

Facial Expression Recognition using CNN

A

Project Report

Submitted for the partial fulfilment of

B.Tech degree

in

COMPUTER SCIENCE & ENGINEERING

By

Eshan Singh(1805210019)

Neelansh Pandey(1805210028)

Ritesh Chaurasia(1805210043)

Under the supervision of

Prof. Y N Singh



Department of Computer Science and Engineering

Institute of Engineering and Technology

Dr A.P.J Abdul Kalam Technical University, Lucknow, Uttar Pradesh

June 2022

Contents

DECLARATION	3
CERTIFICATE	4
ACKNOWLEDGEMENT	5
ABSTRACT	6
LIST OF FIGURES.	7
LIST OF TABLES.	8
1. INTRODUCTION	9
2. LITERATURE REVIEW.....	10
3. METHODOLOGY.....	12
4. EXPERIMENTAL RESULTS.....	17
5. CONCLUSIONS.....	24
REFERENCES	
ANNEXURE	

Declaration

We hereby declare that this submission is our own work and that, to the best of our belief and knowledge, it contains no material previously published or written by another person or material which to a substantial error has been accepted for the award of any degree or diploma of university or another institute of higher learning, except where the acknowledgement has been made in the text. We have not submitted the project to any other institute for other degree requirements.

Submitted by:-

Date:02/06/2022

(1) Name: Eshan Singh

Roll.No.:1805210019

Branch: Computer Science & Engineering

Signature:

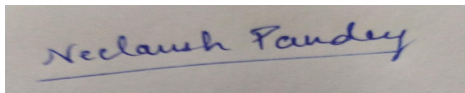


(2) Name: Neelansh Pandey

Roll.No.:1805210028

Branch: Computer Science & Engineering

Signature:

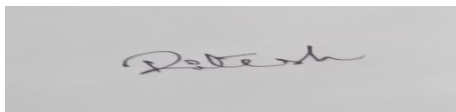


(3) Name: Ritesh Chaurasia

Roll.No.:1805210043

Branch: Computer Science & Engineering

Signature:



Certificate

This is to certify that the project report entitled “Facial Expression Recognition using CNN” presented by Eshan Singh, Neelansh Pandey and Ritesh Chaurasia in the partial fulfilment for the award of Bachelor of Technology in Computer Science and Engineering, is a record of work carried out by them under my supervision and guidance at the Department of Computer Science and Engineering at Institute of Engineering and Technology, Lucknow. It is also certified that this project has not been submitted to any other institute for the award of any other degrees to the best of my knowledge.

Prof. Y N Singh
Department of Computer Science and Engineering
Institute of Engineering & Technology, Lucknow

Acknowledgement

It is a great pleasure to have the opportunity to extend our heartfelt gratitude to everyone who helped us throughout the course of this project. We are profoundly grateful to our supervisor Prof. Y N Singh for his expert guidance, continuous encouragement and ever willingness to spare time from his otherwise busy schedule for the project's progress reviews. His continuous inspiration has made us complete this project and achieve its target.

We would also like to express our deepest appreciation to Prof. Divakar Singh Yadav, Head of Department, Institute of Engineering and Technology(IET) Lucknow, Department of Computer Science and Engineering, for his constant motivation, support and for providing us with a suitable working environment.

We would also like to extend our sincere regards to our co-supervisor Mrs Deepali Awasthi and all the faculty members for their support and encouragement. At last, our special thanks go to all staff members of the CS department who directly and indirectly extended their hands in making this project works a success

Abstract

Face recognition is of great importance to real-world applications such as video surveillance, human-machine interaction and security systems. As compared to traditional machine learning approaches, deep learning-based methods have shown better performances in terms of accuracy and speed of processing in image recognition. This report proposes a Modified Convolutional Neural Network (CNN) architecture by adding two normalization operations to two of the layers. The normalization operation which is batch normalization provided accelerates the network. CNN architecture is employed to extract distinctive face features and a Softmax classifier is used to classify faces in the fully connected layer of CNN. In the experiment part, Georgia Tech Database shows that the proposed approach has improved the face recognition performance with better recognition results.

List of Figures

Figure 1.1: A model of CNN.....	12
Figure 1.2: Max Pooling.....	14
Figure 3 (a): Training Phase.....	17
Figure 3 (b): Testing Phase.....	17
Figure 3.2: Architecture of CNN.....	18
Figure 4.1: Cost Function in Training and Validation.....	20
Figure 4.2: Error Rate in Training and Validation.....	21

List of Tables

Table 4.1: Confusion matrix for facial expression recognition.....	22
Table 4.2: Normalized confusion matrix for facial expression recognition.....	23
Table 4.3: Precision, Recall and F1-Score.....	24

Chapter 1

Introduction

A Facial expression is the visible manifestation of the affective state, cognitive activity, intention, personality and psychopathology of a person and plays a communicative role in interpersonal relations. Human facial expressions can be easily classified into 7 basic emotions: happy, sad, surprise, fear, anger, disgust, and neutral. Our facial emotions are expressed through the activation of specific sets of facial muscles. These sometimes subtle, yet complex, signals in an expression often contain an abundant amount of information about our state of mind. Automatic recognition of facial expressions can be an important component of natural human-machine interfaces; it may also be used in behavioural science and in clinical practice. It has been studied for a long period of time and obtained progress in recent decades. Though much progress has been made, recognizing facial expressions with a high accuracy remains to be difficult due to the complexity and variety of facial expressions [1].

