International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2018): 7.426

Recognizing Emotions Expressed by Body Pose using Different Architectures

Harmeet Thukran[1], Surya Pratap Singh[2]

[1, 2]The North Cap University, Gurgaon, Haryana, India

Abstract: ***The project entitled to "body pose recognition" aims at detecting a person's emotional state by considering their body postures. The project is on Image recognition concept and uses different architectures of Convolutional Neural Networks, it then compares the efficiency among different architectures used. The maximum efficiency achieved in this project work is about 72%.***

1. Introduction

Over the last decades, most studies have concentrated on emotional signals in facial expressions. Recently, researchers have also turned to emotional body language, i.e. the expression of emotions through human body pose and/or body motion. Human emotions are strongly connected with bodily states. Body signals allow us to communicate through non-verbal cues emotional indicators providing emotional abilities to machines could enhance productivity and quality of life, considering the possible interaction that the machine could generate as partner and collaborator.

One of the motivations for the future interrelation with machines would be the happiness of human beings. This motivation would stimulate the collaboration between machines and humans by exchanging information.

The future human-machine interaction could be influenced by an emotional feedback that machines can capture from the environment of humans and evolve with them. We shall also expect improvements in the communicative behavior in machines, which is a very urgent task, and then people could more easily be able to accept and integrate them.

2. Problem definition and Objectives

Recently, it's been observed that most of the research has been concentrated towards the field of emotion recognition using facial gestures, Body language also possesses multiple features that can be used to reliably identify the emotions and can be very helpful in cutting-edge technologies involving artificial intelligence and robotics.

In this project, we focus on analysing and categorizing different emotions depending upon complete body pose images using Convolution neural networks or ConvNets applying deep learning to classify the input into six basic emotions as defined by Ekman namely happy, sad, anger, disgust, surprise, fear.

The project also aims to identify an efficient and robust CNN Architecture that can accurately predict emotional coefficient based on the input body pose.

3. Proposed Approach

The project not only aims to find a good CNN architecture to c classify the input data set into the 7 given classes (happy, sad, fear, anger, disgust, surprise, neutral) but also to compare between standard CNN implementation and performing low-level feature extraction of the body poses prior to feeding them into the neural network.

Therefore our project will be divided into two parts, the first part in which we will train the regular dataset of images of body poses on the CNN and the second part being the training of the dataset on various architectures also trying different low-level feature extraction methods to improve on the CNN models.

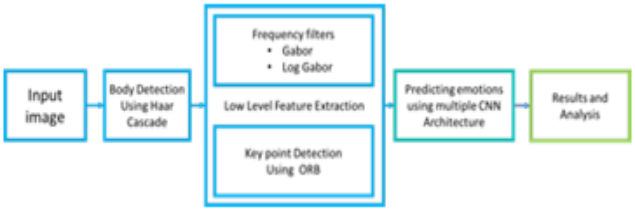


Figure 1.1: Flowchart Of the proposed approach

3.1 Haar cascade (Full Body Classifier)

Haar Cascades is a Machine learning approach to object detection in an image. Proposed by Paul Viola and Michael Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001, Haar Cascades requires training multiple positive and negative images, positive images being the images of the object of interest present in the image and the negative images being the ones without the object of interest.

For every positive image, we need to extract features from it so that we can detect objects, for this purpose we require Haar-like features that resemble the convolution kernels. Each of the kernels helps to extract certain features like edge features, and line features are calculated using different kernels.

Haar Cascades are **Scale Invariant**. This object detection process is done in stages and if in any earlier stage the image is rejected then further detection process is not done on it and it is concluded that the image is a negative image. This is the concept of **Cascade of Classifiers.**

Figure 1.2: (1) shows the Haar filters used as Cascades, (2) Shows the body detec output after applying Haar cascades on dataset

3.2 Low-Level Feature Extraction

Preprocessing of Dataset involves feature extraction and dimensionality reduction. It can be useful for a neural network as it prevents the use of unnecessary features present in a dataset. It also improves the performance by suppressing the unwanted features.

3.2.1 Gabor and Log-Gabor Filter

An image is viewed as formed by superimposing a series of sinusoidal waves of various frequencies oriented multi-dimensionally. Each pixel denotes the intensity of such a wave. The position of a pixel tells us the frequency and orientation.

