

**MINOR PROJECT**  
On  
**EMOTION DETECTION THROUGH FACIAL EXPRESSION  
RECOGNITION**

Submitted in Partial fulfillment of  
the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY  
(COMPUTER SCIENCE AND ENGINEERING)**

**Submitted By**

**Abhishek Negi(00296302715)**

**Achal Goel(00396302715)**

**Aditya Mathur(35196302715)**

Under the Guidance of

**Ms. Nishtha Jatana**



**Department of Computer Science and Engineering  
MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY,  
JANAKPURI DELHI-58  
GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY  
DELHI, INDIA  
NOV-2018**

## **Acknowledgement**

I truly acknowledge the cooperation and help provided by, Ms. Nishtha Jatana (Assistant Professor) Department of Computer Science and Engineering/Information Technology, Maharaja Surajmal Institute of Technology, C-4 Janakpuri, Delhi, Guru Gobind Singh Indraprastha University, Delhi. She has been a constant source of guidance throughout the course of this project. I am also thankful to my friends and family whose silent support led me to complete my project.

(Signature)

Abhishek Negi (00296302715)

Achal Goel (00396302715)

Aditya Mathur (35196302715)

# CERTIFICATE

This is to certify that the project entitled “**EMOTION DETECTION THROUGH FACIAL EXPRESSION RECOGNITION**” is a bonafide work carried out by Mr./Ms **ABHISHEK NEGI, ACHAL GOEL** and **ADITYA MATHUR** under my guidance and supervision and submitted in partial fulfillment of B.Tech degree in Computer Science and Engineering of Guru Gobind Singh Indraprastha University, Delhi. The work embodied in this project has not been submitted for any other degree or diploma.

(Ms. Nishtha Jatana)

(Assistant Professor)

Dr. Naveen Dahiya

Head, Department of Computer Science and Engineering

MSIT, Delhi.

## List of Figures

Figure Number	Name	Page Number
3.1	Structure of Artificial Neural Network	10
3.2	Why Deep Learning?	11
3.3	Support Vectors shown	11
3.4	Demonstration of Clustering Technique	12
3.5	A simple Bayesian Network example	13
3.6	Result of Similarity and Metric Learning	14
3.7	Working of Genetic Algorithm	15
3.8	Learning Classifier System	16
4.1	Machine Learning	19
4.2	Types of Machine Learning	20
4.3	ANN Architecture	22
4.4	CNN Architecture	23
4.5	SVM Model	24
4.6	Non-Linear classification by SVM using Kernel trick	25
4.7	Python logo	27
4.8	Anaconda logo	28
4.9	Anaconda Navigator	29
4.10	Pandas logo	30
4.11	NumPy logo	31
4.12	OpenCV logo	32
4.13	Scikit-learn logo	34
4.14	Scikit-image logo	35
4.15	Matplotlib logo	36
4.16	Keras logo	37
5.1	Code for retrieving images from dataset(SVM)	39
5.2	Training the model on SVM	39
5.3	Accuracy score and classification report(SVM)	40
5.4	SVM prediction	40
5.5	Image used for prediction(SVM)	40
5.6	Code for adding layers of neural network(ANN)	41
5.7	Loss and accuracy score of ANN	41
5.8	ANN prediction	42
5.9	Image used for Ann prediction	42
5.10	Code for creating training dataset(CNN)	43

5.11	CNN detecting all image files	43
5.12	Code for adding convolutional layers	43
5.13	Code for fitting the training set to the model	44
5.14	CNN trained for 100 epochs	44
5.15	CNN prediction	44

## Table of Contents

<b>1. Acknowledgement</b>	<b>ii</b>
<b>2. Certificate</b>	<b>iii</b>
<b>3. List of figures</b>	<b>iv</b>
<b>4. Table of Contents</b>	<b>v</b>
<b>1. Chapter 1 – Introduction</b>	<b>1</b>
<b>2. Chapter 2 – Literature Survey</b>	<b>4</b>
2.1. Abstract of the Papers	
2.2. Findings from the Papers	
<b>3. Chapter 3 – Technologies in the work domain</b>	<b>8</b>
<b>4. Chapter 4 –Research Methodology</b>	<b>23</b>
4.1. Problem Statement	
4.2. Objectives	
4.3. Solution Proposed	
4.4. Methodology and Technologies used	
<b>5. Chapter 5 – Implementation and results</b>	<b>38</b>
5.1. Architecture	
5.2. Implementation of the methodology along with result snapshots	
<b>6. Chapter 6 – Conclusion</b>	<b>45</b>
<b>7. References</b>	<b>47</b>

# **Chapter 1**

## **Introduction**

## Introduction

Many factors contribute in conveying emotions of an individual. Pose, speech, facial expressions, behavior and actions are some of them. From these above mentioned factors, facial expressions have a higher importance since they are easily perceptible. In communicating with others humans, one can recognize emotions of another human with a considerable level of accuracy. If we can efficiently and effectively utilize heretofore found knowledge in computer science to find practical solutions for automatic recognition of facial emotions, we would be able to attain accuracy that is virtually comparable to the human perception.

This project is carried out on the Extended Cohn-Kanade facial expressions dataset having 7 emotions of each subject. In this project, we are comparing some widely used Classification techniques (SVM, ANN and CNN) to find the best possible algorithm among them for the purpose of emotion detection.

We are using the extended Cohn Kanade dataset of images for the above purpose. First and foremost, the collection of dataset and sorting of the images provided in the dataset based on their facial expression into appropriate categories was performed. The Facial expression categories mentioned in this project to classify the images are as follows neutral, anger, contempt, disgust, fear, happy, sadness, surprise. After this step, testing and training of the dataset was carried out and classification was done using the three algorithms SVM, ANN and CNN so as to compare their results.



## **Chapter 2**

### **Literature Survey**

## 2.1 Abstract of the papers referred

[1] R.Rajesh et al provided a quick survey of facial expression recognition. A comparative study was also carried out using various feature extraction techniques on JAFFE dataset by them. Facial Expression Recognition (FER) consists of five steps mainly that are Preprocessing, Face detection, Facial component extraction, Feature extraction and Classifier. The experiment was conducted by using a total of 213 images of 10 subjects with 7 expressions in which number of expression of a subject varies from 3 to 4 out of which 196 images were taken for experiment and were resized to 128x128/256x256. The face detection was done by using Viola-Jones Face Detection Algorithm. The classification was done by using KNN classifier for which the parameter was set to default with k value equal to 2.

[2] Chih-Wei Hsu et al presents us with a guide on how to get satisfactory results on using support vector machine (SVM) as a classification technique. Support vector machine (SVM) is a very popular and reliable classification technique. Beginners who are not familiar with SVM tend to make mistakes in certain simple but significant steps related to SVM. This problem was countered by Chih-Wei Hsu et al in this paper by providing a “cook-book” procedure to implement Support Vector Machine classification technique. The mentioned approach works well for problems with thousands or more data points. For very large data sets a feasible approach mentioned is to randomly choose a subset of the data set, conduct grid-search on them, and then do a better-region-only grid-search on the complete data set.

[3] Dan C. Ciresan et al presents a fast, fully parameterizable GPU implementation of Convolutional Neural Network variants. The feature extractors are neither carefully designed nor pre-wired, but rather learned in a supervised way. Their deep hierarchical architectures achieve the best published results on benchmarks for object classification (NORB, CIFAR10) and handwritten digit recognition (MNIST), with error rates of 2.53%, 19.51%, 0.35%, respectively. Good results required big and deep but sparsely connected CNNs, computationally prohibitive on CPUs, but feasible on current GPUs,

where our implementation is 10 to 60 times faster than a compiler optimized CPU version.

[4] Evgeny Byvatov et al presented a paper containing the comparison of Support Vector Machine and Artificial Neural Network Systems for drug/non-drug classification. Support vector machine (SVM) and artificial neural network (ANN) systems were applied to a drug/nondrug classification problem as an example of binary decision problems in early-phase virtual compound filtering and screening. The results indicated that solutions obtained by SVM training seem to be more robust with a smaller standard error compared to ANN training. Generally, the SVM classifier yielded slightly higher prediction accuracy than ANN, irrespective of the type of descriptors used for molecule encoding, the size of the training data sets, and the algorithm employed for neural network training. Based on their observation they concluded that SVM represents a useful method for classification tasks in QSAR modelling and virtual screening, especially when large numbers of input variables are used.

