**Face emotion based music recommendation using deep learning**

**OBJECTIVE:**

The objective of this project is to develop Automatic Facial Expression Recognition System which can take human facial images containing some expression as input and recognize and classify it into seven different expression classes such as happy, sad, anger, disgust.etc

**ABSTRACT:**

Emotion recognition systems have been used frequently both socially and commercially. Various classification methods are used to perform emotion recognition. The convolutional neural network (CNN) has a popular position among these classification methods. The CNN modeling process, which is left to the user experience, is a hard challenging task. Instead of modeling a new system, deep learning from pre-trained models often increases performance and prevents loss of time. In this paper, we have implemented in video processing with haar cascade for training for face model .CNN(convolutional neural network) for classification.

**CHAPTER 1:**

**INTRODUCTION:**

With the advent of modern technology our desires went high and it binds no bounds. In the present era a huge research work is going on in the field of digital image and image processing. The way of progression has been exponential and it is ever increasing. Image Processing is a vast area of research in the present day world and its applications are very widespread. Image processing is the field of signal processing where both the input and output signals are images. One of the most important applications of Image processing is Facial expression recognition. Our emotion is revealed by the expressions in our face. Facial Expressions plays an important role in interpersonal communication. Facial expression is a non verbal scientific gesture which gets expressed in our face as per our emotions. Automatic recognition of facial expression plays an important role in artificial intelligence and robotics and thus it is a need of the generation. Some application related to this includes Personal identification and Access control, Videophone and Teleconferencing, Forensic application, Human-Computer Interaction, Automated Surveillance, Cosmetology and so on.

**LITERATURE SURVEY:**

# Emotion Detection of Autistic Children Using Image Processing

**Abstract:**

Facial Emotion Detection is an approach towards detecting human emotions through facial expressions. Autism Spectrum Disorder is an advance neurobehavioral disorder. Autistic people have repetitive, rude behavior. They are not ready to do social communication. People with this syndrome have problems with emotion recognition. This paper works on detecting the emotions of autistic children from the expression of their faces. This paper works on four emotions. These emotions are sad, happy, neutral, and angry. To detect the emotion of autistic children is performed with image processing and machine learning algorithms. The features are extracted from the faces of autistic children with local binary pattern. Machine learning algorithms are used for classification of emotions. Machine learning classifiers used in classification process are support vector machine and neural network.

# Author: [Pooja Rani](https://ieeexplore.ieee.org/author/37087500548)

# YEAR: 2020

* [Machine Learning](https://ieeexplore.ieee.org/search/searchresult.jsp?matchBoolean=true&queryText=%22Index%20Terms%22:Machine%20Learning&newsearch=true),
* [Support Vector Machine](https://ieeexplore.ieee.org/search/searchresult.jsp?matchBoolean=true&queryText=%22Index%20Terms%22:Support%20Vector%20Machine&newsearch=true),
* [Neural Network](https://ieeexplore.ieee.org/search/searchresult.jsp?matchBoolean=true&queryText=%22Index%20Terms%22:Neural%20Network&newsearch=true),
* [Emotion Detection](https://ieeexplore.ieee.org/search/searchresult.jsp?matchBoolean=true&queryText=%22Index%20Terms%22:Emotion%20Detection&newsearch=true),
* [Image Processing](https://ieeexplore.ieee.org/search/searchresult.jsp?matchBoolean=true&queryText=%22Index%20Terms%22:Image%20Processing&newsearch=true),
* [Local Binary Pattern](https://ieeexplore.ieee.org/search/searchresult.jsp?matchBoolean=true&queryText=%22Index%20Terms%22:Local%20Binary%20Pattern&newsearch=true)

# A Review on Different Facial Feature Extraction Methods for Face Emotions Recognition System

**Abstract:**

Emotion recognition using facial images is the state-of-the-art research area in the human-computer interaction paradigm. In order to recognize emotions from images, the system needs to extract facial features like mouth, eyes, etc. Emotions can also be extracted from frontal and non-frontal images. Traditional methods used for extracting facial features are geometry based method, template based method, and appearance based method. The main focus of this paper is to review different types of facial feature extraction method and this research article illustrates a comparison between different methods and in the end, it describes some future research works that would be helpful to make FER more reliable and efficient.

Author: [Viha Upadhyay](https://ieeexplore.ieee.org/author/37088468205); [Devangi Kotak](https://ieeexplore.ieee.org/author/37088469244)

Year: 2020

# Multi Label Classification for Emotion Analysis of Autism Spectrum Disorder Children using Deep Neural Networks

**Abstract:**

Emotion recognition and analysis is the process of identifying emotions and feelings of a person. Emotion analysis process is accurate in identifying expression in normal people in a single attempt. Emotion analysis is difficult in case of Autism Spectrum Disorder (ASD) children which are suffering with communication problems and speech problems. This paper proposed an optimized deep learning model with multi label classification for predicting ASD and NoASD with emotion analysis in children of age group 1 to 10 years. The kaggle dataset [1] of 1857 ASD children and 1850 Typically Developed (TD) children are used in this paper. Proposed model performance is tested on Yale Expression Dataset [2], CAFÉ children dataset [3] and also tested on social media dataset of autism parents group. The model is implemented by extracting face landmarks and is used to predict ASD and NoASD as first classification label and emotion is detected based on landmarks by computing internal and external distances by feature wise. Convolutional Neural Networks (CNN) is used to work with extracted face landmarks by using optimization methods, dropout, batch normalization and parameter updating. The proposed model is applied to predict 6 emotions irrespective of 4 general emotions with better accuracy.

AUTHOR: [T. Lakshmi Praveena](https://ieeexplore.ieee.org/author/37086451952); [N.V.Muthu Lakshmi](https://ieeexplore.ieee.org/author/37088977758)

Year: 2021

# 4. Image Processing Methods for Face Recognition using Machine Learning Techniques

**Abstract:**

The face is one of the simplest ways to distinguish one another's personal image. Face recognition is a personal identification system which uses a person's personal features to recognize the identity of the individual. Human facial identification is basically a two-phase procedure, including face detection, where the process is carried out very rapidly in people, whereas the second is the implementation of environments that classify the face as persons, when the eye is positioned within a short distance. Stage is then repeated and established to be one of the most researched biometric strategies and established by experts for facial expression recognition. In this study, we implemented the area of face detection and face recognition image processing MTCNN techniques while utilizing the VGG face model dataset. In this initiative, python framework is the program necessity.

AUTHOR: [T. R. Ganesh Babu](https://ieeexplore.ieee.org/author/37089357320); [K. Shenbagadevi](https://ieeexplore.ieee.org/author/37089356767); [V. Sri Shoba](https://ieeexplore.ieee.org/author/37089357184); [S. Shrinidhi](https://ieeexplore.ieee.org/author/37089358513); [J. Sabitha](https://ieeexplore.ieee.org/author/37089358435); [U. Saravanakumar](https://ieeexplore.ieee.org/author/37087042848)

YEAR: 2021

# Virtual Markers based Facial Emotion Recognition using ELM and PNN Classifiers

1. **Abstract:**

Detecting different types of emotional expressions from the subject's face is important for developing intelligent systems for a variety of applications. This present work proposed virtual markers based on Facial emotion expression recognition using the Extreme Learning Machine (ELM) and Probabilistic Neural Network (PNN). A facial emotional expression database is developed with 55 undergraduate university students (male: 35, female: 20) of age range between 20 - 25 years with a mean age of 23.9 years. A HD webcam is used to capture the facial image and Haar Like features and Ada Boost classifier is used to detect the face and eyes through Open CV. A mathematical model based is used to place ten virtual markers called Action Units (AUs) on subjects face at a defined location. Later, Lucas-kanade optical flow algorithm is used to track the marker movement while the subject expressing different emotions and the distance between the center of the face to each marker is used as a feature for classifying emotions. One way Analysis of Variance (ANOVA) is used to test the statistical significance of the features and five fold cross-validation method is used to input the feature for classifiers. In this work, two non-linear classifiers namely, ELM and PNN are used for emotional expression classification. The experimental results give a maximum mean emotion classification rate of 88% and 92% in ELM and PNN classifiers, respectively. Maximum individual class accuracy of happiness - 96%, surprise - 94%, anger- 92%, sadness - 88%, disgust - 90% and fear 89% is achieved using PNN. The experimental results confirm that the proposed system is able to distinguish six different emotional expressions and could be used as a potential tool for a variety of applications which include, e-learning, pain assessment, psychological counseling, human-machine interaction-based applications, etc.