On a day-to-day basis, humans commonly recognize emotions by characteristic features, displayed as a part of a facial expression. For instance, happiness is undeniably associated with a smile or an upward movement of the corners of the lips. Similarly, other emotions are characterized by other deformations typical to a particular expression. Research into automatic recognition of facial expressions addresses the problems surrounding the representation and categorization of static or dynamic characteristics of these deformations of face pigmentation [2].

In machine learning, a convolutional neural network (CNN, or ConvNet) is a type of feedforward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli in a restricted region of space known as the receptive field. The receptive fields of different neurons partially overlap such that they tile the visual field. The response of an individual neuron to stimuli within its receptive field can be approximated mathematically by a convolution operation.[3] Convolutional networks were inspired by biological processes[4] and are variations of multilayer perceptron designed to use minimal amounts of preprocessing.[5]

Chapter 2

Literature Review

Two different approaches are used for facial expression recognition, both of which include two different methodologies, exist [6]. Dividing the face into separate action units or keeping it as a whole for further processing appears to be the first and the primary distinction between the main approaches. In both of these approaches, two different methodologies, namely the ‘Geometric based’ and the ‘Appearance-based’ parameterizations, can be used. Making use of the whole frontal face image and processing it in order to end up with the classifications of 6 universal facial expression prototypes: disgust, fear, joy, surprise, sadness and anger; outlines the first approach. Here, it is assumed that each of the above-mentioned emotions has characteristic expressions on the face and that’s why recognition of them is necessary and sufficient. Instead of using the face images as a whole, dividing them into some sub-sections for further processing forms the main idea of the second approach for facial expression analysis. As the expression is more related to subtle changes of some discrete features such as eyes, eyebrows and lip corners; these fine-grained changes are used for analyzing automated recognition.

There are two main methods that are used in both of the above-explained approaches. Geometric Based Parameterization is an old way which consists of tracking and processing the motions of some spots on image sequences, firstly presented by Suwa et al to recognize facial expressions [7]. Cohn and Kanade, later on, tried geometrical modelling and tracking of facial features by claiming that each AU is presented with a specific set of facial muscles [8]. The disadvantages of this method are the contours of these features and components have to be adjusted manually in this frame, the problems of robustness and difficulties come out in cases of pose and illumination changes while the tracking is applied on images, as actions & expressions tend to change both in morphological and in dynamical senses, it becomes hard to estimate general parameters for movement and displacement. Therefore, ending up with robust decisions for facial actions under these varying conditions becomes to be difficult.

Rather than tracking spatial points and using positioning and movement parameters that vary within time, colour (pixel) information of related regions of the face is processed in Appearance Based Parameterizations; in order to obtain the parameters that are going to form the feature vectors. Different features such as Gabor, and Haar wavelet coefficients, together with feature extraction and selection methods such as PCA, LDA, and Adaboost are used within this framework.

For classification problems, algorithms like Machine learning, Neural networks, Support Vector Machine, Deep learning, and Naive Bayes are used. Raghuvanshi A. et al have built a Facial expression recognition system upon recent research to classify images of human faces into discrete emotion categories using convolutional neural networks [9]. Alizadeh, Shima, and Azar Fazel have developed a Facial Expression Recognition system using Convolutional Neural Networks based on the Torch model [10].

Chapter 3 Methodology

The facial expression recognition system is implemented using a convolutional neural network. During training, the system received training data comprising grayscale images of faces with their respective expression label and learns a set of weights for the network. The training step took as input of an image with a face. Thereafter, an intensity normalization is applied to the image. The normalized images are used to train the Convolutional Network. To ensure that the training performance is not affected by the order of presentation of the examples, the validation dataset is used to choose the final best set of weights out of a set of training performed with samples presented in different orders. The output of the training step is a set of weights that achieve the best result with the training data. During the test, the system received a grayscale image of a face from the test dataset, and output the predicted expression by using the final network weights learned during training. Its output is a single number that represents one of the seven basic expressions.

LeNet is one of the very first convolutional neural networks which helped propel the field of Deep Learning. This pioneering work by Yann LeCun was named LeNet5 and was used mainly for character recognition tasks such as reading zip codes, digits, etc. The basic architecture of LeNet can be shown below [5]:

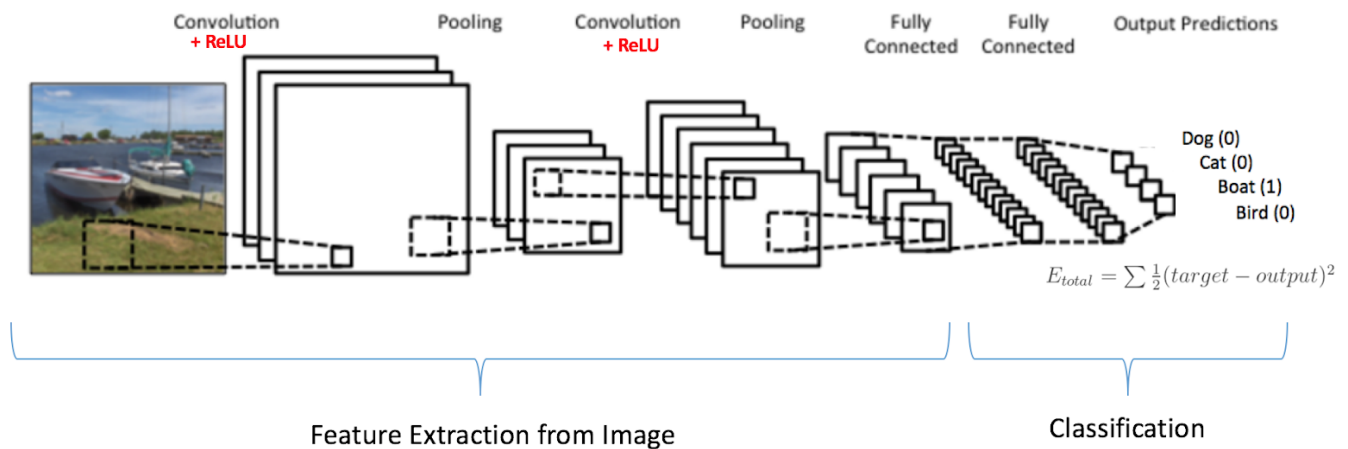


Figure 1.1: A model of CNN[5]

There are four main operations in the Convolution Neural Network shown in Figure 2.2 above:

1. Convolution:

The primary purpose of Convolution in the case of a CNN is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. The convolution 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 the first layer of a CNN might have a size of $3 \times 5 \times 5$ (i.e. images have depth 3 i.e. the colour channels, 5 pixels width and height). During the forward pass, each filter is convolved across the width and height of the input volume and computes dot products between the entries of the filter and the input at any position. As the filter convolves over the width and height of the input volume it produces 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 colour on the first layer, or eventually entire honeycomb or wheel-like patterns on higher layers of the network. Now, there will be an entire set of filters in each convolution layer (e.g. 20 filters), and each of them will produce a separate 2-dimensional activation map.

The 2-dimensional convolution between image A and Filter B can be given as $C(i,j) = \sum_{n=0}^{Na-1} \sum_{m=0}^{Ma-1} A(m, n) * B(i - m, j - n)$ (2.1) where the size of A is $(Ma \times Na)$, size of B is $(Mb \times Nb)$, $0 \leq i < Ma + Mb - 1 \wedge 0 \leq j < Na + Nb - 1$. A filter convolves with the input image to produce a feature map. The convolution of another filter over the same image gives a different feature map. The convolution operation captures the local dependencies in the original image. A CNN learns the values of these filters on its own during the training process (although parameters such as the number of filters, filter size, architecture of the network etc. still needed to specify before the training process). The more filters, the more image features get extracted and the better network becomes at recognizing patterns in unseen images. The size of the Feature Map (Convolved Feature) is controlled by three parameters: Depth: Depth corresponds to the number of filters we use for the convolution operation. Stride: Stride is the size of the filter, if the size of the filter is 5×5 then stride is 5. Zero-padding: Sometimes, it is convenient to pad the input matrix with zeros around the border, so that the filter can be applied to bordering elements of the input image matrix. Using zero padding size of the feature map can be controlled.

2. Rectified Linear Unit:

An additional operation called ReLU has been used after every Convolution operation. A Rectified Linear Unit (ReLU) is a cell of a neural network which uses the following activation function to calculate its output given x : $R(x) = \text{Max}(0, x)$ (2.2) Using these cells is more efficient than sigmoid and still forwards more information compared to binary units. When initializing the weights uniformly, half of the weights are negative. This helps create a sparse feature representation. Another positive aspect is the relatively cheap computation. No exponential function has to be calculated. This function also prevents the vanishing gradient error, since the gradients are linear functions or zero but in no case nonlinear functions.

3. Pooling (sub-sampling)

Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc. In the case of Max Pooling, a spatial neighbourhood (for example, a 2×2 window) is defined and the largest element is taken from the rectified feature map within that window. In the case of average pooling, the average or sum of all elements in that window is taken. In practice, Max Pooling has been shown to work better. Max Pooling reduces the input by applying the maximum function over the input x_i . Let m be the size of the filter, then the output calculates as follows:

$$M(x_i) = \max \{x_{i+k,l} \mid |k| \leq m/2, |l| \leq m/2, k, l \in \mathbb{N}\}$$