[5] Bill Triggs et al presented a paper in which they after reviewing existing edge and gradient based descriptors, showed experimentally that grids of Histograms of Oriented Gradient (HOG) descriptors significantly outperform existing feature sets for human detection. They studied the influence of each stage of the computation on performance, concluding that fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high-quality local contrast normalization in overlapping descriptor blocks are all important for good results. They introduced a more challenging dataset containing over 1800 annotated human images with a large range of pose variations and backgrounds. By default they used a soft ( $C=0.01$ ) linear SVM trained with SVMLight [10] (slightly modified to reduce memory usage for problems with large dense descriptor vectors). Using a Gaussian kernel SVM increased performance by about 3% at 10–4 FPPW at the cost of a much higher run time.

[6] J. Donahue et al presents a paper in which they evaluate whether features extracted from the activation of a deep convolutional network trained in a fully supervised fashion

on a large, fixed set of object recognition tasks can be repurposed to novel generic tasks. They investigate and visualize the semantic clustering of deep convolutional features with respect to a variety of such tasks, including scene recognition, domain adaptation, and fine-grained recognition challenges. In this work, J. Donahue et al analyse the use of deep features applied in a semi-supervised multi-task framework.

[7] A. Krizhevsky et al presented a work in which they trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, they achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers and three fully-connected layers with a final 1000-way softmax. Their work shows that a large, deep convolutional neural network is capable of achieving record breaking results on a highly challenging dataset using purely supervised learning.

[8] Y. Taigman et al presented a paper in which they talk about modern face recognition. In modern face recognition, the conventional pipeline consists of four stages: detect, align, represent and classify. They work on both the alignment step and the representation step by employing explicit 3D face modelling in order to apply a piecewise affine transformation, and derive a face representation from a nine-layer deep neural network. The deep network used for this purpose involves more than 120 million parameters using several locally connected layers without weight sharing, rather than the standard convolutional layers. They used an identity labelled dataset of four million facial images belonging to more than 4,000 identities. Their method reaches an accuracy of 97.35% on the Labelled Faces in the Wild (LFW) dataset, reducing the error of the current state of the art by more than 27%, closely approaching human-level performance

[9] Hoo-Chang Shin et al presents a work related to the computer-aided detection problems. They use deep convolutional networks to tackle the above mentioned problem. They first explore and evaluate different CNN architectures. The models studied by them

contains 5 thousand to 160 million parameters, and vary in numbers of layers. They talk about two specific computer-aided detection (CAdE) problems, namely thoraco-abdominal lymph node (LN) detection and interstitial lung disease (ILD) classification. They achieved the state-of-the-art performance on the mediastinal LN detection, with 85% sensitivity at 3 false positive per patient, and report the first five-fold cross-validation classification results on predicting axial CT slices with ILD categories.

## **2.2 Findings from the Research Papers**

- Facial Expression Recognition (FER) consists of five major steps that are Preprocessing of image, Face detection, Facial component extraction, Feature extraction and Classification.
- Facial Expression Recognition (FER) has many applications some of which are human behavior understanding, detection of mental disorders, detection of synthetic human expression, security systems, lie detection, music for mood etc.
- Facial emotions are categorized into seven, namely, anger, disgust, fear, happy, sad, surprise and neutral.
- Fuzzy Classifier is a powerful classifier to solve classification problem with vague input.
- Some of the most prevalent classification techniques used are Support Vector Machine (SVM), Convolutional Neural Network (CNN) and Artificial Neural Network (ANN).
- SVM classifier yields slightly higher prediction accuracy than ANN according to Evgeny Byvatov et al.
- According to Hoo-Chang Shin et al, the trade-off between using better learning models and using more training data should be carefully considered when searching for an optimal solution to any CAdE problem.
- According to A. Krizhevsky et al, the results improve as the neural network becomes larger and the training time of that network is increased.

## **Chapter 3**

### **Technologies in the Work Domain (Machine Learning)**

## Machine Learning

Machine learning (ML) is a field of artificial intelligence that uses statistical techniques to give computer systems the ability to "learn" (e.g., progressively improve performance on a specific task) from data, without being explicitly programmed.

In the machine learning domain there are various different approaches related to solve a data problem some of which are as follows :-

### Decision Tree Learning

Decision tree learning uses a decision tree as a predictive model, which maps observations about an item to conclusions about the item's target value.

## Building Decision Tree

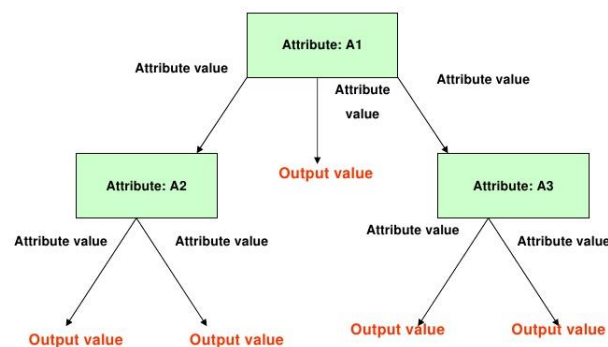


Figure 3.1 Building Decision Tree

### Association Rule Learning

Association rule learning is a method for discovering interesting relations between variables in large databases.

## Artificial Neural Networks

An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is vaguely inspired by biological neural networks. Computations are structured in terms of an interconnected group of artificial neurons, processing information using a connectionist approach to computation. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables.

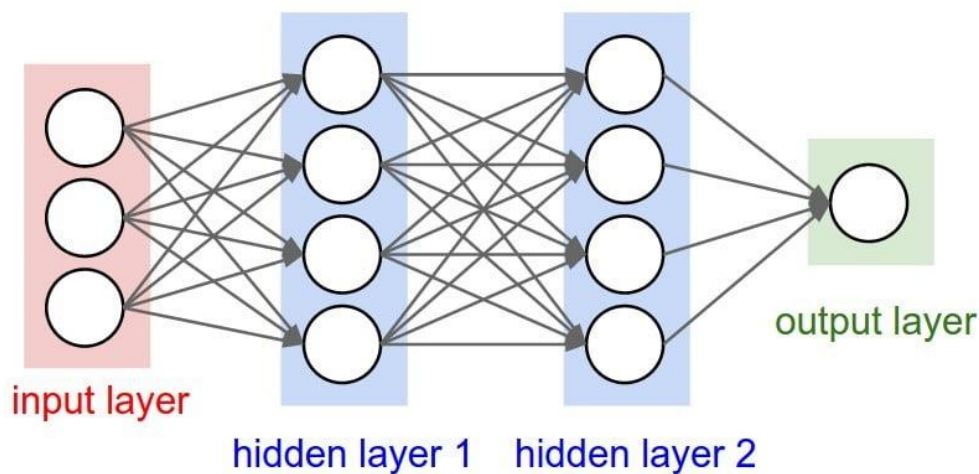


Figure 3.2 Structure of artificial neural network

## Deep Learning

Falling hardware prices and the development of GPUs for personal use in the last few years have contributed to the development of the concept of deep learning which consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing. Some successful applications of deep learning are computer vision and speech recognition.



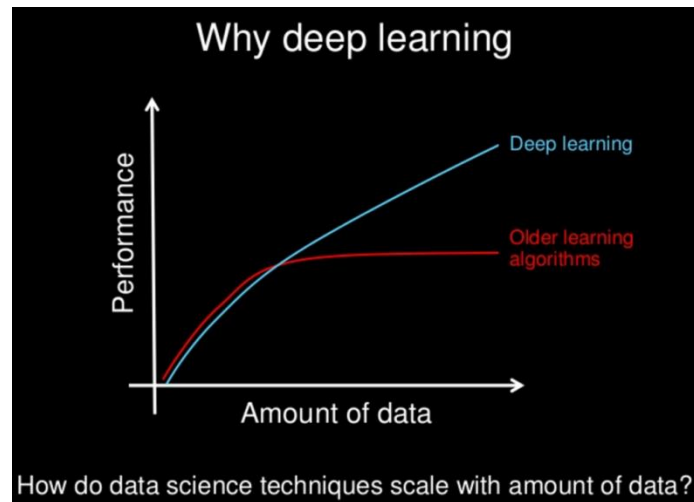


Figure 3.3 Why Deep Learning?

## Support Vector Machines

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other.