Author: [M Murugappan](https://ieeexplore.ieee.org/author/37669341400); [Vasanthan Maruthapillai](https://ieeexplore.ieee.org/author/37088372860); [Wan Khariunizam](https://ieeexplore.ieee.org/author/37088373353); [A M Mutawa](https://ieeexplore.ieee.org/author/37086372175); [Sai Sruthi](https://ieeexplore.ieee.org/author/37088371027); [Chong Wen Yean](https://ieeexplore.ieee.org/author/37088373261)

YEAR:2021

**Chapter 3:**

**SYSTEM ANALYSIS**

**Existing system**

* Principal Component Analysis
* Geometric methods
* Support vector machine.

**Drawbacks**

* Low discriminatory power and high computational load
* In geometric based methods, the geometric features like distance between speech signals.

**Proposed method**

* Face Emotion recognition for transform features system through textural analysis and NN classifier. The system involves like anger, happy, sad disgust and anger.

**ADVANTAGE:**

* It will easily detect the face and recognise the human expression.
* In this image quality is essential for recognising the face.

Input video

Haar cascade

preprocessing

Feature extraction

Database HDF5

CNN classifier

NN training

Music recommendation

Identification of

emotion such as sad,happy,neutral etc

**Chapter 5**

**Modules description**

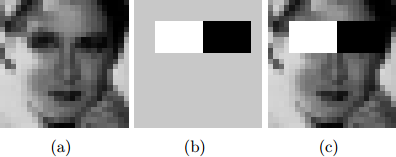
* Input video
* Pre-processing
* Haar cascade
* feature extraction
* CNN CLASSIFIER

**PREPROCESSING**

* Image Pre-processing is a common name for operations with images at the lowest level of abstraction. Its input and output are intensity images. The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing.
* Image restoration is the operation of taking a corrupted/noisy image and estimating the clean original image. Corruption may come in many forms such as motion blur, noise, and camera misfocus. Image restoration is different from image enhancement in that the latter is designed to emphasize features of the image that make the image more pleasing to the observer, but not necessarily to produce realistic data from a scientific point of view. Image enhancement techniques (like contrast stretching or de-blurring by a nearest neighbor procedure) provided by "Imaging packages" use no a priori model of the process that created the image. With image enhancement noise can be effectively be removed by sacrificing some resolution, but this is not acceptable in many applications. In a Fluorescence Microscope resolution in the z-direction is bad as it is. More advanced image processing techniques must be applied to recover the object. De-Convolution is an example of image restoration method. It is capable of: Increasing resolution, especially in the axial direction removing noise increasing contrast.

**Haar – Cascades**

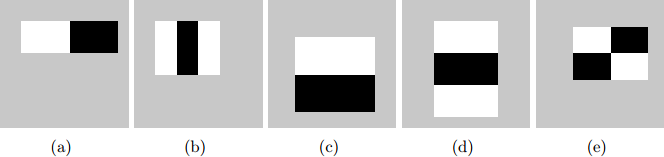
Haar- like features are  rectangular patterns in data. A cascade is a series of “Haar-like features” that are combined to form a classifier [14]. A Haar wavelet is a mathematical function that produces square wave output.



*Figure 2-2.  Haar like Features [13]*

Figure 2.2 shows Haar like features, the background of a template like (b) is painted gray to highlight the pattern’s support. Only those pixels marked in black or white are used when the corresponding feature is calculated [15].

Since no objective distribution can describe the actual prior probability for a given image to have a face, the algorithm must minimize both the false negative and false positive rates in order to achieve an acceptable performance [16]. This then requires an accurate numerical description of what sets human faces apart from other objects. Characteristics that define a face can be extracted from the images with a remarkable committee learning algorithm called Adaboost [17]. Adaboost (Adaptive boost) relies on a committee of weak classifiers that combine to form a strong one through a voting mechanism [18]. A classifier is weak if, in general, it cannot meet a predefined classification target in error terms [7]. The operational algorithm to be used must also work with a reasonable computational budget. Such techniques as the integral image and attention cascades have made the Viola-Jones algorithm [15] highly efficient: fed with a real time image sequence generated from a standard webcam or camera, it performs well on a standard PC.



*Figure 2-3. Haar-like features with different sizes and orientation [13]*

The size and position of a pattern’s support can vary provided its black and white rectangles have the same dimension, border each other and keep their relative positions. Thanks to this constraint, the number of features one can draw from an image is somewhat manageable: a 24 *×* 24 image, for instance, has 43200, 27600, 43200, 27600 and 20736 features of category (a), (b), (c), (d) and (e) respectively as shown in figure 2.3, hence 162336 features in all[13]. In practice, five patterns are considered. The derived features

are assumed to hold all the information needed to characterize a face. Since faces are large and regular by nature, the use of Haar-like patterns.

## How The HAAR – Like Features Work

A scale is chosen for the features say 24 × 24 pixels. This is then slid across the image. The average pixel values under the white area and the black area are then computed. If the difference between the areas is above some threshold then the feature matches [7].

In face detection, since the eyes are of different color tone from the nose, the Haar feature (b) from Figure 2.3 can be scaled to fit that area as shown below,

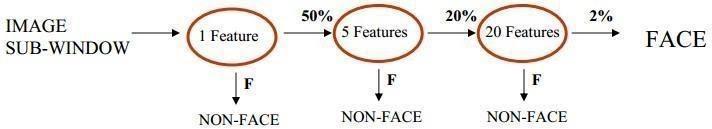


*Figure 2-4. How the Haar like feature of figure 2.3 can be used to scale the eyes*

One Haar feature is however not enough as there are several features that could match it (like the zip drive and white areas at the background of the image of figure 2.4 it is called a “weak classifier.” Haar cascades, the basis of Viola Jones detection framework

[16] therefore consist of a series of weak classifiers whose accuracy is at least 50% correct. If an area passes a single classifier, it moves to the next weak classifier and so on, otherwise, the area does not match.

1. **Cascaded Classifier**



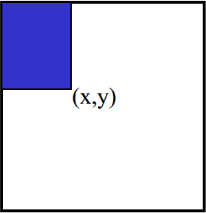
*Figure 2-5. several classifiers combined to enhance face detection*

From figure 2.5, a 1 feature classifier achieves 100% face detection rate and about 50% false positive rate. A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative). A 20 feature classifier achieves 100% detection rate with 10% false positive rate (2% cumulative)[17].Combining several weak classifiers improves the accuracy of detection.

A training algorithm called Adaboost, short for adaptive boosting [14], which had no application before Haar cascades [14], was utilized to combine a series of weak classifiers in to a strong classifier. Adaboost tries out multiple weak classifiers over several rounds, selecting the best weak classifier in each round and combining the best weak classifier to create a strong classifier [7]. Adaboost can use classifiers that are consistently wrong by reversing their decision [7]. In the design and development, it can take weeks of processing time to determine the final cascade sequence [18].

After the final cascade had been constructed, there was a need for a way to quickly compute the Haar features i.e. compute the differences in the two areas. The integral image was instrumental in this.

#### Integral Image



*Figure 2-6. Pixel Coordinates of an integral image*

The Integral image also known as the “summed area table” developed in 1984 came in to widespread use in 2001 with the Haar cascades [4]. A summed area table is created in a single pass. This makes the Haar cascades fast, since the sum of any region in the image can be computed using a single formula [17].