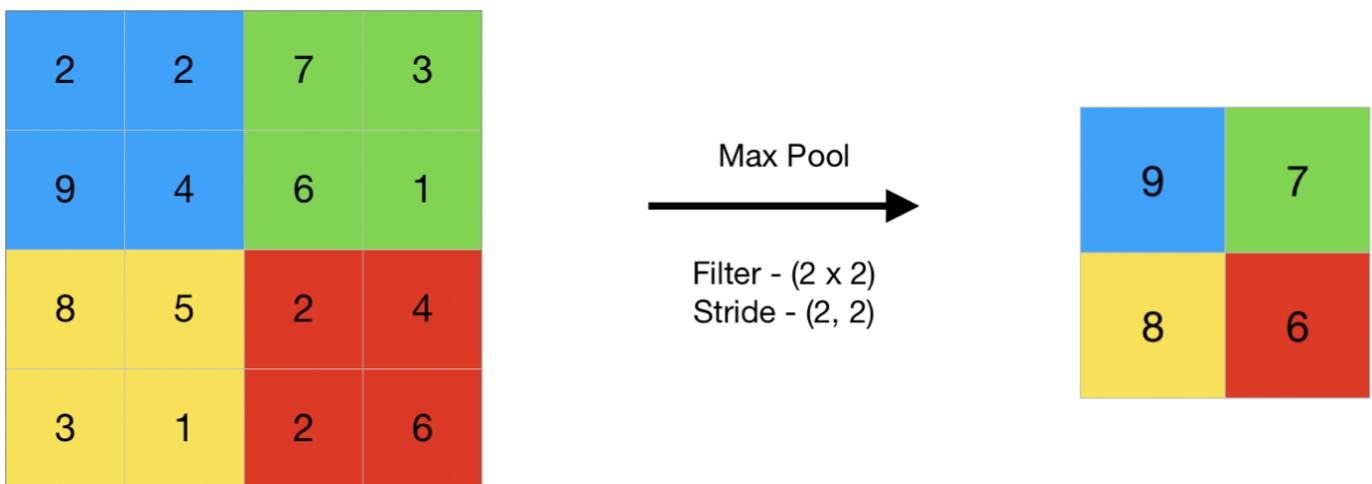


Figure 1.2: Max Pooling

The function of Pooling is to progressively reduce the spatial size of the input representation. In particular, pooling

1. Makes the input representations (feature dimension) smaller and more manageable.
2. Reduces the number of parameters and computations in the network, therefore, controlling over-fitting.
3. Makes the network invariant to small transformations, distortions and translations in the input image (a small distortion in the input will not change the output of Pooling).
4. Helps us arrive at an almost scale invariant representation. This is very powerful since objects can be detected in an image no matter where they are located.

4. Classification (Multilayer Perceptron):

The Fully Connected layer is a traditional Multi-Layer Perceptron that uses a softmax activation function in the output layer. The term “Fully Connected” implies that every neurone in the previous layer is connected to every neurone in the next layer. The output from the convolutional and pooling layers represents high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.

Softmax is used for the activation function. It treats the outputs as scores for each class. In the Softmax, the function mapping stayed unchanged and these scores are interpreted as the unnormalized log probabilities for each class. Softmax is calculated as:

$$f(z)_j = \frac{\exp(z_j)}{\sum_{k=1}^K \exp(z_k)}$$

where j is the index for image and K is the number of total facial expression class.

Apart from classification, adding a fully-connected layer is also a (usually) cheap way of learning non-linear combinations of these features. Most of the features from convolutional and pooling layers may be good for the classification task, but combinations of those features might be even better.

The sum of output probabilities from the Fully Connected Layer is 1. This is ensured by using the activation function in the output layer of the Fully Connected Layer. The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sums to one.

Chapter 4

Experimental Results

4.1 Dataset

The dataset from a Kaggle Facial Expression Recognition Challenge (FER2013) is used for the training and testing. It comprises pre-cropped, 48-by-48-pixel grayscale images of faces each labeled with one of the 7 emotion classes: anger, disgust, fear, happiness, sadness, surprise, and neutral. Dataset has training set of 35887 facial images with facial expression labels.. The dataset has class imbalance issue, since some classes have large number of examples while some has few. The dataset is balanced using oversampling, by increasing numbers in minority classes. The balanced dataset contains 40263 images, from which 29263 images are used for training, 6000 images are used for testing, and 5000 images are used for validation.