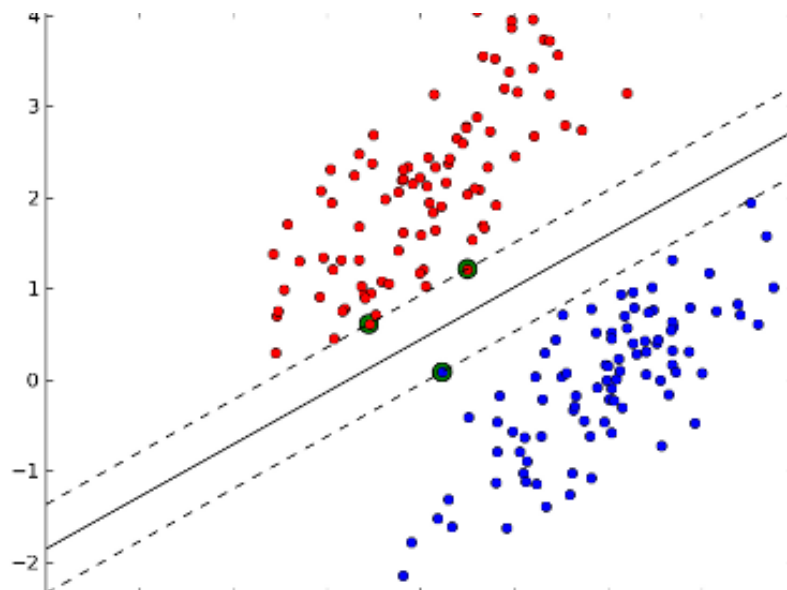


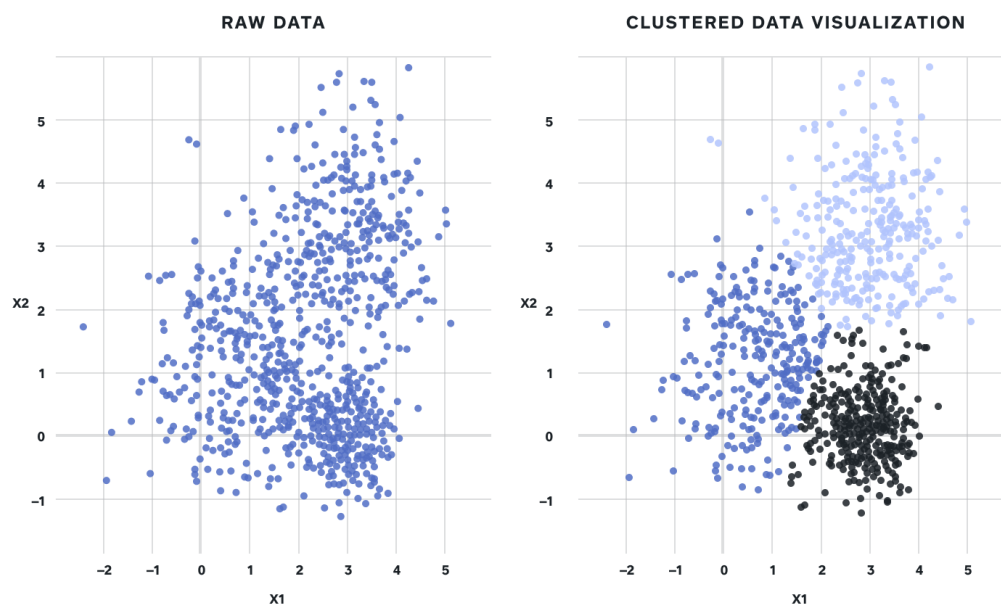
Figure 3.4 Support vectors shown

## Inductive Logic Programming

Inductive logic programming (ILP) is an approach to rule learning using logic programming as a uniform representation for input examples, background knowledge, and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesized logic program that entails all positive and no negative examples. Inductive programming is a related field that considers any kind of programming languages for representing hypotheses (and not only logic programming), such as functional programs.

## Clustering

Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to some predesignated criterion or criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated for example by internal compactness (similarity between members of the same cluster) and separation between different clusters. Other methods are based on estimated density and graph connectivity. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis.



**Figure 3.5 Demonstration of Clustering Technique**

## Bayesian Networks

A Bayesian network, belief network or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning.

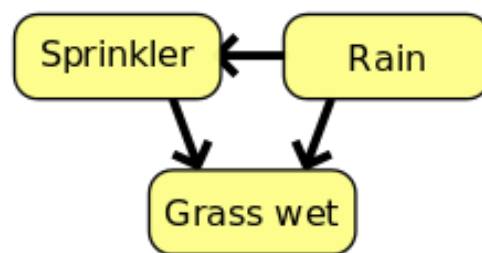


Figure 3.6 A simple Bayesian Network example

In the above Bayesian network example, the rain influences whether the sprinkler is activated, and both rain and the sprinkler influence whether the grass is wet.

## Representation Learning

Several learning algorithms, mostly unsupervised learning algorithms, aim at discovering better representations of the inputs provided during training. Classical examples include principal components analysis and cluster analysis. Representation learning algorithms often attempt to preserve the information in their input but transform it in a way that makes it useful, often as a pre-processing step before performing classification or predictions, allowing reconstruction of the inputs coming from the unknown data generating distribution, while not being necessarily faithful for configurations that are implausible under that distribution.

Manifold learning algorithms attempt to do so under the constraint that the learned representation is low-dimensional. Sparse coding algorithms attempt to do so under the constraint that the learned representation is sparse (has many zeros). Multilinear subspace learning algorithms aim to learn low-dimensional representations directly from tensor representations for multidimensional data, without reshaping them into (high-dimensional) vectors. Deep learning algorithms discover multiple levels of representation, or a hierarchy of features, with higher-level, more abstract features defined in terms of (or

generating) lower-level features. It has been argued that an intelligent machine is one that learns a representation that disentangles the underlying factors of variation that explain the observed data.

## Similarity and Metric Learning

In this problem, the learning machine is given pairs of examples that are considered similar and pairs of less similar objects. It then needs to learn a similarity function (or a distance metric function) that can predict if new objects are similar. It is sometimes used in Recommendation systems.

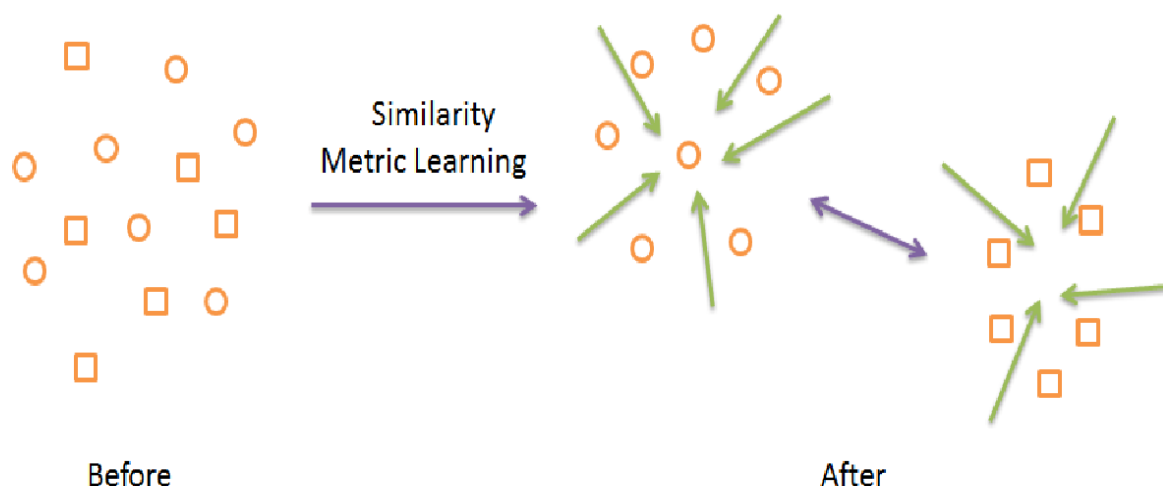


Figure 3.7 Result of Similarity and Metric Learning

## Sparse Dictionary Learning

In this method, a datum is represented as a linear combination of basis functions, and the coefficients are assumed to be sparse. Let  $x$  be a  $d$ -dimensional datum,  $D$  be a  $d$  by  $n$  matrix, where each column of  $D$  represents a basis function.  $r$  is the coefficient to represent  $x$  using  $D$ . Mathematically, sparse dictionary learning means solving  $x \approx Dr$  where  $r$  is sparse. Generally speaking,  $n$  is assumed to be larger than  $d$  to allow the freedom for a sparse representation.

Learning a dictionary along with sparse representations is strongly NP-hard and also difficult to solve approximately. A popular heuristic method for sparse dictionary learning is K-SVD.

Sparse dictionary learning has been applied in several contexts. In classification, the problem is to determine which classes a previously unseen datum belongs to. Suppose a dictionary for each class has already been built. Then a new datum is associated with the class such that it's best sparsely represented by the corresponding dictionary. Sparse dictionary learning has also been applied in image de-noising. The key idea is that a clean image patch can be sparsely represented by an image dictionary, but the noise cannot.