The integral image computes a value at each pixel (x, y) as is shown in figure 2.6, that is the sum of the pixel values above and to the left of (x, y), inclusive. This can quickly be computed in one pass through the image.

Let A, B, C D be the values of the integral image at the corners of a rectangle as shown in figure 2.7.

The sum of original image values within the rectangle can be computed.

𝑆𝑢= 𝐴 − 𝐵 − 𝐶 + 𝐷 - (2.1)

Only three additions are required for any size of rectangle[17]. This face detection approach minimizes computation time while achieving high detection accuracy[15]. It is now used in many areas of computer vision [4] [7].

Improving Face Detection

Face detection can be improved by tuning the detectors parameters to yield satisfactory results. The parameters to be adjusted are explained as follows.

#### Scale Increase Rate.

The scale increase rate specifies how quickly the face detector function should increase the scale for face detection with each pass it makes over an image. Setting the scale increase rate high makes the detector run faster by running fewer passes. If it is set too high it may jump quickly between the scales and miss the faces. The default increase rate in Open CV is 1.1. This implies that the scale increases by a factor of 10 % each pass .The parameters assume a value of 1.1, 1.2, 1.3 or 1.4.

***Minimum Neighbors Threshold***

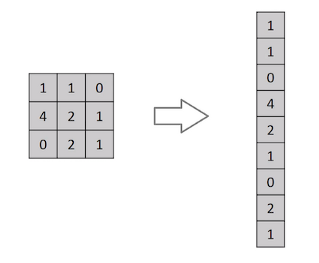
The minimum neighbor’s threshold sets the cutoff level for discarding or keeping rectangle groups as either faces or not. This is based on the number of raw detections in the group and its values ranges from zero to four.

When the face detector is called behind the scenes, each positive face region generates many hits from the Haar detector as in Figure 2.8. The face region itself generates a large cluster of rectangles that to a large extend overlap. The isolated detections are usually false detections and are discarded. The multiple face region detections are then merged in to a single detection. The face detection function does all this before returning the list of the detected faces. The merge step groups rectangles that contain a large number of overlaps and then finds the average.

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

## Why ConvNets over Feed-Forward Neural Nets?



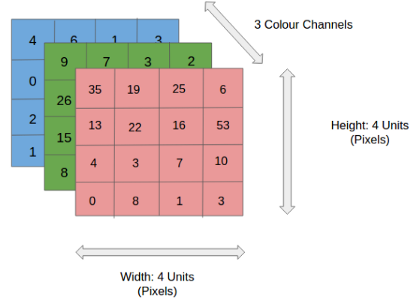
Flattening of a 3x3 image matrix into a 9x1 vector

An image is nothing but a matrix of pixel values, right? So why not just flatten the image (e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes? Uh.. not really.

In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

A ConvNet is able to **successfully capture the Spatial and Temporal dependencies** in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

## Input Image

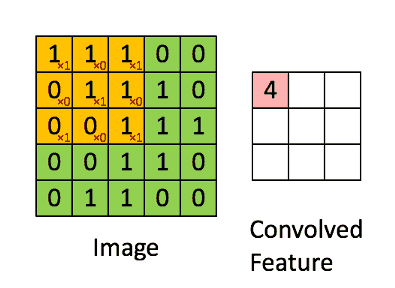


4x4x3 RGB Image

In the figure, we have an RGB image which has been separated by its three color planes — Red, Green, and Blue. There are a number of such color spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc.

You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

## Convolution Layer — The Kernel



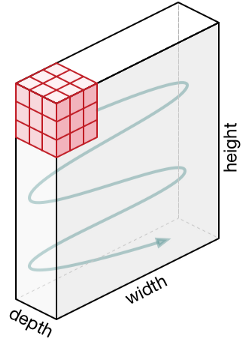
Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature

Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB)

In the above demonstration, the green section resembles our **5x5x1 input image, I**. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the **Kernel/Filter, K**, represented in the color yellow. We have selected **K as a 3x3x1 matrix.**

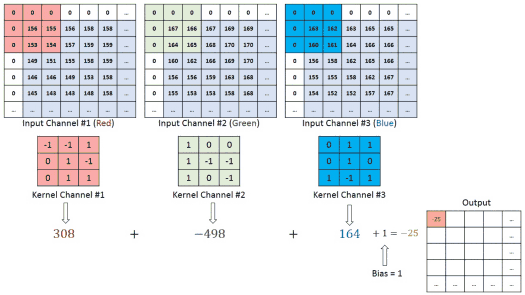
Kernel/Filter, K = 1 0 1  
0 1 0  
1 0 1

The Kernel shifts 9 times because of **Stride Length = 1 (Non-Strided)**, every time performing a **matrix multiplication operation between K and the portion P of the image** over which the kernel is hovering.



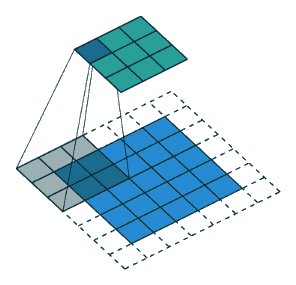
Movement of the Kernel

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.



Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel

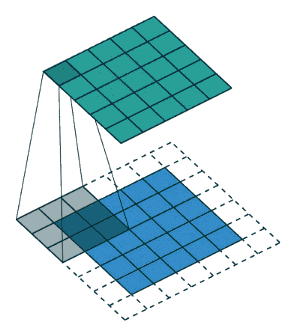
In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image. Matrix Multiplication is performed between Kn and In stack ([K1, I1]; [K2, I2]; [K3, I3]) and all the results are summed with the bias to give us a squashed one-depth channel Convoluted Feature Output.



Convolution Operation with Stride Length = 2

The objective of the Convolution Operation is to **extract the high-level features** such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset, similar to how we would.

There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying **Valid Padding** in case of the former, or **Same Padding** in the case of the latter.



**SAME padding:** 5x5x1 image is padded with 0s to create a 6x6x1 image

When we augment the 5x5x1 image into a 6x6x1 image and then apply the 3x3x1 kernel over it, we find that the convolved matrix turns out to be of dimensions 5x5x1. Hence the name — **Same Padding**.

On the other hand, if we perform the same operation without padding, we are presented with a matrix which has dimensions of the Kernel (3x3x1) itself — **Valid Padding**.

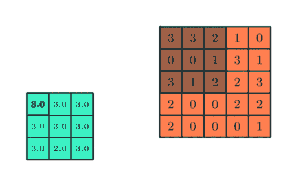
The following repository houses many such GIFs which would help you get a better understanding of how Padding and Stride Length work together to achieve results relevant to our needs.

## [vdumoulin/conv\_arithmetic](https://github.com/vdumoulin/conv_arithmetic" \t "_blank)

### [A technical report on convolution arithmetic in the context of deep learning - vdumoulin/conv\_arithmetic](https://github.com/vdumoulin/conv_arithmetic" \t "_blank)

[github.com](https://github.com/vdumoulin/conv_arithmetic" \t "_blank)

## Pooling Layer

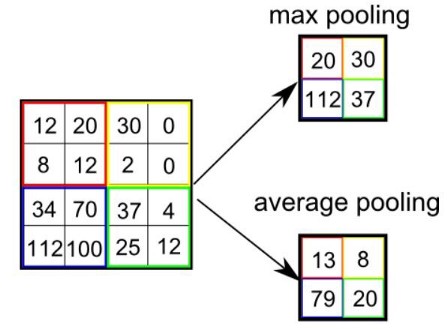


3x3 pooling over 5x5 convolved feature

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to **decrease the computational power required to process the data** through dimensionality reduction. Furthermore, it is useful for **extracting dominant features** which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

There are two types of Pooling: Max Pooling and Average Pooling. **Max Pooling** returns the **maximum value** from the portion of the image covered by the Kernel. On the other hand, **Average Pooling**returns the **average of all the values**from the portion of the image covered by the Kernel.