Low-level feature extraction aims to use Gabor and Log-Gabor filters as many of the so- called band pass filters that allow you to cut the Fourier transform and isolate only required information in our case the Edges.

Low-level filters are orientation-sensitive filters, used for texture analysis. They are used in packs, one for each direction. A filter set with a given direction gives a strong response for locations having structures in this given direction.

A linear filter that is used for detection of edge. Gabor filters with dissimilar frequencies and with orientation in various directions have been used to localize and extract edges with higher-frequency components & with the relatively smooth background.

The Log-Gabor filter is capable of describing a signal in terms of the responses of the local frequency. Because it is an elementary technique for signal analysis. Log-Gabor filters are used since they have zero dc component for large bandwidth, and size distribution is often logarithmic, which is not the case with Gabor filters.

The benefit of having zero dc component is that the response from the filter doesn't depend on the average value of the signal. If we designed a filter kernel to match an edge, for feature detection. The response would vary with the average image level, if the filter had a dc component.

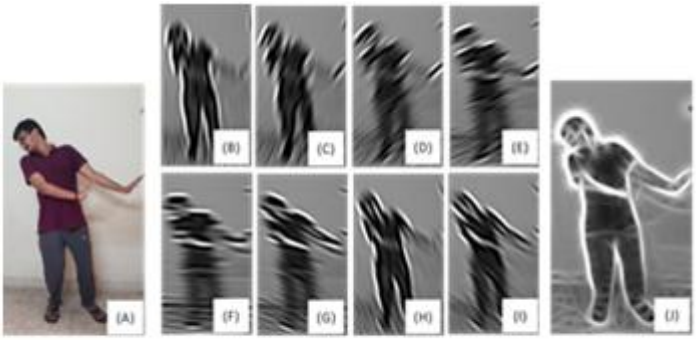


Fig 1.3 (A) input image from dataset to Gabor filter, images (B) to (I) are applying individual filter kernels with orientation angle difference of 22.5 degrees between 0 to 180 degrees, (J) is the output after accumulating the image from each filter kernel giving the edges of the input image

Fig 1.4 (A) Input image From the dataset to Log Gabor Filters, (B) Output image after applying Log Gabor Filter to the input image with default parameters

3.3. ORB Feature Extraction

(Oriented FAST and Rotated BRIEF) is a key point detection algorithm. It is a combination of FAST key point detector and BRIEF descriptor. First, it uses FAST algorithm to find key points, then finds top N points among them.

3.3.1 FAST Corner Detection

FAST corner detector is a computationally efficient algorithm. It is faster than many other feature extraction methods, such as difference of Gaussians used by the SIFT FAST corner detector uses a circle of 16 pixels to classify whether a candidate point is actually a corner. Each pixel in the circle is labeled number 1 to 16 clockwise. If a set of N contiguous pixels in the circle are of high intensity compared to the candidate pixel p (denoted by Ip) plus a threshold value t or all have less intensity than the candidate pixel p minus threshold value t, then the candidate pixel is classified as a corner. Either of the below conditions if met implies the pixel is a corner pixel:

Condition 1: A set of N contiguous pixels S, ∀ x ∈ S, the intensity of x (Ix)>Ip + threshold t
Condition 2: A set of N contiguous pixels S, ∀ x ∈ S, Ix <Ip – t

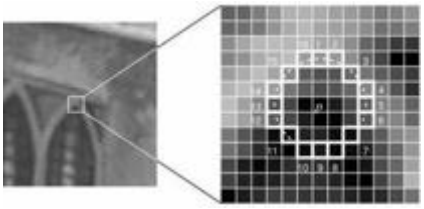
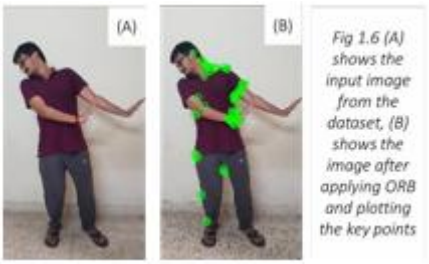