## Genetic Algorithms

A genetic algorithm (GA) is a search heuristic that mimics the process of natural selection, and uses methods such as mutation and crossover to generate new genotype in the hope of finding good solutions to a given problem. In machine learning, genetic algorithms found some uses in the 1980s and 1990s. Conversely, machine learning techniques have been used to improve the performance of genetic and evolutionary algorithms.

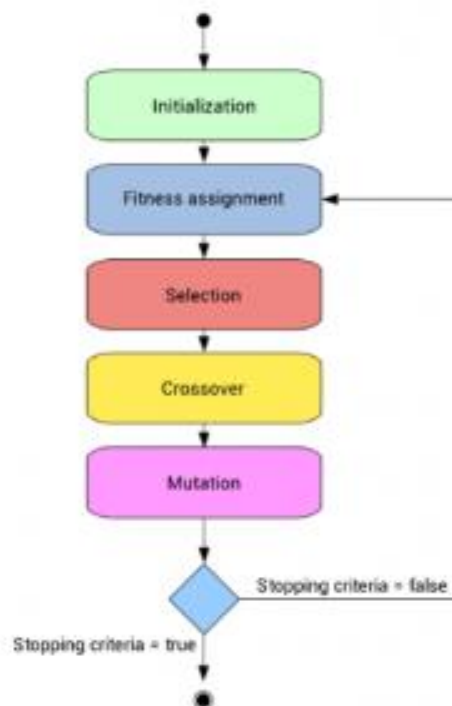


Figure 3.7 Working of Genetic Algorithm

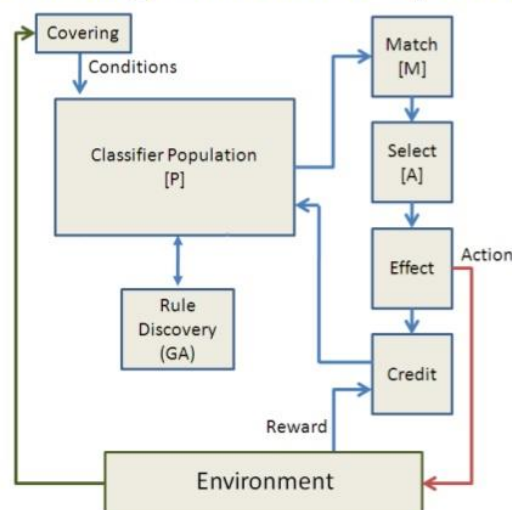
## Rule Based Machine Learning

Rule-based machine learning is a general term for any machine learning method that identifies, learns, or evolves "rules" to store, manipulate or apply, knowledge. The defining characteristic of a rule-based machine learner is the identification and utilization of a set of relational rules that collectively represent the knowledge captured by the system. This is in contrast to other machine learners that commonly identify a singular model that can be universally applied to any instance in order to make a prediction. Rule-based machine learning approaches include learning classifier systems, association rule learning, and artificial immune systems.

### Learning Classifier Systems

Learning classifier systems (LCS) are a family of rule-based machine learning algorithms that combine a discovery component (e.g. typically a genetic algorithm) with a learning component (performing either supervised learning, reinforcement learning, or unsupervised learning). They seek to identify a set of context-dependent rules that collectively store and apply knowledge in a piecewise manner in order to make predictions.

### Learning Classifier System



13

Figure 3.8 Learning Classifier System

## **Chapter 4**

### **Research Methodology**

## **4.1 Problem Statement**

Many factors contribute in conveying emotions of an individual. Pose, speech, facial expressions, behavior and actions are some of them. From these above mentioned factors, facial expressions have a higher importance since they are easily perceptible. In communicating with others humans, one can recognize emotions of another human with a considerable level of accuracy.

If we can efficiently and effectively utilize heretofore found knowledge in computer science to find practical solutions for automatic recognition of facial emotions, we would be able to attain accuracy that is virtually comparable to the human perception.

With this project we are trying to compare the most widely used algorithms (SVM, CNN and ANN) used for classification purposes so that we can present the best algorithm out of the three based on our observations.

In this project, we are using the Cohn-Kanade Image Dataset for emotion detection purposes.

## **4.2 Objective**

The project mainly aims to come up with a solution to the facial expression recognition problem by trying out different algorithms and comparing to find out the best efficient algorithm and to enhance it further. For this, different methodologies and techniques for feature extraction, normalization, selection and classification solutions to these problems as well as taking the computational complexity and timing issues into consideration. Various algorithms and methodologies are studied and hardware resources planning will be done to achieve the goal.

## **4.3 Solution Proposed**

1. Comparing of selected techniques so as to produce a good prediction of emotions by using static images and stating the best among them.
2. Working on different techniques and trying to increase the efficiency and emotion detection rate.





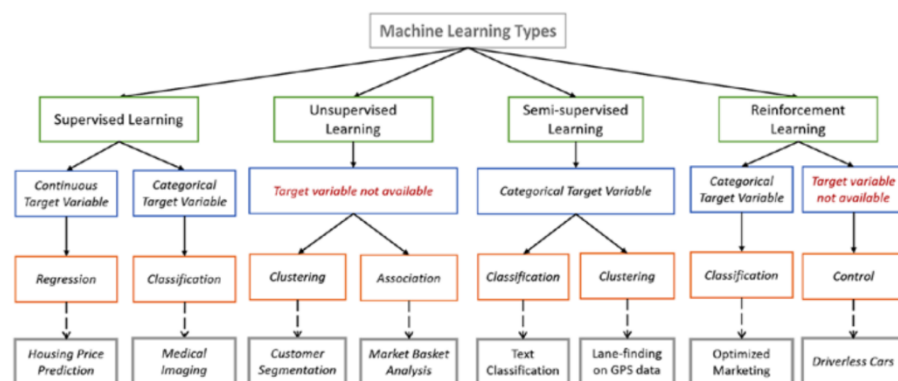
Within the field of data analytics, machine learning is a technique used to devise complicated models and algorithms that lend themselves to prediction; in industrial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

Machine learning tasks are usually categorized into various broad categories:

- Supervised learning: The computer is with example inputs and their preferred outputs, given by means of a "teacher", and the purpose is to study a normal rule that maps inputs to outputs. As special cases, the input signal can be only partly available, or constrained to specific feedback.
- Semi-supervised learning: The computer is given only an incomplete training signal:

a  
training  
set with  
some  
(often  
many)

of the  
target  
outputs missing.



**Figure 4.2 Types of Machine Learning**

- Active learning: The computer can only attain training labels for a limited set of cases (based on a budget), and also has to optimize its preference of objects to acquire labels for. When used interactively, these can be introduced to the user for labelling.
- Unsupervised learning: No labels are given to the learning algorithm, leaving it on its very own to discover structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

- Reinforcement learning: Data (in form of rewards and punishments) are given solely as feedback to the program's actions in a dynamic environment, such as driving a car or playing a game against an opponent.

Another categorization of machine learning tasks arises when one considers the desired output of a machine-learned system:

- In classification, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".
- In regression, also a supervised problem, the outputs are continuous rather than discrete.
- In clustering, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.
- Density estimation finds the distribution of inputs in some space.
- Dimensionality reduction simplifies inputs by mapping them into a lower-dimensional space. Topic modelling is a related problem, where a program is given a list of human language documents and is tasked to find out which documents cover similar topics.

Among different categories of machine learning problems, learning to learn learns its own inductive bias primarily based on previous experience. Developmental learning, elaborated for robotic learning, generates its very own sequences (also referred to as curriculum) of studying situations to cumulatively gather repertoires of novel skills via autonomous self-exploration and social interaction with human instructors and using coaching mechanisms such as active learning, maturation, motor synergies, and imitation.

#### **4.4.2 Artificial Neural Network (ANN)**

An artificial neural network (ANN) learning algorithm, normally called "neural network" (NN), is a learning algorithm that is vaguely stimulated with the aid of biological neural networks. Computations are structured in terms of an interconnected group of artificial

neurons, processing records using a connectionist approach to computation. Modern neural networks are non-linear statistical data modelling tools. They are normally used to model complicated relationships between inputs and outputs, to discover patterns in data, or to seize the statistical structure in an unknown joint probability distribution between observed variables.