Max Pooling also performs as a**Noise Suppressant**. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that **Max Pooling performs a lot better than Average Pooling**.

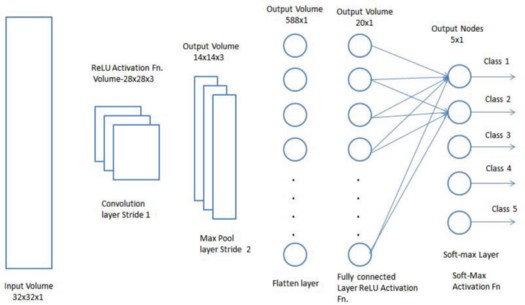


Types of Pooling

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.

After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

## Classification — Fully Connected Layer (FC Layer)



Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the **Softmax Classification** technique.

**Neural Network:**

Neural networks are predictive models loosely based on the action of biological neurons.

The selection of the name “neural network” was one of the great PR successes of the Twentieth Century. It certainly sounds more exciting than a technical description such as “A network of weighted, additive values with nonlinear transfer functions”. However, despite the name, neural networks are far from “thinking machines” or “artificial brains”. A typical artificial neural network might have a hundred neurons. In comparison, the human nervous system is believed to have about 3x1010 neurons. We are still light years from “Data”.

The original “Perceptron” model was developed by Frank Rosenblatt in 1958. Rosenblatt’s model consisted of three layers, (1) a “retina” that distributed inputs to the second layer, (2) “association units” that combine the inputs with weights and trigger a threshold step function which feeds to the output layer, (3) the output layer which combines the values. Unfortunately, the use of a step function in the neurons made the perceptions difficult or impossible to train. A critical analysis of perceptrons published in 1969 by Marvin Minsky and Seymore Paper pointed out a number of critical weaknesses of perceptrons, and, for a period of time, interest in perceptrons waned.

Interest in neural networks was revived in 1986 when David Rumelhart, Geoffrey Hinton and Ronald Williams published “Learning Internal Representations by Error Propagation”. They proposed a multilayer neural network with nonlinear but differentiable transfer functions that avoided the pitfalls of the original perceptron’s step functions. They also provided a reasonably effective training algorithm for neural networks.

**Types of Neural Networks:**

1. Artificial Neural Network
2. Probabilistic Neural Networks
3. General Regression Neural Networks

**DTREG** implements the most widely used types of neural networks:

a) Multilayer Perceptron Networks (also known as multilayer feed-forward network),

b) Cascade Correlation Neural Networks,

c) Probabilistic Neural Networks (NN)

d) General Regression Neural Networks (GRNN).

**Radial Basis Function Networks**:

1. Functional Link Networks,

b) Kohonen networks,

c) Gram-Charlier networks,

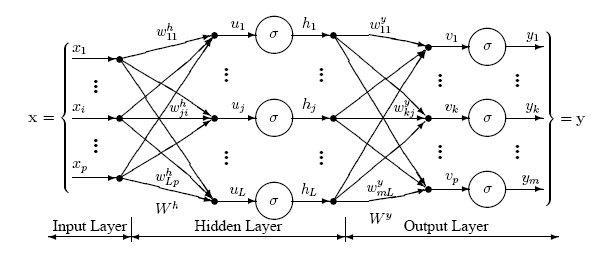
d) Hebb networks,

e) Adaline networks,

f) Hybrid Networks.

**The Multilayer Perceptron Neural Network Model**

The following diagram illustrates a perceptron network with three layers:



This network has an **input layer** (on the left) with three neurons, one **hidden layer** (in the middle) with three neurons and an **output layer** (on the right) with three neurons.

There is one neuron in the input layer for each predictor variable. In the case of categorical variables, *N*-1 neurons are used to represent the *N* categories of the variable.

**Input Layer** — A vector of predictor variable values (*x1...xp*) is presented to the input layer. The input layer (or processing before the input layer) standardizes these values so that the range of each variable is -1 to 1. The input layer distributes the values to each of the neurons in the hidden layer. In addition to the predictor variables, there is a constant input of 1.0, called the *bias* that is fed to each of the hidden layers; the bias is multiplied by a weight and added to the sum going into the neuron.

**Hidden Layer** — Arriving at a neuron in the hidden layer, the value from each input neuron is multiplied by a weight (*wji*), and the resulting weighted values are added together producing a combined value *uj*. The weighted sum (*uj*) is fed into a transfer function, σ, which outputs a value *hj*. The outputs from the hidden layer are distributed to the output layer.

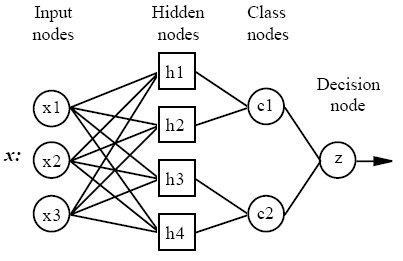
**Output Layer** — Arriving at a neuron in the output layer, the value from each hidden layer neuron is multiplied by a weight (*wkj*), and the resulting weighted values are added together producing a combined value *vj*. The weighted sum (*vj*) is fed into a transfer function, σ, which outputs a value *yk*. The *y* values are the outputs of the network.

If a regression analysis is being performed with a continuous target variable, then there is a single neuron in the output layer, and it generates a single y value. For classification problems with categorical target variables, there are *N* neurons in the output layer producing *N* values, one for each of the *N* categories of the target variable.

**Neural Networks (NN):**

Neural Network (NN) and General Regression Neural Networks (GRNN) have similar architectures, but there is a fundamental difference: networks perform classification where the target variable is categorical, whereas general regression neural networks perform regression where the target variable is continuous. If you select a NN/GRNN network, DTREG will automatically select the correct type of network based on the type of target variable.

**Architecture of a NN:**



All NN networks have four layers:

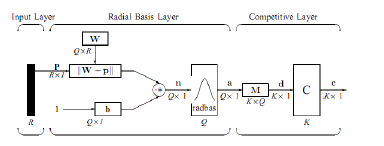
1. **Input layer** — There is one neuron in the input layer for each predictor variable. In the case of categorical variables, *N*-1 neurons are used where *N* is the number of categories. The input neurons (or processing before the input layer) standardizes the range of the values by subtracting the median and dividing by the interquartile range. The input neurons then feed the values to each of the neurons in the hidden layer.
2. **Hidden layer** — This layer has one neuron for each case in the training data set. The neuron stores the values of the predictor variables for the case along with the target value. When presented with the *x* vector of input values from the input layer, a hidden neuron computes the Euclidean distance of the test case from the neuron’s center point and then applies the RBF kernel function using the sigma value(s). The resulting value is passed to the neurons in the pattern layer.
3. **Pattern layer / Summation layer** — The next layer in the network is different for NN networks and for GRNN networks. For NN networks there is one pattern neuron for each category of the target variable. The actual target category of each training case is stored with each hidden neuron; the weighted value coming out of a hidden neuron is fed only to the pattern neuron that corresponds to the hidden neuron’s category. The pattern neurons add the values for the class they represent (hence, it is a weighted vote for that category).

For GRNN networks, there are only two neurons in the pattern layer. One neuron is the denominator summation unit the other is the numerator summation unit. The denominator summation unit adds up the weight values coming from each of the hidden neurons. The numerator summation unit adds up the weight values multiplied by the actual target value for each hidden neuron.

1. **Decision layer** — The decision layer is different for NN and GRNN networks. For NN networks, the decision layer compares the weighted votes for each target category accumulated in the pattern layer and uses the largest vote to predict the target category.

For GRNN networks, the decision layer divides the value accumulated in the numerator summation unit by the value in the denominator summation unit and uses the result as the predicted target value.

The following diagram is actual diagram or propose network used in our project



**1) Input Layer:**

The input vector, denoted as **p**, is presented as the black vertical bar.Its dimension is *R × 1*. In this paper, *R = 3*.