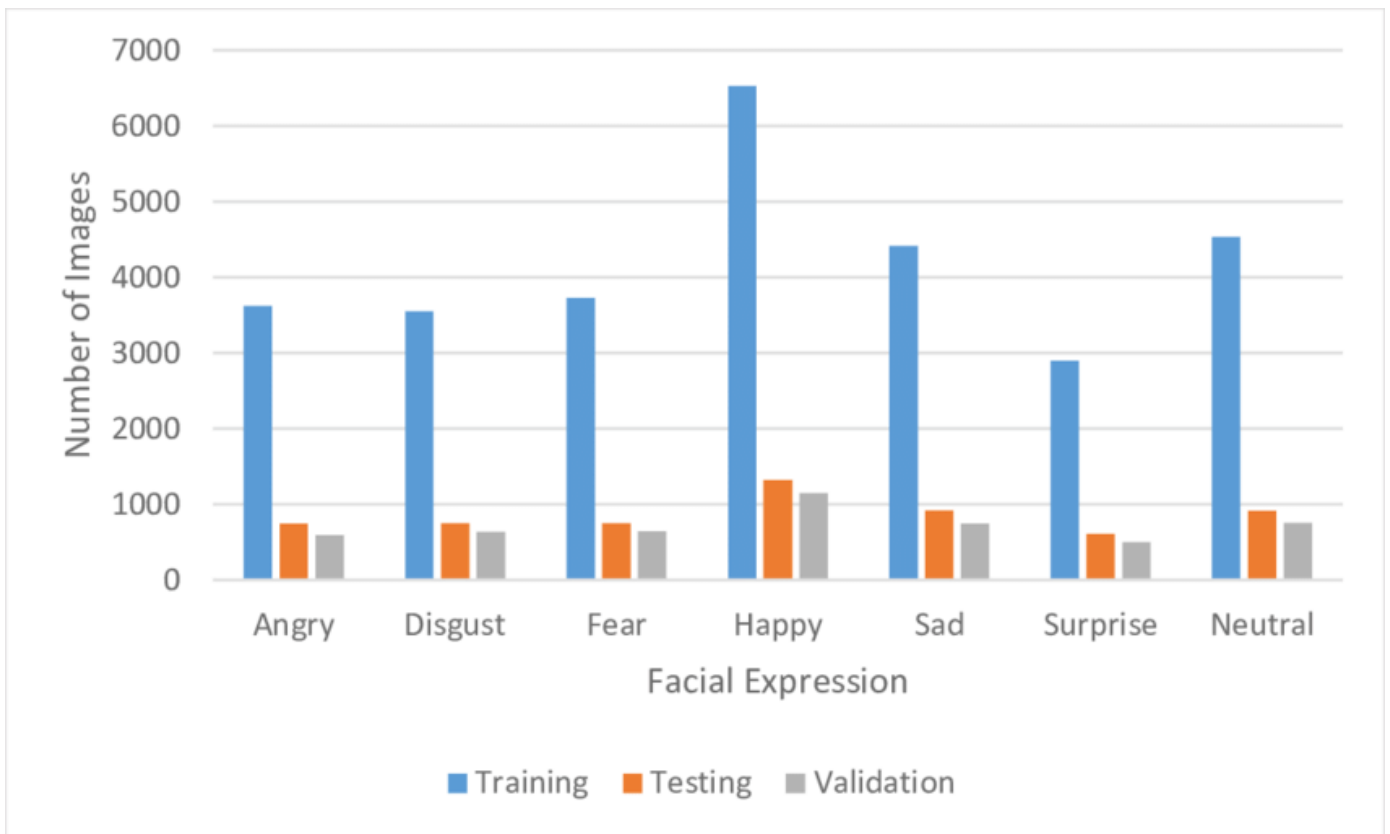


Figure 3.1: Training, Testing and Visualisation dataset

4.2 Architecture of CNN

A typical architecture of a convolutional neural network contains an input layer, some convolutional layers, some fully-connected layers, and an output layer. CNN is designed with some modification on LeNet Architecture [10]. It has 6 layers without considering input and output. The architecture of the Convolution Neural Network used in the project is shown in the following figure.

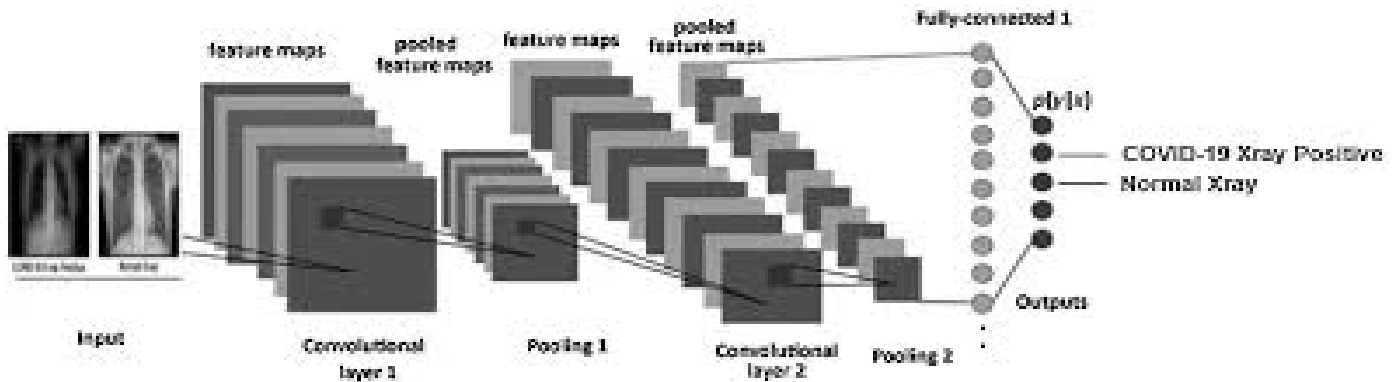


Figure. 3.2: Architecture of CNN(modified from LeNet architecture)

1. Input Layer:

The input layer has pre-determined, fixed dimensions, so the image must be pre-processed before it can be fed into the layer. Normalized gray scale images of size 48 X 48 pixels from Kaggle dataset are used for training, validation and testing. For testing propose laptop webcam images are also used, in which face is detected and cropped using OpenCV Haar Cascade Classifier and normalized.