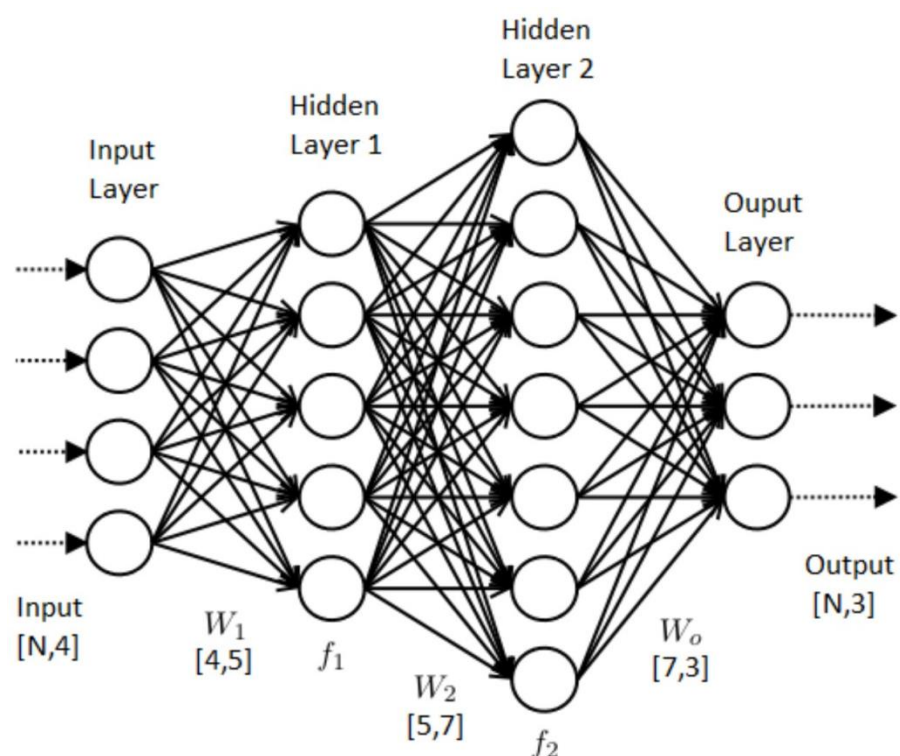
An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal from one artificial neuron to another. An

artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it.

In common ANN implementations, the signal at a connection between artificial

neurons is a real number, and the

output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called 'edges'. Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the



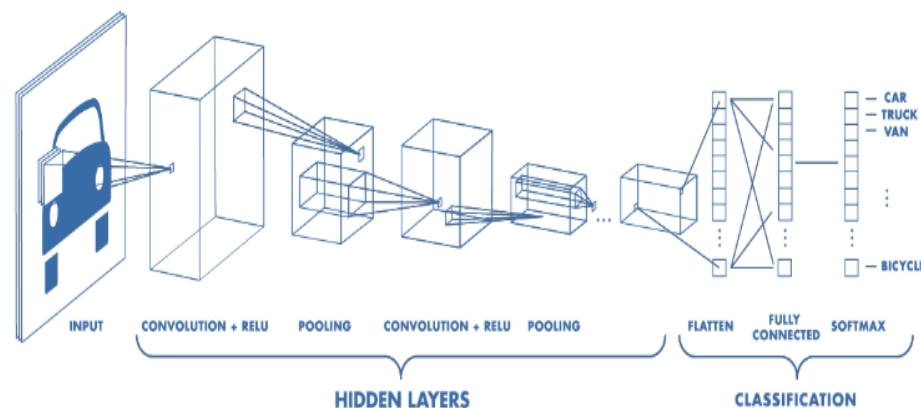
**Figure 4.3 ANN Architecture**

input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

The original goal of the ANN approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. Artificial neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.

#### 4.4.3 Convolutional Neural Network (CNN)

In machine learning, a network (CNN, or ConvNet) is a class of deep, feed forward artificial neural networks, most commonly



**Figure 4.4 CNN Architecture**

applied to analyzing visual imagery. CNNs use a variation of multilayer perceptrons designed to require minimal pre-processing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights structure and translation invariance characteristics.

Convolutional networks have been inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a constrained region of the visual field acknowledged as the receptive field. The receptive fields of distinct neurons partly overlap such that they cover the whole visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms have been hand-engineered. This independence from prior expertise and human effort in feature design is an important advantage. They have applications in image and video recognition, recommender systems and natural language processing.

#### 4.4.4 Support Vector Machine (SVM)

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one class or the other.

Given a set of training examples, each marked as belonging to one or

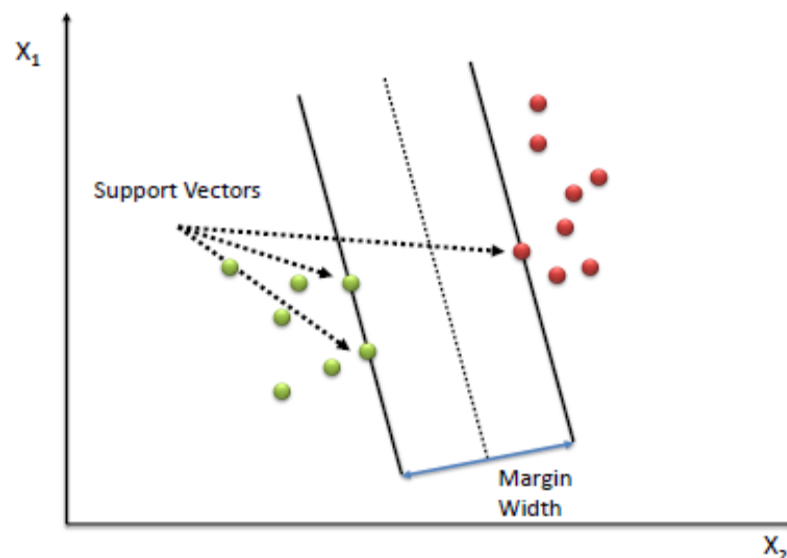


Figure 4.5 SVM Model

the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier(although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). 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.

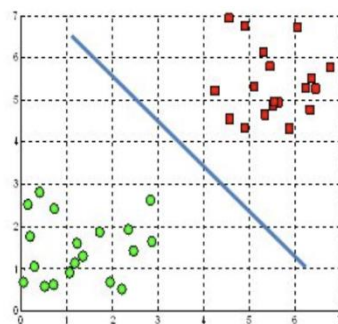
In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data is unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The support vector clustering algorithm, created by Hava Siegelmann and Vladimir Vapnik, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data, and is one of the most widely used clustering algorithms in industrial applications.

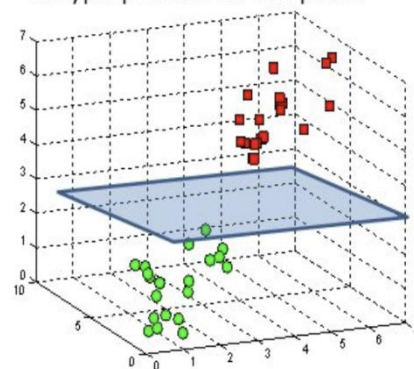
SVMs can be used to solve various real world problems:

- SVMs are helpful in text and hyper-text categorization as

A hyperplane in  $\mathbb{R}^2$  is a line



A hyperplane in  $\mathbb{R}^3$  is a plane



**Figure 4.6 Non-Linear Classification by SVM using Kernel Trick**

their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings.

- Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true of image segmentation systems, including those using a modified version SVM that uses the privileged approach as suggested by Vapnik.
- Hand-written characters can be recognized using SVM.
- The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90% of the compounds



classified correctly. Permutation tests based on SVM weights have been suggested as a mechanism for interpretation of SVM models. Support vector machine weights have also been used to interpret SVM models in the past. Posthoc interpretation of support vector machine models in order to identify features used by the model to make predictions is a relatively new area of research with special significance in the biological sciences.

The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. In 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes. The current standard[according to whom?]incarnation (soft margin) was proposed by Corinna Cortes and Vapnik in 1993 and published in 1995.

Technologies used in this project:-

- Python
- Anaconda

Libraries for Python programming language used in this project:-

- Pandas
- NumPy
- Cv2
- Glob
- Scikit Learn
- Scikit Image
- Pickle
- Matplotlib
- Keras



#### 4.4.5 Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a layout philosophy that emphasizes code readability, notably using large whitespace. It offers constructs that allow clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community after 30 years.

Python features a dynamic type system and automatic memory management. It supports more than one programming paradigms, along with object-oriented, imperative, functional and procedural, and has a massive and comprehensive standard library.

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development

model, as do nearly all of Python's other implementations . Python and CPython are managed with the aid of the non-profit Python Software Foundation.



**Figure 4.7 Python Logo**

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta-programming and meta-objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

#### 4.4.6 Anaconda

Anaconda is a free and open-source distribution of the Python and R programming languages for data science and machine learning purposes (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment. Package versions are managed via the package management system conda. The Anaconda distribution is used by over 6 million users and includes more than 250 popular data-science packages suitable for Windows, Linux, and MacOS.

Anaconda distribution comes with more than 1,000 data packages as well as the Conda package and virtual environment manager, known as Anaconda Navigator, so it eliminates the need to learn to install each library independently.