**2) Radial Basis Layer:**

In Radial Basis Layer, the vector distances between input vector p and the weight vector made of each row of weight matrix **W** are calculated. Here, the vector distance is defined as the dot product between two vectors [8]. Assume the dimension of **W** is *Q×R*. The dot product between **p** and the *i*-th row of **W** produces the *i*-th element of the distance vector ||**W-p**||, whose dimension is *Q×1*. The minus symbol, “**-**”, indicates that it is the distance between vectors. Then, the bias vector **b** is combined with ||**W- p**|| by an element-by-element multiplication, .The result is denoted as **n** = ||**W- p**|| **.**.**p**. The transfer function in NN has built into a distance criterion with respect to a center. In this paper, it is defined as *radbas(n)* = 2 *n e*- (1) Each element of **n** is substituted into Eq. 1 and produces corresponding element of **a**, the output vector of Radial Basis Layer. The *i*-th element of **a** can be represented as ai = *radbas*(||**W**i **- p**|| **.**.**b**i) (2) where **W**i is the vector made of the *i*-th row of **W** and **b**i is the *i*-th element of bias vector **b**.

Some characteristics of Radial Basis Layer:

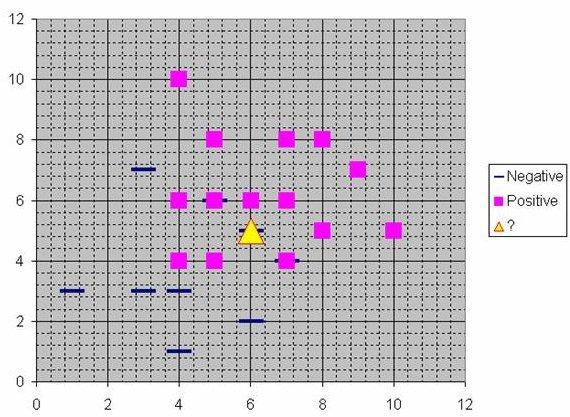
The *i*-th element of a equals to 1 if the input **p** is identical to the *i*th row of input weight matrix **W**. A radial basis neuron with a weight vector close to the input vector **p** produces a value near 1 and then its output weights in the competitive layer will pass their values to the competitive function. It is also possible that several elements of **a** are close to 1 since the input pattern is close to several training patterns. .

**3) Competitive Layer:**

There is no bias in Competitive Layer. In Competitive Layer, the vector **a** is firstly multiplied with layer weight matrix **M**, producing an output vector **d**. The competitive function, denoted as **C** in Fig. 2, produces a 1 corresponding to the largest element of **d**, and 0’s elsewhere. The output vector of competitive function is denoted as **c**. The index of 1 in **c** is the number of tumor that the system can classify. The dimension of output vector, *K*, is 5 in this paper.

**How NN network work:**

Although the implementation is very different, neural networks are conceptually similar to *K-Nearest Neighbor* (k-NN) models. The basic idea is that a predicted target value of an item is likely to be about the same as other items that have close values of the predictor variables. Consider this figure:



Assume that each case in the training set has two predictor variables, *x* and *y*. The cases are plotted using their *x,y* coordinates as shown in the figure. Also assume that the target variable has two categories, *positive* which is denoted by a square and *negative* which is denoted by a dash. Now, suppose we are trying to predict the value of a new case represented by the triangle with predictor values *x*=6, *y*=5.1. Should we predict the target as positive or negative?

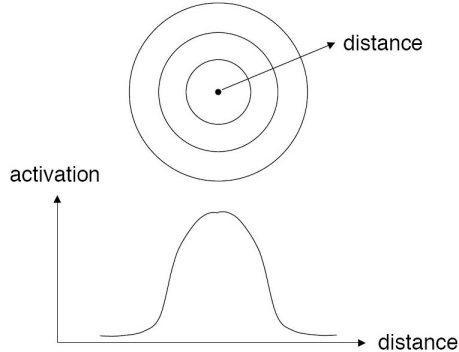
Notice that the triangle is position almost exactly on top of a dash representing a negative value. But that dash is in a fairly unusual position compared to the other dashes which are clustered below the squares and left of center. So it could be that the underlying negative value is an odd case.

The nearest neighbor classification performed for this example depends on how many neighboring points are considered. If 1-NN is used and only the closest point is considered, then clearly the new point should be classified as negative since it is on top of a known negative point. On the other hand, if 9-NN classification is used and the closest 9 points are considered, then the effect of the surrounding 8 positive points may overbalance the close negative point.

A neural network builds on this foundation and generalizes it to consider all of the other points. The distance is computed from the point being evaluated to each of the other points, and a *radial basis function* (RBF) (also called a *kernel function*) is applied to the distance to compute the weight (influence) for each point. The radial basis function is so named because the radius distance is the argument to the function.

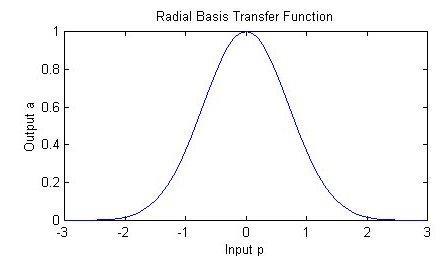
Weight = RBF (*distance*)

The further some other point is from the new point, the less influence it has.



**Radial Basis Function**

Different types of radial basis functions could be used, but the most common is the Gaussian function:



**Advantages and disadvantages of NN networks:**

1. It is usually much faster to train a NN/GRNN network than a multilayer

perceptron network.

1. NN/GRNN networks often are more accurate than multilayer perceptron

networks.

3) NN/GRNN networks are relatively insensitive to outliers (wild points).

4) NN networks generate accurate predicted target probability scores.

5)NN networks approach Bayes optimal classification.

6) NN/GRNN networks are slower than multilayer perceptron networks at

classifying new cases.

7)NN/GRNN networks require more memory space to store the model.

**Removing unnecessary neurons**

One of the disadvantages of NN models compared to multilayer perceptron networks is that NN models are large due to the fact that there is one neuron for each training row. This causes the model to run slower than multilayer perceptron networks when using scoring to predict values for new rows.

DTREG provides an option to cause it remove unnecessary neurons from the model after the model has been constructed.

Removing unnecessary neurons has three benefits:

1. The size of the stored model is reduced.
2. The time required to apply the model during scoring is reduced.
3. Removing neurons often improves the accuracy of the model.

The process of removing unnecessary neurons is an iterative process. Leave-one-out validation is used to measure the error of the model with each neuron removed. The neuron that causes the least increase in error (or possibly the largest reduction in error) is then removed from the model. The process is repeated with the remaining neurons until the stopping criterion is reached.

When unnecessary neurons are removed, the “Model Size” section of the analysis report shows how the error changes with different numbers of neurons. You can see a graphical chart of this by clicking Chart/Model size.

There are three criteria that can be selected to guide the removal of neurons:

1. Minimize error – If this option is selected, then DTREG removes neurons as long as the leave-one-out error remains constant or decreases. It stops when it finds a neuron whose removal would cause the error to increase above the minimum found.
2. Minimize neurons – If this option is selected, DTREG removes neurons until the leave-one-out error would exceed the error for the model with all neurons.
3. # of neurons – If this option is selected, DTREG reduces the least significant neurons until only the specified number of neurons remain.

**CHAPTER 5**

**UML DIAGRAM**

Class diagram:

Database hdf5

preprocessing

Eyebrow detector

Input video

CNN CLASSIFIER

Lip detector

Face detector

Emotions such as happy,sad disgust,neutral etc..