2. Convolution and Pooling (ConvPool) Layers:

Convolution and pooling is done based on batch processing. Each batch has N images and CNN filter weights are updated on those batches. Each convolution layer takes image batch input of four dimension $N \times \text{Color-Channel} \times \text{width} \times \text{height}$. Feature map or filter for convolution are also four dimensional (Number of feature maps in, number of feature maps out, filter width, filter height). In each convolution layer, four dimensional convolution is calculated between image batch and feature maps. After convolution only parameter that change is image width and height. New image width = old image width – filter width + 1
New image height = old image height – filter height + 1 After each convolution layer downsampling / subsampling is done for dimensionality reduction. This process is called Pooling. Max pooling and Average Pooling are two famous pooling method. In this project max pooling is done after convolution. Pool size of (2x2) is taken, which splits the image into grid of blocks each of size 2x2 and takes maximum

of 4 pixels. After pooling only height and width are affected. Two convolution layer and pooling layer are used in the architecture. At first convolution layer size of input image batch is $N \times 1 \times 48 \times 48$. Here, size of image batch is N , number of color channel is 1 and both image height and width are 48 pixel. Convolution with feature map of $1 \times 20 \times 5 \times 5$ results image batch is of size $N \times 20 \times 44 \times 44$. After convolution pooling is done with pool size of 2×2 , which results image batch of size $N \times 20 \times 22 \times 22$. This is followed by second convolution layer with feature map of $20 \times 20 \times 5 \times 5$, which results image batch of size $N \times 20 \times 18 \times 18$. This is followed by pooling layer with pool size 2×2 , which results image batch of size $N \times 20 \times 9 \times 9$.

3. Fully Connected Layer

This layer is inspired by the way neurons transmit signals through the brain. It takes a large number of input features and transforms features through layers connected with trainable weights. Two hidden layers of sizes 500 and 300 units are used in the fully-connected layer. The weights of these layers are trained by forwarding propagation of training data and then backward propagation of its errors. Backpropagation starts from evaluating the difference between prediction and true value and back calculates the weight adjustment needed to every layer before. We can control the training speed and the complexity of the architecture by tuning the hyper-parameters, such as learning rate and network density. Hyper-parameters for this layer include learning rate, momentum, regularization parameter, and decay. The output from the second pooling layer is of size $N \times 20 \times 9 \times 9$ and input of first hidden layer of fully-connected layer is of size $N \times 500$. So, output of pooling layer is flattened to $N \times 1620$ size and fed to first hidden layer. Output from first hidden layer is fed to second hidden layer. Second hidden layer is of size $N \times 300$ and its output is fed to output layer of size equal to number of facial expression classes.

4. Output Layer

Output from the second hidden layer is connected to output layer having seven distinct classes. Using Softmax activation function, the output is obtained using the probabilities for each of the seven class. The class with the highest probability is the predicted class.

4. Results and Analysis

CNN architecture for facial expression recognition as mentioned above was implemented in Python. Along with Python programming language, Numpy, Theano and CUDA libraries were used. Training image batch size was taken as 30, while filter map is of size $20 \times 5 \times 5$ for both convolution layer. Validation set was used to validate the training process. In the last batch of every epoch in validation cost, validation error, training

cost, and training error are calculated. Input parameters for training are the image set and corresponding output labels. The training process updated the weights of feature maps and hidden layers based on hyper-parameters such as learning rate, momentum, regularization and decay. In this system, batch-wise learning rate was used as $10e-5$, momentum as 0.99, regularization as $10e-7$ and decay as 0.99999. The comparison of validation cost, validation error, training cost, and training error are shown in the figures below.