The open source data packages can be individually installed from the Anaconda repository with the

conda install command or the use of the pip install command that is set up with Anaconda.

Pip packages grant many of the features of conda packages and in most instances



**Figure 4.8 Anaconda Logo**

they can work together. You can additionally make your very own custom packages using the conda build command, and you can share them with others by uploading them to Anaconda Cloud, PyPI or other repositories.

The default installation of Anaconda2 consists of Python 2.7 and Anaconda3 consists of Python 3.7. However, you can create new environments that consist of any version of Python packaged with conda.

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them

in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

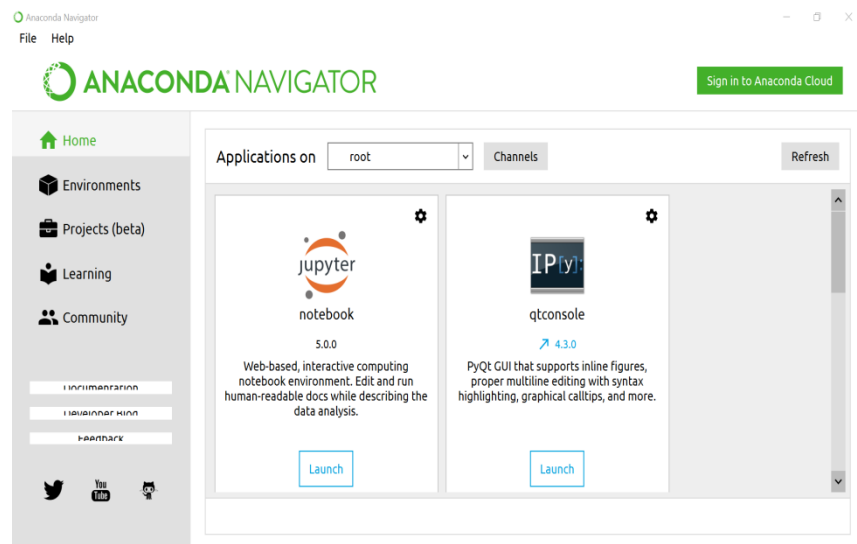
Navigator is automatically included with Anaconda version 4.0.0 or higher.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glueviz
- Orange
- Rstudio
- Visual Studio Code

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs,

runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The Conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.



**Figure 4.9 Anaconda Navigator**

#### 4.4.7 Pandas

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it provides data structures and operations for manipulating numerical tables and time series. It is free software program released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that consist of observations over multiple time periods for the same individuals.

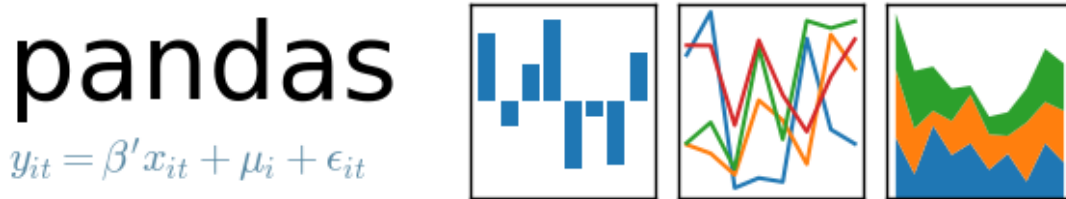


Figure 4.10 Pandas Logo

Features of this library:-

- DataFrame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and subsetting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.
- Provides data filtration.

The library is highly optimized for performance, with critical code paths written in Cython or C.

#### 4.4.8 NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, alongside with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from various other developers. In 2005, Travis Oliphant created NumPy via incorporating facets of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software program and has many contributors.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted,<sup>[15]</sup> and they both allow the user to write fast programs as long as most



**Figure 4.11 NumPy Logo**

operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as

universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

#### 4.4.9 Cv2 (OpenCV)

OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day.

Currently OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations. OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

Python is a general purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability.

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background)



**Figure 4.12 OpenCV Logo**

and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

And the support of Numpy makes the task more easier. Numpy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this.

So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

Alongside entrenched organizations like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that utilize the library, there are numerous new businesses, for example, Applied Minds, VideoSurf, and Zeitera, that make broad utilization of OpenCV. OpenCV's conveyed uses length the range from sewing streetview pictures together, identifying interruptions in reconnaissance video in Israel, observing mine hardware in China, helping robots explore and get objects at Willow Garage, recognition of swimming pool suffocating mishaps in Europe, running intuitive workmanship in Spain and New York, checking runways for flotsam and jetsam in Turkey, reviewing names on items in processing plants far and wide on to fast face identification in Japan.

#### **4.4.10 Glob**

The glob module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell, although results are returned in arbitrary order. No tilde expansion is done, but \*, ?, and character ranges expressed with [] will be correctly matched. This is done by using the `os.scandir()` and `fnmatch.fnmatch()` functions in concert, and not by actually invoking a subshell. Note that unlike `fnmatch.fnmatch()`, glob treats filenames beginning with a dot (.) as special cases. (For tilde and shell variable expansion, use `os.path.expanduser()` and `os.path.expandvars()`.) For a literal match, wrap the meta-characters in brackets. For example, `'[?]'` matches the character '?'.  

---

#### 4.4.11 Scikit-Learn

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy. The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and made the first public release on February the 1st 2010. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012.

As of 2018, scikit-learn is under active development.

Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.



Figure 4.13 scikit-learn Logo



#### 4.4.12 Scikit-Image

scikit-image (formerly scikits.image) is an open source image processing library for the Python programming language. It includes algorithms for segmentation, geometric transformations, color space manipulation, analysis, filtering, morphology, feature detection, and more. It is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The scikit-image project started as scikits.image, by Stéfan van der Walt. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy. The original codebase was later extensively rewritten by other developers. Of the various scikits, scikit-image as well as scikit-learn were described as "well-maintained and popular" in November 2012. Scikit-image has also been active in the Google Summer of Code. Scikit-image is largely written in Python, with some core algorithms written in Cython to achieve performance.



Figure 4.14 scikit-image Logo

#### 4.4.13 Pickle

The pickle module implements binary protocols for serializing and de-serializing a Python object structure. "Pickling" is the process whereby a Python object hierarchy is converted into a byte stream, and "unpickling" is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as "serialization", "marshalling," or "flattening"; however, to avoid confusion, the terms used here are "pickling" and "unpickling".

#### 4.4.14 Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of matplotlib.

Matplotlib was originally written by John D. Hunter, has an active development community, and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.



**Figure 4.15 Matplotlib Logo**

As of 23 June 2017, matplotlib 2.0.x supports Python versions 2.7 through 3.6. Matplotlib 1.2 is the first version of matplotlib to support Python 3.x. Matplotlib 1.4 is the last version of matplotlib to support Python 2.6.

Matplotlib has pledged to not support Python 2 past 2020 by signing the Python 3 Statement. Pyplot is a matplotlib module which provides a MATLAB-like interface.

Matplotlib is designed to be as usable as MATLAB, with the ability to use Python, and the advantage of being free and open-source.

#### 4.4.15 Keras

Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer.

In 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library. Chollet explained that Keras was conceived to be an interface rather than a standalone machine-learning framework. It offers a higher-level, more intuitive set of abstractions that make it easy to develop deep learning models regardless of the computational backend used. Microsoft added a CNTK backend to Keras as well, available as of CNTK v2.0.

Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep learning models on clusters of Graphics Processing Units (GPU) and Tensor processing units (TPU).

Keras claims over 200,000 users as of November 2017. Keras was the 10th most cited tool in the KD Nuggets 2018 software poll and registered a 22% usage.