Activity diagram:

Input video

preprocessing

Face detector

Haar feature extraction

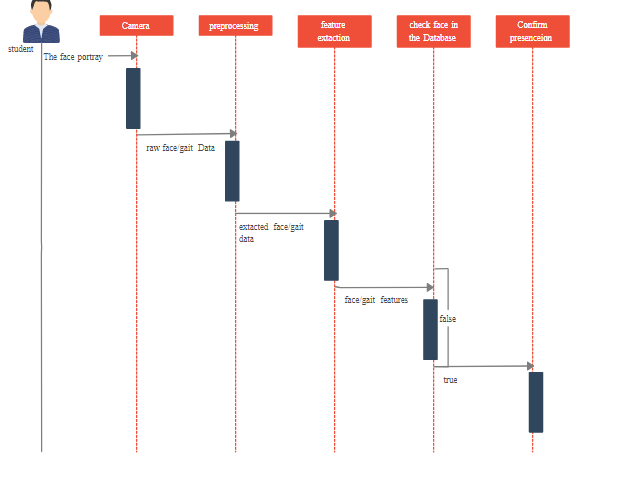
Cnn classifier

Apply algorithm

Emotion happy ,sad,disgust etc..

Music recommendation

**Sequence diagram:**



**Chapter 6**

**Digital image processing**

**DIGITAL IMAGE PROCESSING**

The identification of objects in an image would probably start with image processing techniques such as noise removal, followed by (low-level) feature extraction to locate lines, regions and possibly areas with certain textures.

The clever bit is to interpret collections of these shapes as single objects, e.g. cars on a road, boxes on a conveyor belt or cancerous cells on a microscope slide. One reason this is an AI problem is that an object can appear very different when viewed from different angles or under different lighting. Another problem is deciding what features belong to what object and which are background or shadows etc. The human visual system performs these tasks mostly unconsciously but a computer requires skillful programming and lots of processing power to approach human performance. Manipulating data in the form of an image through several possible techniques. An image is usually interpreted as a two-dimensional array of brightness values, and is most familiarly represented by such patterns as those of a photographic print, slide, television screen, or movie screen. An image can be processed optically or digitally with a computer.

To digitally process an image, it is first necessary to reduce the image to a series of numbers that can be manipulated by the computer. Each number representing the brightness value of the image at a particular location is called a picture element, or pixel. A typical digitized image may have 512 × 512 or roughly 250,000 pixels, although much larger images are becoming common. Once the image has been digitized, there are three basic operations that can be performed on it in the computer. For a point operation, a pixel value in the output image depends on a single pixel value in the input image. For local operations, several neighbouring pixels in the input image determine the value of an output image pixel. In a global operation, all of the input image pixels contribute to an output image pixel value.

These operations, taken [singly](http://www.answers.com/topic/singly) or in combination, are the means by which the image is enhanced, restored, or compressed. An image is enhanced when it is modified so that the information it contains is more clearly evident, but enhancement can also include making the image more visually appealing.

An example is noise smoothing. To smooth a [noisy](http://www.answers.com/topic/noisy) image, median filtering can be applied with a 3 × 3 pixel window. This means that the value of every pixel in the noisy image is recorded, along with the values of its nearest eight neighbours. These nine numbers are then ordered according to size, and the median is selected as the value for the pixel in the new image. As the 3 × 3 window is moved one pixel at a time across the noisy image, the filtered image is formed.

Another example of enhancement is contrast manipulation, where each pixel's value in the new image depends solely on that pixel's value in the old image; in other words, this is a point operation. Contrast manipulation is commonly performed by adjusting the brightness and contrast controls on a television set, or by controlling the exposure and development time in [printmaking](http://www.answers.com/topic/printmaking). Another point operation is that of [pseudo colouring](http://www.answers.com/topic/pseudocoloring) a black-and-white image, by assigning arbitrary colours to the gray levels. This technique is popular in [thermograph](http://www.answers.com/topic/thermography) (the imaging of heat), where hotter objects (with high pixel values) are assigned one color (for example, red), and cool objects (with low pixel values) are assigned another color (for example, blue), with other colours assigned to intermediate values.

Recognizing object classes in real-world images is a long standing goal in Computer vision. Conceptually, this is challenging due to large appearance variations of object instances belonging to the same class. Additionally, distortions from background clutter, scale, and viewpoint variations can render appearances of even the same object instance to be vastly different. Further challenges arise from interclass similarity in which instances from different classes can appear very similar. Consequently, models for object classes must be flexible enough to accommodate class variability, yet discriminative enough to sieve out true object instances in cluttered images. These seemingly paradoxical requirements of an object class model make recognition difficult. This paper addresses two goals of recognition are image classification and object detection. The task of image classification is to determine if an object class is present in an image, while object detection localizes all instances of that class from an image. Toward these goals, the main contribution in this paper is an approach for object class recognition that employs edge information only. The novelty of our approach is that we represent contours by very simple and generic shape primitives of line segments and ellipses, coupled with a flexible method to learn discriminative primitive combinations. These primitives are complementary in nature, where line segment models straight contour and ellipse models curved contour. We choose an ellipse as it is one of the simplest circular shapes, yet is sufficiently flexible to model curved shapes. These shape primitives possess several attractive properties. First, unlike edge-based descriptors they support abstract and perceptually meaningful reasoning like parallelism and adjacency. Also, unlike contour fragment features, storage demands by these primitives are independent of object size and are efficiently represented with four parameters for a line and five parameters for an ellipse.

Additionally, matching between primitives can be efficiently computed (e.g., with geometric properties), unlike contour fragments, which require comparisons between individual edge pixels. Finally, as geometric properties are easily scale normalized, they simplify matching across scales. In contrast, contour fragments are not scale invariant, and one is forced either to rescale fragments, which introduces aliasing effects (e.g., when edge pixels are pulled apart), or to resize an image before extracting fragments, which degrades image resolution.

In recent studies it is shown that the generic nature of line segments and ellipses affords them an innate ability to represent complex shapes and structures. While individually less distinctive, by combining a number of these primitives, we empower a combination to be sufficiently discriminative. Here, each combination is a two-layer abstraction of primitives: pairs of primitives (termed shape tokens) at the first layer, and a learned number of shape tokens at the second layer. We do not constrain a combination to have a fixed number of shape-tokens, but allow it to automatically and flexibly adapt to an object class. This number influences a combination’s ability to represent shapes, where simple shapes favor fewer shape-tokens than complex ones. Consequently, discriminative combinations of varying complexity can be exploited to represent an object class. We learn this combination by exploiting distinguishing shape, geometric, and structural constraints of an object class. Shape constraints describe the visual aspect of shape tokens, while geometric constraints describe its spatial layout (configurations). Structural constraints enforce possible poses/structures of an object by the relationships (e.g., XOR relationship) between shape-tokens.

**CLASSIFICATION OF IMAGES:**

There are 3 types of images used in Digital Image Processing. They are

1. Binary Image
2. Gray Scale Image
3. Colour Image

**BINARY IMAGE:**

A binary image is a [digital image](http://en.wikipedia.org/wiki/Digital_image) that has only two possible values for each [pixel](http://en.wikipedia.org/wiki/Pixel).  Typically the two colours used for a binary image are black and white though any two colours can be used.  The color used for the object(s) in the image is the foreground color while the rest of the image is the background color.

Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit (0 or 1).This name black and white, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as [grayscale images](http://en.wikipedia.org/wiki/Grayscale)

Binary images often arise in [digital image processing](http://en.wikipedia.org/wiki/Digital_image_processing) as [masks](http://en.wikipedia.org/w/index.php?title=Mask_(image_processing)&action=edit&redlink=1) or as the result of certain operations such as [segmentation](http://en.wikipedia.org/wiki/Segmentation_(image_processing)), [thresholding](http://en.wikipedia.org/wiki/Thresholding_(image_processing)), and [dithering](http://en.wikipedia.org/wiki/Dither). Some input/output devices, such as [laser printers](http://en.wikipedia.org/wiki/Laser_printer), [fax machines](http://en.wikipedia.org/wiki/Fax), and bi-level [computer displays](http://en.wikipedia.org/wiki/Visual_display_unit), can only handle bi-level images

**GRAY SCALE IMAGE**

A grayscale Image is [digital image](http://en.wikipedia.org/wiki/Digital_image) is an image in which the value of each [pixel](http://en.wikipedia.org/wiki/Pixel) is a single [sample](http://en.wikipedia.org/wiki/Sample_(signal)), that is, it carries only [intensity](http://en.wikipedia.org/wiki/Luminous_intensity) information. Images of this sort, also known as [black-and-white](http://en.wikipedia.org/wiki/Black-and-white), are composed exclusively of shades of [gray](http://en.wikipedia.org/wiki/Gray) (0-255), varying from black (0) at the weakest intensity to white (255) at the strongest.