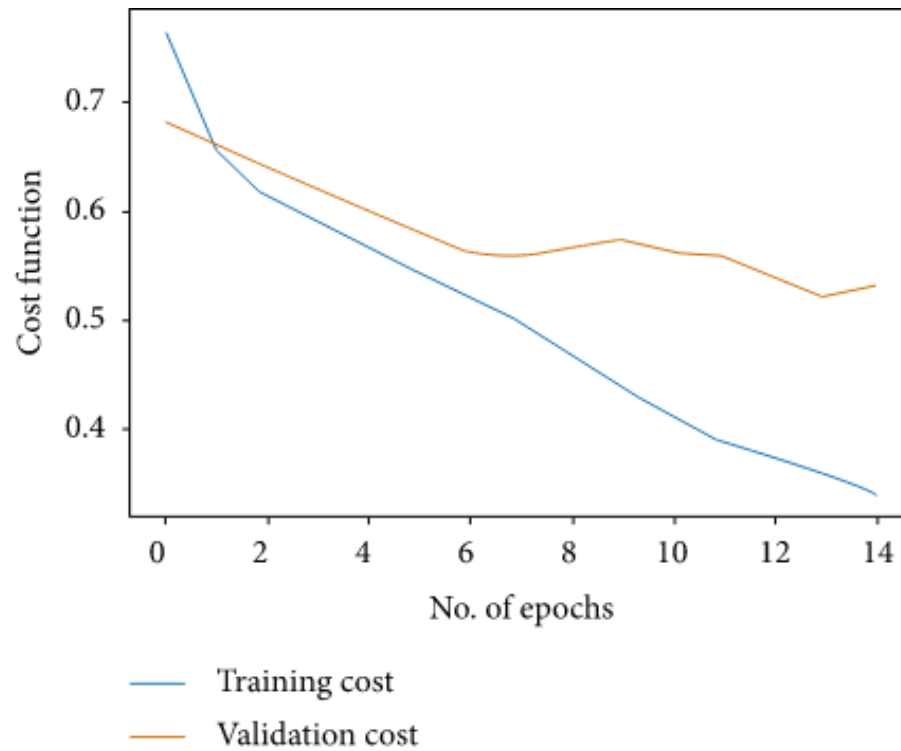


Figure 4.1: Cost function in training and validation

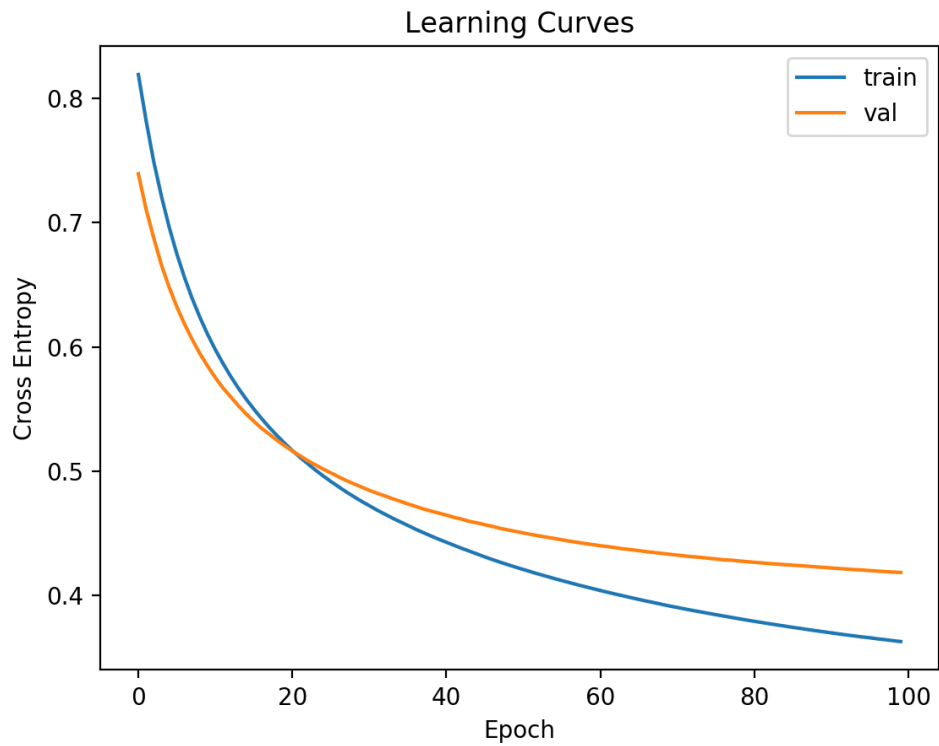


Figure 4.2: Error rate in training and validation

Above graph shows that the in 50 epochs training cost and error rate is reduced 0.35 and 0.1 respectively while validation cost and error is 1.25 and 0.45 respectively.

The testing of the model is carried out using 6000 images. The classifier provided 56.77 % accuracy. The confusion matrix for seven facial expression classes is shown below:

Predicted Actual	Anger	Disgust	Fear	Happy	Sad	Surprise	Neutral
Anger	0.421622	0.021769	0.118774	0.069099	0.109729	0.043339	0.11193
Disgust	0.047619	0.95238	0	0	0	0	0
Fear	0.166216	0.009524	0.383142	0.053571	0.113122	0.107544	0.123548
Happy	0.114865	0.006803	0.094508	0.688665	0.090498	0.046549	0.135164
Sad	0.185135	0.008163	0.155811	0.069099	0.382353	0.043339	0.174234
Surprise	0.036486	0.005442	0.095785	0.034938	0.027149	0.645265	0.048574
Neutral	0.154054	0.005442	0.11622	0.090062	0.13914	0.051364	0.493136

Above tables showed that this model highest accuracy for disgust emotion with 95.23 %, followed by happy with 68.86 %, surprise with 64.52 %, neutral with 49.31 %, anger with 42.16 %, fear with 38.31 % and lowest accuracy for sad emotion as 38.23 %.

Table 4.1: Confusion matrix for facial expression recognition

The normalized confusion matrix is shown below:

Predicted Actual	Anger	Disgust	Fear	Happy	Sad	Surprise	Neutral
Anger	0.421622	0.021769	0.118774	0.069099	0.109729	0.043339	0.11193
Disgust	0.047619	0.95238	0	0	0	0	0
Fear	0.166216	0.009524	0.383142	0.053571	0.113122	0.107544	0.123548
Happy	0.114865	0.006803	0.094508	0.688665	0.090498	0.046549	0.135164
Sad	0.185135	0.008163	0.155811	0.069099	0.382353	0.043339	0.174234
Surprise	0.036486	0.005442	0.095785	0.034938	0.027149	0.645265	0.048574
Neutral	0.154054	0.005442	0.11622	0.090062	0.13914	0.051364	0.493136

Above tables showed that this model highest accuracy for disgust emotion with 95.23 %, followed by happy with 68.86 %, surprise with 64.52 %, neutral with 49.31 %, anger with 42.16 %, fear with 38.31 % and lowest accuracy for sad emotion as 38.23 %.