Figure 1.5: Illustrates Fast algorithm on pixel p

3.3.2 BRIEF (Binary Robust Independent Elementary Features)

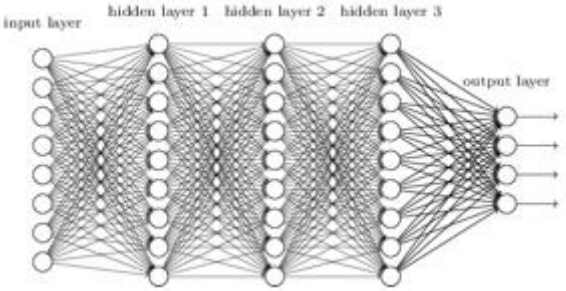
BRIEF is a general-purpose feature point descriptor that can be combined with arbitrary detectors. It is vigorous to respective classes of photometric and geometric image transformations.



3.4 Convolution Neural Network

Convolutional neural networks (CNNs) are widely used in the tasks of pattern and image recognition because of their superiority over other techniques. Convolutional Neural Networks are very much like the simple Neural Networks: that consists of neurons with learnable weights and biases. Each neuron gets some inputs, then performs a dot product operation and optionally follows with a non-linearity. From the raw image pixels on one end to the scores of classes at the other, the entire network expresses a single differential score function. And they still have a loss function (e.g. Softmax) on the last (fully-connected) layer and all the tricks learned for developing regular Neural Networks are still applicable.

The basic need of CNN aroused for image recognition problems as incase of images the no of parameters in input layer become large and in order to make the recognizing system efficient the number of hidden layers in the neural network are also large, due to which the effect of the weights of initially hidden layers is not much during backpropagation. This increases the number of iterations needed to adjust the weights in order to obtain good accuracy from the system, thereby increasing the computation power.



For example, an input image of size 300 x 300 pixels will require 90000 neurons in the input layer. Adjusting weights for such a large number of input neurons will require a large number of hidden layers and also good accuracy can't be guaranteed.

CNN take into consideration spatial information in an image. It extracts the important features and trains the system accordingly. Given below is the basic architecture overview of CNN:

3.4.1 Input Layer

This layer holds the raw pixel values of the image e.g., in this case, an image of width and height 48 and 48, and with the three color channels which are R, G, and B it would have dimensions 48 x 48 x3.

3.4.2 CONV Layer

In this layer, each neuron performs a dot product between their weight and their local receptive fields. This may result in volume such as [48x48x12] if we have decided to employ 12 filters.

3.4.3 RELU Layer

This layer performs an element-wise activation function, such as the max (0, x) which is used to do the thresholding at zero. This removes negative intensities while keeping the volume unchanged [48x48x12].

3.4.4 POOL layer

This layer will do a down sampling along the spatial dimensions (width, height), thus resulting in a volume as [24x24x12] in case if we are using a 2 x 2 pooling filter. Eg. A 2 x 2 max pool filter.

3.4.5 FC Layer

This fully-connected layer computes the class scores, resulting in a volume of size [1x1x7], where each number corresponds to a particular class score, among the 7 possible categories. As the name suggests, every single neuron in this layer will be connected to all the neurons in the preceding volume.

The filters in the CONV layer are simple edge detection filters at different angles (horizontal, vertical, +45 degree i.e. diagonals, etc.) as the edges carry the useful information in the image. We can multiple combinations of CONV, RELU, and POOL layer in between the Input and the FC layer to try out different architectures in order to obtain the best possible results.

The three basic ideas used in Convolutional Neural Networks are as Local receptive fields, pooling, and shared weights. They are described as:

Local Receptive Fields: In a convolution neural network each unit in a hidden layer is only connected to a small number of units in the previous layer. For instance, the very first hidden layer will only be attached to a small localized region of the input image. This region is known as the receptive field.

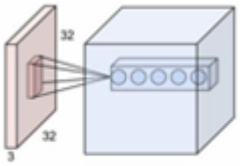


Fig 1.7 Local Receptive field of a Fiber

Shared Weights: When we take the dot product between a filter and a patch of an image, we are measuring the correlation between the filter and the patch. The larger is the value of the dot product, the more alike the patch and the filter look. Because we want to capture the same feature at different places in an image.

