Friesen. Emotional facial action coding system. Unpublished manuscript, University of California at San Francisco, 1983.

[3] Y. Tian, T. Kanade, and J. Cohn. Recognizing action units for facial expression analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(2), 2001.

[4] M.S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan. Fully automatic facial action recognition in spontaneous behavior. In Proceedings of the IEEE Conference on Automatic Facial and Gesture Recognition, 2006.

[5] M. Pantic and J.M. Rothkrantz. Facial action recognition for facial expression analysis from static face images. IEEE Transactions on Systems, Man and Cybernetics, 34(3), 2004.

[6] G. Littlewort, M. Bartlett, I. Fasel, J. Susskind, and J. Movellan. Dynamics of facial expression extracted automatically from video. Image and Vision Computing, 24(6), 2006.

[7] M.S. Bartlett, G. Littlewort, M.G. Frank, C. Lainscsek, I. Fasel, and J.R. Movellan. Automatic recognition of facial actions in spontaneous expressions. Journal of Multimedia, 2006.

[8] P. Ekman,W. Friesen, Facial Action Coding System: A Technique for the Measurement of Facial Movement, Consulting Psychologists Press, 1978.

[9] Cohen, Ira, et al. ”Evaluation of expression recognition techniques.” Image and Video Retrieval. Springer Berlin Heidelberg, 2003. 184- 195.

[10] Padgett, C., Cottrell, G.: Representing face images for emotion classification. In: Conf. Advances in Neural Information Processing Systems. (1996) 894900.

[11] John D. Mayer Peter Salovey and David R. Caruso, “Emotional Intelligence New Ability or Eclectic Traits?” Copyright 2008 by the American Psychological Association 0003-066X/08/$12.00 Vol. 63, No. 6, 503–517 DOI: 10.1037/0003-066X.63.6.503.

[12] R. W. Picard, “Affective Computing” MIT Media Laboratory; Perceptual Computing; 20 Ames St., Cambridge, MA 02139.

[13] M. J. Lyons, S. Akemastu, M. Kamachi, J. Gyoba, “Coding Facial Expressions with Gabor Wavelets”, *3rd IEEE International Conference on Automatic Face and Gesture Recognition,* pp. 200-205, 1998.

[14] Mencattini A, Salmeri M, Caselli F, Sciunzi B, Lojacono R. Subband variance computation of homoscedastic additive noise in discrete dyadic wavelet transform. Int J Wavelets Multiresolut Inf Process. 2008;6:1–12.

[15] Song X, Liu F, Zhang Z, Yang C, Luo X, Chen L. 2D Gabor Filters-Based Steganalysis of Content-Adaptive JPEG Steganography, Springer Multimedia Tools and Applications. December 2016, pp.1-29.

[16] G.-B. Huang, Q.-Y. Zhu, & C.-K. Siew. "Extreme learning machine: a new learning scheme of feedforward neural networks." Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on, Vol. 2, pp. 985-990, 2004.

[17] Jia, Bo, et al. "Two-Dimensional Extreme Learning Machine." Mathematical Problems in Engineering 2015 (2015).

[18] Uçar, Aysegül, Yakup Demir, and Cüneyt Güzelis. "A new facial expression recognition based on curvelet transform and online sequential extreme learning machine initialized with spherical clustering." Neural Computing and Applications 27.1 (2016): 131-142.

[19] Chibelushi, Claude C., and Fabrice Bourel. "Facial expression recognition: A brief tutorial overview." CVonline: On-Line Compendium of Computer Vision 9 (2003).

[20] J. G. Daugman, "Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression", IEEE Trans. Acoustic, speech and signal processing, Vol. 36 1988, pp.1169-1179.

[21] Lyons Michael J ,Akamatsu Shigeru, Kamachi Miyuki, et al. Coding facial expressions with Gabor wavelets[C] .Proceedings of the3rdIEEEInternational Conference on Automatic Face and Gesture Recognition. Nara. 1998, :200- 205.

[22] Preparata Franco , Shamos Michael Ian. Computational Geometry : anIntroduction[M] . New York : Springer - Verlag , 1988.

[23] S. Kumar, M. K. Bhuyan and B. K. Chakraborty, "Extraction of informative regions of a face for facial expression recognition," in IET Computer Vision, vol. 10, no. 6, pp. 567-576, 9 2016 doi: 10.1049/ietcvi.2015.0273.

[24] J. V. Patil and P. Bailke, "Real time facial expression recognition using RealSense camera and ANN," 2016 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, 2016, pp. 1- 6.

[25] J. d. A. Fernandes, L. N. Matos and M. G. d. S. Aragão, "Geometrical Approaches for Facial Expression Recognition Using Support Vector Machines," 2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), Sao Paulo, Brazil, 2016, pp. 347-354.

[26] L. A. Jeni, D. Takacs and A. Lorincz, "High quality facial expression recognition in video streams using shape related information only," 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Barcelona, 2011, pp. 2168-2174.

[27] S. Jain, Changbo Hu and J. K. Aggarwal, "Facial expression recognition with temporal modeling of shapes," 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Barcelona, 2011, pp. 1642-1649.

[28] H. Li; H. Zou; H. Hu, "Modified Hidden Factor Analysis for Cross-Age Face Recognition," in IEEE Signal Processing Letters , vol.PP, no.99, pp.1-1.

[29] S. An and Q. Ruan, "3D facial expression recognition algorithm using local threshold binary pattern and histogram of oriented gradient," 2016 IEEE 13th International Conference on Signal Processing (ICSP), Chengdu, 2016, pp. 265-270.

[30] J. Kumari, R. Rajesh and A. Kumar, "Fusion of features for the effective facial expression recognition," 2016 International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, 2016, pp. 0457-0461.

[31] T. W. Shen et al., "Facial expression recognition using depth map estimation of Light Field Camera," 2016 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), Hong Kong, 2016, pp. 1-4.

[32] J. S. B. and S. M., "Multi-posture Human Detection Based on Hybrid HOG-BO Feature," 2015 Fifth International Conference on Advances in Computing and Communications(ICACC),Kochi,2015,pp.37-40. doi: 10.1109/ICACC.2015.99.

[33] Yunsheng Jiang and Jinwen Ma, "Combination features and models for human detection," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston,MA,2015,pp240-248. doi: 10.1109/CVPR.2015.7298620.

[34] CS231n Convolutional Neural Networks for Visual Recognition, http://cs231n.github.io/convolutional-networks/#architectures.

[35] Alex Krizhevsky,Ilya Sutskever,Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks".

[36] Shima Alizadeh, Azar Fazel,"Convolutional Neural Networks for Facial Expression Recognition".

[37] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar and I. Matthews, “The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression”, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, San Francisco, CA, June 13-18, pp. 94-101, 2010.