Figure 4.16 Keras Logo

## **Chapter 5**

### **Implementations and Results**

## 5.1 Implementation

### SVM Implementation

```
72 #====Reading happy image dataset=====#
73 happy_images=[]
74
75 for image in glob.glob('C:/Project/Dataset/sorted_set/happy/*'):
76     happy_images.append(image)
77
78 for i in range(0,len(happy_images)):
79     t=cv2.imread(happy_images[i])
80     t=cv2.cvtColor(t,cv2.COLOR_RGB2GRAY)
81     #t=filters.sobel(t)
82     t1=cv2.resize(t,(64,64),interpolation=cv2.INTER_AREA)
83
84 df=pd.DataFrame(t1)
85 df=df.astype('float32')
86 df4=df/255
87 lab4=np.repeat(3,len(df4))
88
```

Figure 5.1 Retrieving images from dataset

All the images from different emotion folders, in the already sorted dataset are taken together to form a training set for the model. In Figure 5.1, it shown that all the images from happy emotion folder are retrieved and put together into a data frame.

```
209 from sklearn.model_selection import train_test_split
210 X_train,X_test,y_train,y_test=train_test_split(train_data,labels,test_size=0.33,random_state=42)
211 from sklearn.svm import SVC
212 model=SVC(C=0.9, kernel='poly', degree=3, gamma=0.7)
213 model.fit(X_train,y_train)
214
215 print("Model has been sucessfully trained")
```

Figure 5.2 Training the model on SVM

While training the model for SVM algorithm, we used 67 percent of data for training of the model and the rest 33 percent of data for testing of the model after the model has been successfully trained as shown in the Figure 5.2.

```

Windows PowerShell
f 'block_norm == L1' is deprecated and will be changed to 'L2-Hys' in v0.15. To suppress this message, specify explicitly the normalization method.
skimage_deprecation)
C:\Python\Python36\lib\site-packages\sklearn\utils\validation.py:752: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
y = column_or_1d(y, warn=True)
Model has been successfully trained
Training accuracy :
0.42011834319526625
Training classification report :

```

	precision	recall	f1-score	support
0	0.25	0.19	0.22	21
1	0.72	0.64	0.68	28
2	0.61	0.73	0.67	15
3	0.78	0.67	0.72	27
4	0.00	0.00	0.00	1
5	0.12	0.25	0.16	12
6	0.68	0.28	0.39	54
7	0.11	0.18	0.14	11
micro avg	0.42	0.42	0.42	169
macro avg	0.41	0.37	0.37	169
weighted avg	0.56	0.42	0.46	169

**Figure 5.3 Accuracy Score and Classification Report**

In Figure 5.3, it is shown that the training accuracy of the SVM is about 42 percent and also shows classification report for each of the class of emotions where 0=angry, 1=disgust, 2=fear, 3=happy, 4=sad, 5=surprise, 6=contempt, 7=neutral.

```

Windows PowerShell
PS C:\Project\SVM> python test_svm.py
Enter model name :- model1
Enter image name :- img6.png
C:\Python\Python36\lib\site-packages\sklearn\externals\joblib\externals\cloudpickle\cloudpickle.py:47:
DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentati
on for alternative uses
import imp
angry
PS C:\Project\SVM>

```

**Figure 5.4 SVM Prediction**

The above Figure 5.4 shows the prediction of an emotion of a picture named img6.png according to the model trained on SVM algorithm which is shown as 'angry'. The actual image is shown below in Figure 5.5 which also shows an angry emotion.



**Figure 5.5 Image used for prediction**

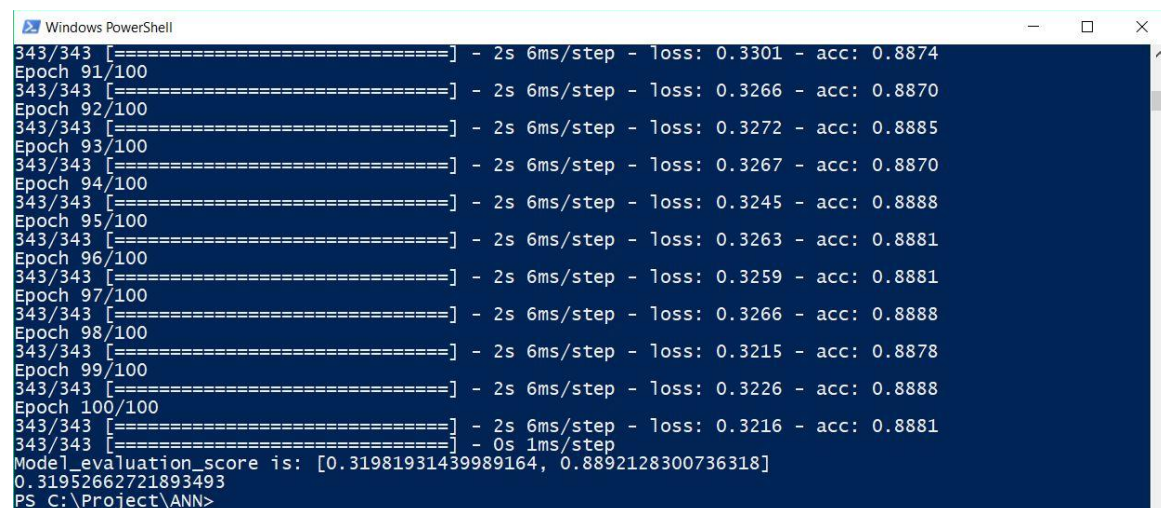
## ANN Implementation

In the process of implementation of ANN algorithm, the construction of dataset for training of model is similar to that which is used in SVM. All the images from different emotion folders are extracted and put together into a single data frame and then is used for training the model.

```
207 from keras.models import Sequential
208 model = Sequential()
209 from keras.layers import Dense
210 model.add(Dense(units=4096, input_dim=X_train.shape[1],activation='relu')) #Input_dimension is number of columns in dataset.
211 model.add(Dense(4096,activation='relu'))
212 model.add(Dense(120,activation='relu'))
213
214 model.add(Dense(8,activation='softmax')) #Here the last layer of neural_network must contain number of classes to be predict
215 #=====compile_model=====
216 model.compile(loss='binary_crossentropy',
217               optimizer='sgd',
218               metrics=['accuracy'])
219 #=====Split_data_into_train_test_split=====
220 # x_train and y_train are Numpy arrays --just like in the Scikit-Learn API.
221 model.fit(X_train,y_train, epochs=100, batch_size=32)
```

Figure 5.6 Adding layers of neural network

Figure 5.6 shows the addition of neural network layers to form a ANN neural network. Sequential model is used for the implementation of ANN, keras.layers.Dense library is used for the purpose of adding network layers and SGD optimizer is used while compilation process. The model shown is trained for 100 epochs as shown in Figure 5.6.



```
Windows PowerShell
343/343 [=====] - 2s 6ms/step - loss: 0.3301 - acc: 0.8874
Epoch 91/100
343/343 [=====] - 2s 6ms/step - loss: 0.3266 - acc: 0.8870
Epoch 92/100
343/343 [=====] - 2s 6ms/step - loss: 0.3272 - acc: 0.8885
Epoch 93/100
343/343 [=====] - 2s 6ms/step - loss: 0.3267 - acc: 0.8870
Epoch 94/100
343/343 [=====] - 2s 6ms/step - loss: 0.3245 - acc: 0.8888
Epoch 95/100
343/343 [=====] - 2s 6ms/step - loss: 0.3263 - acc: 0.8881
Epoch 96/100
343/343 [=====] - 2s 6ms/step - loss: 0.3259 - acc: 0.8881
Epoch 97/100
343/343 [=====] - 2s 6ms/step - loss: 0.3266 - acc: 0.8888
Epoch 98/100
343/343 [=====] - 2s 6ms/step - loss: 0.3215 - acc: 0.8878
Epoch 99/100
343/343 [=====] - 2s 6ms/step - loss: 0.3226 - acc: 0.8888
Epoch 100/100
343/343 [=====] - 2s 6ms/step - loss: 0.3216 - acc: 0.8881
343/343 [=====] - 0s 1ms/step
Model_evaluation_score is: [0.31981931439989164, 0.8892128300736318]
0.31952662721893493
PS C:\Project\ANN>
```

Figure 5.7 Loss and Accuracy score of ANN



The above figure, Figure 5.7 shows the loss and accuracy score of the ANN algorithm trained for 100 epochs and the final epoch shows the accuracy of about 88 percent for the given set of images.

```
PS C:\Project\ANN> python test_ANN.py
Using TensorFlow backend.
Enter model name :- model1
Enter image name :- asd.png
2018-11-12 00:02:36.634281: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports ins
tructions that this TensorFlow binary was not compiled to use: AVX2
Prediction : sad
```

**Figure 5.8 ANN Prediction**



**Figure 5.9 Image used for prediction**

Figure 5.9 is the image used for the prediction of emotion for ANN which is shown in Figure 5.8. It can be seen from the above shown images that ANN is able to identify the emotion of subject shown in Figure 5.9.