3.4.6 Softmax Activation Function

The Softmax function is also known as normalized exponential function. This Activation Function is a generalization of the logistic function that "squashes" a K-dimensional vector 'z' of arbitrary real values to a K-dimensional vector sigma(z) of real values in the range (0, 1) that add up to 1. The function is given by:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

The final layer of neural network uses Softmax function, which are applied to the problems of classification. Such networks are commonly trained under a log loss (or cross-entropy) arrangement, giving a non-linear variant of multinomial logistic regression.

3.4.7 Architectural Designs

In this project, we are reviewing the ways to scale up networks in a manner that aims at utilizing the added computation as efficiently as feasible by suitably factorized convolutions and aggressive regularization.

The frequent most form of a ConvNet architecture is as follows:
(A) Stack a few CONV-RELU layers,
(B) Conv-ReLU layer follows POOL layers and iterate this pattern unless the image has been merged spatially to a small size.
(C) Sometimes, it is usual to transition to fully-connected layers. The output is held by the last fully- connected layer, such as the class scores.

This architecture can be summarized in a small formula:

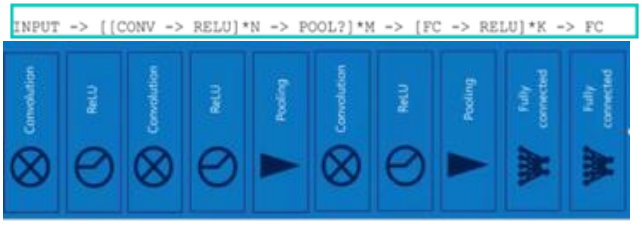


Fig 1.8 Graphical representation of a Layered CNN Architecture

CNN models can be designed in different ways. Here we are trying to build an architecture which can efficiently deal with the computational cost as well as training time.

By altering the convolution neural network model we can improve the performance of our network.

3.4.8 General Design Principles

In the project, while designing the architectures some design principles were kept in mind as mentioned in the paper "Rethinking the Inception Architecture for Computer Vision" by Google[14].

1) Pooling should not be drastic i.e. the representation size should decrease considerable from inputs to the outputs before arriving at the final representation. If pooling is drastic some important features can be lost during this process.
2) We should try to maintain a balance between the width and depth of the network. Width and depth of a network should be increased in parallel.
3) The filter size should increase slowly at each convolution layer. This allows the architecture to first find the bold features and then search for deeper features.
4) Instead of using one large filter we should use multiple small filters. This simulates larger filter i.e. more features while still keeping the parameter count low.

3.4.9 Different CNN Architectures

1. Conv (16) – Relu – Pool – FC

One Convolution Layer of 16 Filters: The first model comprises of **one convolution layer** with 16 filters and receptive field of 3x3 Pixels with stride 1, followed by a **2x2 Pooling Layer** and then the architecture is flattened and is connected to a **dense fully connected layer of 128 neurons** and then the output layer has **7 fully connected neurons one for each emotion** with activation set to Softmax**.**

Layer (type)	Output Shape	Param #	Connected to
convolution2d_7 (Convolution2D)	(None, 16, 62, 34)	160	convolution2d_input_3[0][0]
maxpooling2d_7 (MaxPooling2D)	(None, 16, 31, 17)	0	convolution2d_7[0][0]
dropout_7 (Dropout)	(None, 16, 31, 17)	0	maxpooling2d_7[0][0]
flatten_3 (Flatten)	(None, 8432)	0	dropout_7[0][0]
dense_4 (Dense)	(None, 128)	1079424	flatten_3[0][0]
dense_5 (Dense)	(None, 7)	903	dense_4[0][0]

Total params: 1,080,487

Fig 1.9 Model Summary of the First CNN model with one convolution layer with 16 Filters

2. Conv (64) – Relu – Pool – FC

One Convolution Layer of 64 Filters: The first model comprises of **one convolution layer** with 64 filters and receptive field of 3x3 Pixels with stride 1, followed by a **2x2 Pooling Layer** and then the architecture is flattened and is connected to a **dense fully connected layer of 128 neurons** and then the output layer has **7 fully connected neurons one for each emotion** with activation set to Softmax