Table 4.2: Normalized confusion matrix for facial expression recognition

The precision, recall and F1-score of each facial expression class is shown below:

Class	Precision	Recall	F-measure
Anger	0.47	0.74	0.57
Disgust	0.96	0.74	0.84
Fear	0.97	0.76	0.85
Joy	0.60	0.86	0.71
Sadness	0.75	0.56	0.64
Surprise	0.94	0.73	0.82
Weighted Average	0.79	0.73	0.74

Table 4.3: Precision, Recall and F1-Score

The overall precision and recall are 0.57 and 0.57 respectively. The model performs really well on classifying positive emotions resulting in relatively high precision scores for happy and surprised. Disgust has the highest precision and recall as 0.95 and 0.99 as images in this class were oversampled to address the class imbalance. Happy has a precision of 0.68 and recall of 0.69 which could be explained by having the most examples (6500) in the training set. Interestingly, 16 surprise has a precision of 0.69 and a recall of 0.65 having the least examples in the training set. There must be very strong signals in the surprise expressions.

Model performance seems weaker across negative emotions on average. In particular, the emotion sad has a low precision of only 0.44 and a recall of 0.38. The model frequently misclassified anger, fear and neutral as sad. In addition, it is most confused when predicting sad and neutral faces because these two emotions are probably the least expressive (excluding crying faces).

The overall F1 score is also 0.57. F1-score is highest for disgust due to oversampling of images. Happy and surprise have higher F1-score of 0.69 and 0.67 respectively. Fear has least F1 score of 0.39 and sad, anger and neutral also have low F1-score.

CNN Classifier is then used to classify images taken from the webcam on Laptop. The face is detected in webcam frames using the Haar cascade classifier from OpenCV. Then detected face is cropped and normalized and fed to CNN Classifier. Some classification results using a webcam are listed in Appendix A.

Chapter 5

Conclusion

5. 1 Conclusion

In this project, a LeNet architecture based six layers convolution neural network is implemented to classify human facial expressions i.e. happy, sad, surprise, fear, anger, disgust, and neutral. The system has been evaluated using Accuracy, Precision, Recall and F1-score. The classifier achieved an accuracy of 56.77 %, the precision of 0.57, recall of 0.57 and F1-score of 0.57. The overall precision and recall are 0.57 and 0.57 respectively. The model performs really well on classifying positive emotions resulting in relatively high precision scores for happy and surprised. Disgust has highest precision and recall as 0.95 and 0.99 as images in this class were oversampled to address class imbalance. Happy has a precision of 0.68 and recall of 0.69 which could be explained by having the most examples (6500) in the training set. Interestingly, 16 surprise has a precision of 0.69 and recall of 0.65 having the least examples in the training set. Model performance seems weaker across negative emotions on average. In particular, the emotion sad has a low precision of only 0.44 and a recall 0.38. The model frequently misclassified angry, fear and neutral as sad. In addition, it is most confused when predicting sad and neutral faces because these two emotions are probably the least expressive (excluding crying faces). The overall F1 score is also 0.57. F1-score is highest for disgust due to oversampling of images. Happy and surprise have higher F1-score of 0.69 and 0.67 respectively. Fear has least F1 score of 0.39 and sad, anger and neutral also have low F1-score.

References

- [1] Shan, C., Gong, S., & McOwan, P. W. (2005, September). Robust facial expression recognition using local binary patterns. In Image Processing, 2005. ICIIP 2005. IEEE International Conference on (Vol. 2, pp. II-370). IEEE.
- [2] Chibelushi, C. C., & Bourel, F. (2003). Facial expression recognition: A brief tutorial overview. CVonline: On-Line Compendium of Computer Vision, 9.
- [3] "Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation". DeepLearning 0.1. LISA Lab.
- [4] Matusugu, Masakazu; Katsuhiko Mori; Yusuke Mitari; Yuji Kaneda (2003). "Subject independent facial expression recognition with robust face detection using a convolutional neural network" (PDF). Neural Networks. 16 (5): 555–559. doi:10.1016/S0893-6080(03)00115-1.
- [5] LeCun, Yann. "LeNet-5, convolutional neural networks".
- [6] C. Zor, "Facial expression recognition," Master's thesis, University of Surrey, Guildford, 2008.
- [7] Suwa, M.; Sugie N. and Fujimora K. A Preliminary Note on Pattern Recognition of Human Emotional Expression, Proc. International Joint Conf, Pattern Recognition, pages 408-410, 1978
- [8] Recognizing action units for facial expression analysis YI Tian, T Kanade, JF Cohn IEEE Transactions on pattern analysis and machine intelligence 23 (2), 97- 115
- [9] Raghuvanshi, Arushi, and Vivek Choksi. "Facial Expression Recognition with Convolutional Neural Networks." Stanford University, 2016
- [10] Alizadeh, Shima, and Azar Fazel. "Convolutional Neural Networks for Facial Expression Recognition." Stanford University, 2016

- [11] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.
- [12] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).