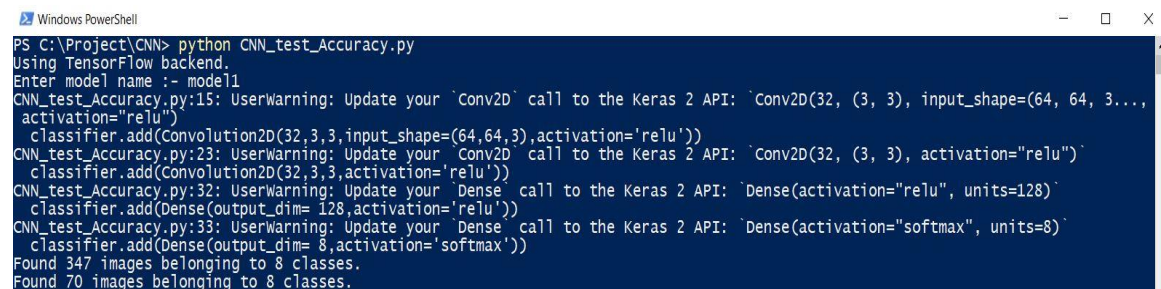


## CNN Implementation

In CNN, the images are not extracted separately for different emotions and combined into single dataset, instead all the images are fed to CNN all together and CNN classifies the images based on the directories in which the images are stored as shown in Figure 5.10 and Figure 5.11.

```
51 training_set = train_datagen.flow_from_directory('C:/Project/CNN/train_data',
52         target_size=(64, 64),
53         batch_size=32,
54         class_mode='categorical')
55
```

Figure 5.10 Creating training set



```
PS C:\Project\CNN> python CNN_test_Accuracy.py
Using TensorFlow backend.
Enter model name :- model1
CNN_test_Accuracy.py:15: UserWarning: Update your `Conv2D` call to the Keras 2 API: `Conv2D(32, (3, 3), input_shape=(64, 64, 3..., activation='relu')`
  classifier.add(Convolution2D(32,3,3,input_shape=(64,64,3),activation='relu'))
CNN_test_Accuracy.py:23: UserWarning: Update your `Conv2D` call to the Keras 2 API: `Conv2D(32, (3, 3), activation='relu')`
  classifier.add(Convolution2D(32,3,3,activation='relu'))
CNN_test_Accuracy.py:32: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(activation='relu', units=128)`
  classifier.add(Dense(output_dim= 128,activation='relu'))
CNN_test_Accuracy.py:33: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(activation='softmax', units=8)`
  classifier.add(Dense(output_dim= 8,activation='softmax'))
Found 347 images belonging to 8 classes.
Found 70 images belonging to 8 classes.
```

Figure 5.11 CNN detecting all the image files

```
13 #==== 1) Building Convolution layer=====#
14
15 classifier.add(Convolution2D(32,3,3,input_shape=(64,64,3),activation='relu'))
16
17 #====2) Pooling=====#
18
19 classifier.add(MaxPooling2D(pool_size=(2,2)))
20
21 #===Adding second CNN layer===|=====#
22
23 classifier.add(Convolution2D(32,3,3,activation='relu'))
24 classifier.add(MaxPooling2D(pool_size=(2,2)))
25
```

Figure 5.12 Adding Convolution Layers

Figure 5.12 shows the addition of two convolution layers in the implementation of CNN algorithm. After the addition of two convolution layers Flattening is done on the neural network.

```

65 classifier.fit_generator(training_set,
66                         steps_per_epoch = 100,
67                         epochs = 50,
68                         validation_data = test_set,
69                         validation_steps = 100)
70
71
72 from keras.models import load_model
73 print(training_set.class_indices) #===This will give the label for all images=====#
74 classifier.save("C:/Project/CNN/saved_model/"+model_name+".h5")
75 print("Training Done")

```

Figure 5.13 Fitting the training set into model

The training of model is done for 50 epochs having 100 steps per epochs and the model is saved for further use as shown in Figure 5.13 and Figure 5.14. The training set accuracy score for CNN is about 94 percent.

```

Windows PowerShell
100/100 [=====] - 40s 399ms/step - loss: 0.1807 - acc: 0.9281 - val_loss: 6.5189 - val_acc: 0.2146
Epoch 40/50
100/100 [=====] - 38s 384ms/step - loss: 0.1604 - acc: 0.9413 - val_loss: 6.9581 - val_acc: 0.1998
Epoch 41/50
100/100 [=====] - 38s 377ms/step - loss: 0.1729 - acc: 0.9370 - val_loss: 6.6052 - val_acc: 0.2434
Epoch 42/50
100/100 [=====] - 40s 396ms/step - loss: 0.1790 - acc: 0.9311 - val_loss: 6.5511 - val_acc: 0.1995
Epoch 43/50
100/100 [=====] - 41s 414ms/step - loss: 0.1769 - acc: 0.9345 - val_loss: 7.0677 - val_acc: 0.2417
Epoch 44/50
100/100 [=====] - 39s 392ms/step - loss: 0.1472 - acc: 0.9489 - val_loss: 7.4773 - val_acc: 0.2156
Epoch 45/50
100/100 [=====] - 39s 389ms/step - loss: 0.1648 - acc: 0.9369 - val_loss: 6.3125 - val_acc: 0.2003
Epoch 46/50
100/100 [=====] - 39s 390ms/step - loss: 0.1504 - acc: 0.9397 - val_loss: 7.2034 - val_acc: 0.2156
Epoch 47/50
100/100 [=====] - 39s 393ms/step - loss: 0.1524 - acc: 0.9436 - val_loss: 6.6808 - val_acc: 0.2276
Epoch 48/50
100/100 [=====] - 40s 398ms/step - loss: 0.1586 - acc: 0.9362 - val_loss: 6.5421 - val_acc: 0.2569
Epoch 49/50
100/100 [=====] - 39s 392ms/step - loss: 0.1428 - acc: 0.9463 - val_loss: 7.0781 - val_acc: 0.2152
Epoch 50/50
100/100 [=====] - 40s 401ms/step - loss: 0.1532 - acc: 0.9418 - val_loss: 6.9680 - val_acc: 0.1990
{'anger': 0, 'contempt': 1, 'disgust': 2, 'fear': 3, 'happy': 4, 'neutral': 5, 'sadness': 6, 'surprise': 7}
Training Done

```

Figure 5.14 Trained for 100 epochs

```

PS C:\Project\CNN> python test_CNN.py
Using TensorFlow backend.
Enter model name :- model1
Enter image name :- img.png
2018-11-12 01:20:37.425687: I tensorflow/core/platform/cpu_feature_guard.cc:141]
ow binary was not compiled to use: AVX2
Predicted Result :
happy
PS C:\Project\CNN> python test_CNN.py
Using TensorFlow backend.
Enter model name :- model1
Enter image name :- img2.png
2018-11-12 01:20:51.149827: I tensorflow/core/platform/cpu_feature_guard.cc:141]
ow binary was not compiled to use: AVX2
Predicted Result :
happy

```

Figure 5.15 CNN Prediction

In Figure 5.15, the result of prediction done CNN on a .png image file is shown for which it predicts the emotion to be 'happy'.

## **Chapter 6**

### **Conclusion**

## Conclusion

In this project, we compare the recognition rate of three different algorithms namely Support Vector Machine (SVM), Convolutional Neural Network (CNN), Artificial Neural Network (ANN) on Extended Cohn Kanade dataset of images for Facial Expression Recognition. First, we collected the dataset and sorted the images provided in the dataset based on their facial expression into appropriate categories. Facial expressions used in this project to classify the images are as follows neutral, anger, contempt, disgust, fear, happy, sadness, surprise. For this purpose we collected the first image that contained the neutral expression and last image that contained the final emotion expression of every subject from every image sequence.

As we proceeded with the testing and training of the dataset using the three different algorithms which are SVM, CNN and ANN. We noted different recognition rate of these algorithms and observed that CNN algorithm stands out and has higher recognition rate than the other two algorithms. The accuracy rate of the algorithms used in this project on Extended Cohn Kanade dataset are as follows, SVM has accuracy rate of 42%, ANN has an accuracy rate of 88% and CNN has an accuracy rate of 94%.

## References

- [1] Jyoti Kumari, R.Rajesh, KM.Pooja, “Facial expression recognition: A survey”, *Procedia Computer Science* 58 (2015) 486-491, Second International Symposium on Computer Vision and the Internet (VisionNet’ 15), 2015.
- [2] Hsu CW, Chang CC, Lin CJ, et al. A practical guide to support vector classification. 2003.
- [3] Dan C. Cires, an, UeliMeier, Jonathan Masci, Luca M. Gambardella, J’urgen Schmidhuber “Flexible, High Performance Convolutional Neural Networks for Image Classification”.
- [4] Evgeny Byvatov, Uli Fechner, Jens Sadowski, and Gisbert Schneider, “Comparison of Support Vector Machine and Artificial Neural Network Systems for Drug/Nondrug Classification” , *J. Chem. Inf. Comput. Sci.* 2003.
- [5] Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: *Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society Conference on. IEEE; 2005;1: 886±93.*
- [6] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.
- [7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.
- [8] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014.
- [9] Hoo-Chang Shin, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, Ronald M. Summers: Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Transactions on Medical Imaging*, 2016.