Layer (type)	Output Shape	Param #	Connected to
convolution2d_8 (Convolution2D)	(None, 64, 62, 34)	640	convolution2d_input_4[0][0]
maxpooling2d_8 (MaxPooling2D)	(None, 64, 31, 17)	0	convolution2d_8[0][0]
dropout_8 (Dropout)	(None, 64, 31, 17)	0	maxpooling2d_8[0][0]
flatten_4 (Flatten)	(None, 33728)	0	dropout_8[0][0]
dense_6 (Dense)	(None, 128)	4317312	flatten_4[0][0]
dense_7 (Dense)	(None, 7)	903	dense_6[0][0]

Total params: 4,318,855

Figure 1.10: Model Summary of the second CNN model with one convolution layer with 64 filters

3. Conv (16) – Relu – Pool - Conv (32) – Relu – Pool - Conv (64) – Relu – Pool – FC

1) **Convolution2D layer with 16 filters**, with the receptive field of 7x7 and no padding with activation function being ReLU.
2) **Maxpooling2D** layer with pooling window size of 2x2
3) Dropout layer for regularization with value 0.2
4) **Convolution2D layer with 32 filters**, with the receptive field of 3x3 and no padding with activation function being ReLU.
5) **Maxpooling2D** layer with pooling window size of 2x2
6) Dropout layer for regularization with value 0.2
7) **Convolution2D layer with 64 filters**, with the receptive field of 3x3 and no padding with activation function being ReLU.
8) **Maxpooling2D** layer with pooling window size of 2x2
9) Dropout layer for regularization with value 0.2
10) Then the model is flattened and connected to a fully connected dense layer with 128 nodes.
11) The last layer is of 7 nodes each representing a class of emotion with activation function set to Softmax.

Layer (type)	Output Shape	Param #	Connected to
convolution2d_4 (Convolution2D)	(None, 16, 62, 34)	160	convolution2d_input_2[0][0]
maxpooling2d_4 (MaxPooling2D)	(None, 16, 31, 17)	0	convolution2d_4[0][0]
dropout_4 (Dropout)	(None, 16, 31, 17)	0	maxpooling2d_4[0][0]
convolution2d_5 (Convolution2D)	(None, 32, 29, 15)	4640	dropout_4[0][0]
maxpooling2d_5 (MaxPooling2D)	(None, 32, 14, 7)	0	convolution2d_5[0][0]
dropout_5 (Dropout)	(None, 32, 14, 7)	0	maxpooling2d_5[0][0]
convolution2d_6 (Convolution2D)	(None, 64, 12, 5)	18496	dropout_5[0][0]
maxpooling2d_6 (MaxPooling2D)	(None, 64, 6, 2)	0	convolution2d_6[0][0]
dropout_6 (Dropout)	(None, 64, 6, 2)	0	maxpooling2d_6[0][0]
flatten_2 (Flatten)	(None, 768)	0	dropout_6[0][0]
dense_2 (Dense)	(None, 128)	98432	flatten_2[0][0]
dense_3 (Dense)	(None, 7)	903	dense_2[0][0]

Total params: 122,631

Figure 1.10: Model Summary of the third deep stocked CNN model with three convolution layer with 16, 32, 64 filters

All the models are then compiled using the categorical cross entropy loss function for 10 iterations.

4. Conv (64) – Relu – Pool - Conv (32) – Relu – Pool - Conv (16) – Relu – Pool – FC

1) **Convolution2D layer with 64 filters**, with the receptive field of 7x7 and no padding with activation function being ReLU.
2) **Maxpooling2D** layer with pooling window size of 2x2
3) Dropout layer for regularization with value 0.2
4) **Convolution2D layer with 32 filters**, with the receptive field of 3x3 and no padding with activation function being ReLU.
5) **Maxpooling2D** layer with pooling window size of 2x2
6) Dropout layer for regularization with value 0.2
7) **Convolution2D layer with 16 filters**, with the receptive field of 3x3 and no padding with activation function being ReLU.
8) **Maxpooling2D** layer with pooling window size of 2x2
9) Dropout layer for regularization with value 0.2
10) Then the model is flattened and connected to a fully connected dense layer with 128 nodes.
11) The last layer is of nodes each representing a class of emotion with activation function set to Softmax.