Grayscale images are distinct from one-bit [black-and-white](http://en.wikipedia.org/wiki/Black-and-white) images, which in the context of computer imaging are images with only the two [colors](http://en.wikipedia.org/wiki/Color), [black](http://en.wikipedia.org/wiki/Black), and [white](http://en.wikipedia.org/wiki/White) (also called bi-level or [binary images](http://en.wikipedia.org/wiki/Binary_image)). Grayscale images have many shades of gray in between. Grayscale images are also called [monochromatic](http://en.wikipedia.org/wiki/Monochromatic), denoting the absence of any [chromatic](http://en.wikipedia.org/wiki/Chromaticity) variation.

Grayscale images are often the result of measuring the intensity of light at each pixel in a single band of the [electromagnetic spectrum](http://en.wikipedia.org/wiki/Electromagnetic_spectrum) (e.g. [infrared](http://en.wikipedia.org/wiki/Infrared), [visible light](http://en.wikipedia.org/wiki/Visible_spectrum), [ultraviolet](http://en.wikipedia.org/wiki/Ultraviolet), etc.), and in such cases they are monochromatic proper when only a given [frequency](http://en.wikipedia.org/wiki/Frequency) is captured. But also they can be synthesized from a full color image; see the section about converting to grayscale.

**COLOUR IMAGE:**

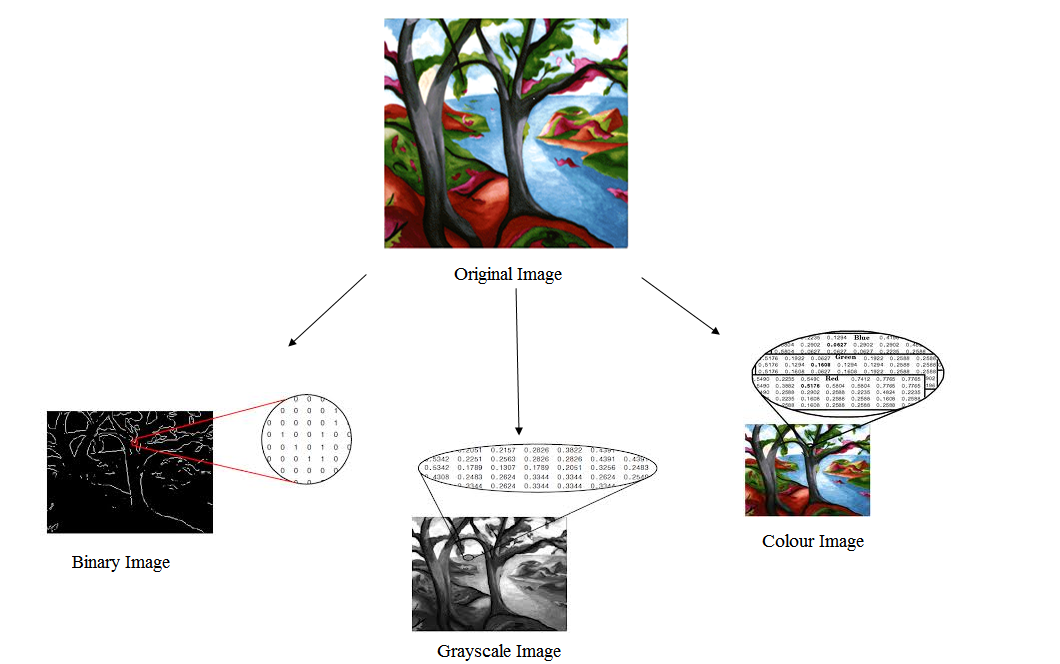
A (digital) color image is a [digital image](http://en.wikipedia.org/wiki/Digital_image) that includes [color](http://en.wikipedia.org/wiki/Color) information for each [pixel](http://en.wikipedia.org/wiki/Pixel). Each pixel has a particular value which determines it’s appearing color. This value is qualified by three numbers giving the decomposition of the color in the three primary colours Red, Green and Blue. Any color visible to human eye can be represented this way. The decomposition of a color in the three primary colours is quantified by a number between 0 and 255. For example, white will be coded as R = 255, G = 255, B = 255; black will be known as (R,G,B) = (0,0,0); and say, bright pink will be : (255,0,255).

In other words, an image is an enormous two-dimensional array of color values, pixels, each of them coded on 3 bytes, representing the three primary colours. This allows the image to contain a total of 256x256x256 = 16.8 million different colours. This technique is also known as RGB encoding, and is specifically adapted to human vision

**CLASSIFICATION OF IMAGES:**

There are 3 types of images used in Digital Image Processing. They are

1. Binary Image
2. Gray Scale Image
3. Colour Image



**Figure 1.1**Representation of a image in binary,grayscale and colour form.

**Binary image:**

A binary image is a [digital image](http://en.wikipedia.org/wiki/Digital_image) that has only two possible values for each [pixel](http://en.wikipedia.org/wiki/Pixel).  Typically the two colours used for a binary image are black and white though any two colours can be used.  The colour used for the objects inthe image is the foreground colour while the rest of the image is the background colour.

Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit (0 or 1) [Refer figure 1.1].This name black and white, monochrome or monochromatic are often used for this concept, but also designate any images that have only one sample per pixel, such as [grayscale images](http://en.wikipedia.org/wiki/Grayscale).

Binary images often arise in [digital image processing](http://en.wikipedia.org/wiki/Digital_image_processing) as [masks](http://en.wikipedia.org/w/index.php?title=Mask_(image_processing)&action=edit&redlink=1) or as the result of certain operations such as [segmentation](http://en.wikipedia.org/wiki/Segmentation_(image_processing)), [thresholding](http://en.wikipedia.org/wiki/Thresholding_(image_processing)) and [dithering](http://en.wikipedia.org/wiki/Dither). Some input/output devices, such as [laser printers](http://en.wikipedia.org/wiki/Laser_printer), [fax machines](http://en.wikipedia.org/wiki/Fax), and bi-level [computer displays](http://en.wikipedia.org/wiki/Visual_display_unit), can only handle bi-level images.

**Gray scale image:**

A grayscale Image is [digital image](http://en.wikipedia.org/wiki/Digital_image) is an image in which the value of each [pixel](http://en.wikipedia.org/wiki/Pixel) is a single [sample](http://en.wikipedia.org/wiki/Sample_(signal)), that is, it carries only [intensity](http://en.wikipedia.org/wiki/Luminous_intensity) information. Images of this sort, also known as [black-and-white](http://en.wikipedia.org/wiki/Black-and-white), are composed exclusively of shades of [gray](http://en.wikipedia.org/wiki/Gray) (0-255), varying from black (0) at the weakest intensity to white (255) at the strongest [Refer figure 1.1].

Grayscale images are distinct from one-bit [black-and-white](http://en.wikipedia.org/wiki/Black-and-white) images, which in the context of computer imaging are images with only the two [colours](http://en.wikipedia.org/wiki/Color), [black](http://en.wikipedia.org/wiki/Black), and [white](http://en.wikipedia.org/wiki/White) (also called bi-level or [binary images](http://en.wikipedia.org/wiki/Binary_image)). Grayscale images have many shades of gray in between. Grayscale images are also called[monochromatic](http://en.wikipedia.org/wiki/Monochromatic), denoting the absence of any [chromatic](http://en.wikipedia.org/wiki/Chromaticity) variation.