4. Hardware and Software requirements

1) Python 2.7.13
2) OpenCV 3.2
3) Anaconda (Environment for python to manage packages and versions)
4) Keras (Python library to implement CNN)
5) Theano (works as backend for Keras)
6) Haar-Cascade library (Open CV library for Body detection)

4.1 Dataset

The data we are going to use for our study is being originally created by us. The data consists of photographic still images enacting different emotions. All images are taken in a frontal position with the figure facing the camera, on a controlled solid color background. The stimulus set follows the list of six basic emotions originally inventoried by Ekman (1970).

The Camera used is Samsung Galaxy S7 dual Pixel camera with image resolution of 4032x2268 pixels with aspect ratio 16:9. The distance between camera and image is between 2 to 3 meters, in the presence of uniform luminance.

The dataset was created by the help of 24 male actors and contains 1100 images with actors doing multiple frames for their poses.

Figure 1.12: Dataset images

5. Results and Performance Comparison

5.1 Comparison between different CNN models:

Without using any low-level extraction technique, the aim was to find a good model that could provide a good learning rate and also does not compromise on the processing time. After training the models separately it was observed,

1) CNN with one Convolution layer of 16 filters:

a) The training rate was fast i.e. each epoch took less time to train as the number of filters were very less hence the parameters were less
b) The learning rate was very slow, i.e. it took a lot of epochs for the model to achieve a somewhat decent accuracy.

2) CNN with one Convolution layer of 64 filters:

a) The training rate was slow i.e. each epoch took more time to train as the number of filters were high hence there were many parameters.

b) The learning rate was much better than the first model as with more filters more features could be learned.

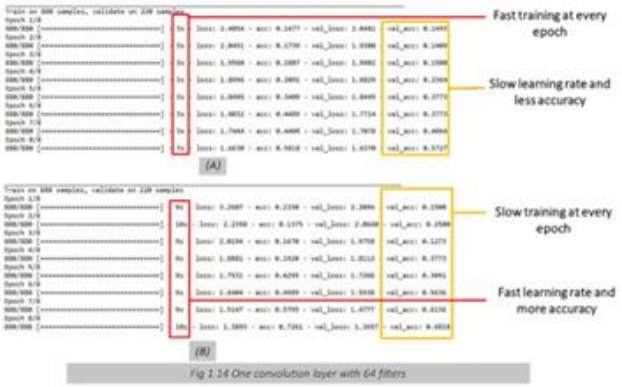


Fig 1.14 One convolution layer with 64 filters

After looking at the observations from the first two models a third model was proposed that could provide an amalgamation of the descent processing time by each epoch along with a good learning rate.

3) CNN with three convolution layers with 16, 32 and 64 filters respectively:

(a) Taking in consideration the Design principles as we slowly increase the number of filters as we go on increasing the depth the learning is not slow and also the processing of each epoch is faster than if only a convolution layer is applied at the beginning.

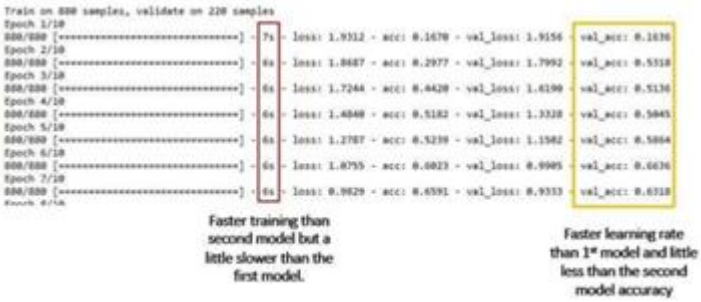


Fig 1.15 CNN with three convolution layers with 16, 32, 64 filters

By observation, it is realized that a good CNN can be designed if the layers are deep and the number of filters increases gradually. Thus, the Final CNN architecture that is used to consist of three convolution layers of 16, 32, 64 filters respectively each having a ReLU Activation, and after every convolution layer there is a 2x2 Pooling layer, with a dense layer of 128 neurons at the end and an output layer of dense network of 7 neurons.

5.2 Comparison between Gabor and Log-Gabor Filters:

The next step is to find a suitable Low-level Feature extraction method thus two of such methods used and tested in the research are Gabor and Log-Gabor filters which help in edge detection. It was important as before the image are sent as input to CNN some unwanted features can be suppressed and in Body pose as body orientation is of utmost importance as a learning criteria for the machine thus these filters were the best choices as they provide with the edge related information of the body which is critical for recognition. After applying both the Filters it was observed that the Log-Gabor Low-level feature extraction achieved better accuracy than the general Gabor Filters.

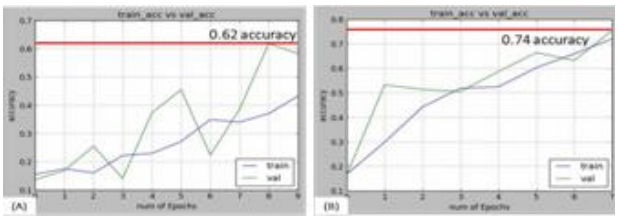


Figure 1.16: Accuracy v/s number of speech plot of CNN model with (A) Low level fracture extraction using Gabor Filters, (B) Low level fracture extraction using Log Gabor Filters

Figure 1.17: One instance of Confusion Matrix of the Final CNN model with Log Gabor Filters out of several tests, with total 210 test images with 30 images with 30 images per expression giving an average accuracy of 73.08%. With individual average accuracy of (Anger85%, Disgusted 94%, Fear 75%, Happy 72%, Sad 73%, Surprised 32%, Neutral 85%)

It was observed that using the Log – Gabor Filters helped the CNN to achieve a gain of 5 % accuracy than a standard model without any preprocessing or low-level feature extraction.

6. Conclusion

From the experiments it can be concluded that the CNN model with Deep Layered Structured is able to learn better and faster than the other two CNN Models with only one layer, also with so preprocessing such as Low-level Feature extraction like Log-Gabor the accuracy can be further improved.

Fig 1.18 shows the Average accuracies noted for various models that were tried and tested and confirms the claims that are made that Log-Gabor low-level Feature extraction before CNN can improve the accuracy of the model.

Convolution Layers	Low level features Extraction	Prediction Accuracy
1 (16 Filters)	-	63%
1 (64 Filters)	-	66%
3 (16-32-64 Filters)	-	68%
3 (16-32-64 Filters)	Gabor	62%
3 (16-32-64 Filters)	**Log Gabor**	72%
3 (16-32-64 Filters)	Log Gabor	69%

Figure 1.18: Result table showing average accuracies for various models

7. Future Prospects

Emotion recognition is one of the key researches these days. With the help of research done in this field using facial, voice, body pose, a multi-model emotion recognition approach can be developed which will be helpful in artificial intelligent systems.

References

[1] **Selene Mota and Rosalind W. Picard**, Automated Posture Analysis for Detecting Learner's Interest Level
[2] **Philipp M. M¨uller, Sikandar Amin, PrateekVermay, MykhayloAndriluka and Andreas Bulling**, Emotion recognition from embedded bodily expressions and speech during dyadic interactions
[3] **Javier G. Rázuri1, Aron Larsson1, Rahim Rahmani1, David Sundgren1, Isis Bonet2, and Antonio Moran,** Recognition of emotions by the emotional feedback through behavioral human poses
[4] **Konrad Schindler, Luc Van Gool, Beatrice de Gelder**, Recognizing emotions expressed by body pose: A biologically inspired neural model
[5] **Jian Bo Yang, Minh Nhut Nguyen, PhyoPhyo San, Xiao Li Li**, Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition
[6] **V. L. Rozaliev, A. V. Zaboleeva-Zotova**, Methods and Models for Identifying Human Emotions by Recognition Gestures and Motion
[7] **Nuria Oliver Eric Horvitz Ashutosh Garg** Layered Representations for Human Activity Recognition
[8] **Ginevra Castellano, Santiago D. Villalba, and Antonio Camurri**, Recognizing Human Emotions from Body Movement and Gesture Dynamics
[9] **Jaeyong Sung and Colin Ponce and Bart Selman and AshutoshSaxena**Human Activity Detection from RGBD Images
[10] **Mark Coulson**, Attributing emotion to static body postures: recognition accuracy, confusions, and viewpoint dependence
[11] https://cs231n.github.io/convolutional-networks/
[12] https://en.wikipedia.org/wiki/Convolutional_neural_network
[13] https://www.coursera.org/learn/machine-learning
[14] Christian Szegedy Google Inc., Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens,
[15] Rethinking the Inception Architecture for Computer Vision