Grayscale images are often the result of measuring the intensity of light at each pixel in a single band of the [electromagnetic spectrum](http://en.wikipedia.org/wiki/Electromagnetic_spectrum) (e.g. [infrared](http://en.wikipedia.org/wiki/Infrared), [visible light](http://en.wikipedia.org/wiki/Visible_spectrum), [ultraviolet](http://en.wikipedia.org/wiki/Ultraviolet), etc.), and in such cases they are monochromatic in nature when only a given [frequency](http://en.wikipedia.org/wiki/Frequency) is captured.

**Colour image:**

A colour image is a [digital image](http://en.wikipedia.org/wiki/Digital_image) that includes [colour](http://en.wikipedia.org/wiki/Color) information for each [pixel](http://en.wikipedia.org/wiki/Pixel). Each pixel has a particular value which determines it’s appearing colour. This value is qualified by three numbers giving the decomposition of the colour in the three primary colours Red, Green and Blue. Any colour visible to human eye can be represented this way. The decomposition of a colour in the three primary colours is quantified by a number between 0 and 255. For example, white will be coded as R = 255, G = 255, B = 255; black will be known as (R,G,B) = (0,0,0)[Refer figure 1.1].In other words, an image is an enormous two-dimensional array of colour values, pixels, each of them coded on 3 bytes, representing the three primary colours. This allows the image to contain a total of 256x256x256 = 16.8 million different colours. This technique is also known as RGB encoding, and is specifically adapted to human vision.

**APPLICATIONS**

**Digital camera images**

Digital cameras generally include dedicated digital image processing chips to convert the raw data from the [image sensor](http://en.wikipedia.org/wiki/Image_sensor) into a [colour-corrected](http://en.wikipedia.org/wiki/Color_correction) image in a standard [image file format](http://en.wikipedia.org/wiki/Image_file_formats). Images from digital cameras often receive further processing to improve their quality, a distinct advantage that digital cameras have over [film](http://en.wikipedia.org/wiki/Photographic_film) cameras. The digital image processing typically is executed by special software programs that can manipulate the images in many ways. Many digital cameras also enable viewing of [histograms](http://en.wikipedia.org/wiki/Histogram) of images, as an aid for the photographer to understand the rendered brightness range of each shot more readily.

**Intelligent transportation systems**

Digital image processing has wide applications in intelligent transportation systems, such as [automatic number plate recognition](http://en.wikipedia.org/wiki/Automatic_number_plate_recognition) and [traffic sign recognition](http://en.wikipedia.org/wiki/Traffic_sign_recognition).

**Chapter 6**

**Algorithm description**

**Convolution Neural** **Networks**

**Hardware and software requirement**

**Hardware requirement:**

* Hdd=1tb

**Software requirement**

* Python ,
* anaconda navigator

**4.2.2PYTHON**

Python is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level programming language](https://en.wikipedia.org/wiki/High-level_programming_language) for programming Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

**PYTHON FEATURES:**

Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple programming paradigms, including object-oriented , [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive library. Python is a [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language). [Object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) and [structured programming](https://en.wikipedia.org/wiki/Structured_programming) are fully supported, and many of its features support functional programming and aspect [-oriented programming](https://en.wikipedia.org/wiki/Aspect-oriented_programming) (including by meta [programming](https://en.wikipedia.org/wiki/Metaprogramming) and [meta objects](https://en.wikipedia.org/wiki/Metaobject) (magic methods)). Many other paradigms are supported via extensions, including [design by contract](https://en.wikipedia.org/wiki/Design_by_contract) and [logic programming](https://en.wikipedia.org/wiki/Logic_programming).

**PYTHON LIBRARIES**

Python's large [standard library](https://en.wikipedia.org/wiki/Standard_library), commonly cited as one of its greatest strengths, provides tools suited too many tasks. For Internet-facing applications,

18

many standard formats and protocols such as [MIME](https://en.wikipedia.org/wiki/MIME) and [HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) are supported. It

includes modules for creating [graphical user interfaces](https://en.wikipedia.org/wiki/Graphical_user_interface), connecting to [relational databases](https://en.wikipedia.org/wiki/Relational_database), [generating pseudorandom numbers](https://en.wikipedia.org/wiki/Pseudorandom_number_generator), arithmetic with arbitrary precision decimals, manipulating [regular expressions](https://en.wikipedia.org/wiki/Regular_expression), and [unit testin](https://en.wikipedia.org/wiki/Unit_testing)g.

As of March 2018, the [Python Package Index](https://en.wikipedia.org/wiki/Python_Package_Index) (PyPI), the official repository for third-party Python software, contains over 130,000 packages with a wide range of functionality, including:

* Graphical user interfaces
* Web frameworks
* Multimedia
* Databases
* Networking
* Test frameworks
* Automation
* Web scraping
* Documentation
* System administration
* Scientific computing
* Text processing
* Image processing

**DEPLOYMENT ENVIRONMENT**

Most Python implementations (including CPython) include a [read–eval–print loop](https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop) (REPL), permitting them to function as a [command line interpreter](https://en.wikipedia.org/wiki/Command_line_interpreter) for

which the user enters statements sequentially and receives results immediately.

**OPENCV-PYTHON**

 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) 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.

Since OpenCV is an open source initiative, all are welcome to make contributions to this library. And it is same for this tutorial also. So, if you find any mistake in this tutorial (whether it be a small spelling mistake or a big error in code or concepts, whatever), feel free to correct it

And that will be a good task for freshers who begin to contribute to open source projects. Just fork the OpenCV in github, make necessary corrections and send a pull request to OpenCV. OpenCV developers will check your pull request, give you important feedback and once it passes the approval of the reviewer, it will be merged to OpenCV. Then you become a open source contributor. Similar is the case with other tutorials, documentation etc. As new modules are added to OpenCV-Python, this tutorial will have to be expanded. So those who knows about particular algorithm can write up a tutorial which includes a basic theory of the algorithm and a code showing basic usage of the algorithm and submit it to OpenCV. Remember, we together can make this project a great success !!!

**ARRAY ATTRIBUTES**

Array attributes reflect information that is intrinsic to the array itself. Generally, accessing an array through its attributes allows you to get and sometimes set intrinsic properties of the array without creating a new array. The exposed attributes are the core parts of an array and only some of them can be reset meaningfully without creating a new array.

For a 1-D array, this has no effect. (To change between column and row vectors, first cast the 1-D array into a matrix object.) For a 2-D array, this is the usual matrix transpose. For an n-D array, if axes are given, their order indicates how the axes are permuted (see Examples). If axes are not provided and a.shape = (i[0], i[1], ... i[n-2], i[n-1]), then a.transpose().shape = (i[n-1], i[n-2], ... i[1], i[0]).

**SCALARS**

Python defines only one type of a particular data class (there is only one integer type, one floating-point type, etc.). This can be convenient in applications that don’t need to be concerned with all the ways data can be represented in a computer. For scientific computing, however, more control is often needed. In NumPy, there are 24 new fundamental Python types to describe different types of scalars. These type descriptors are mostly based on the types available in the C language that CPython is written in, with several additional types compatible with Python’s types.

**METHODS**

Array scalars have exactly the same methods as arrays. The default behavior of these methods is to internally convert the scalar to an equivalent 0-dimensional array and to call the corresponding array method. In addition, math operations on array scalars are defined so that the same hardware flags are set and used to interpret the results as for ufunc, so that the error state used for ufuncs also carries over to the math on array scalars.

**DATA TYPE OBJECTS (DTYPE)**

A data type object (an instance of numpy.dtype class) describes how the bytes in the fixed-size block of memory corresponding to an array item should be interpreted. It describes the following aspects of the data: 1. Type of the data (integer, float, Python object, etc.) 2. Size of the data (how many bytes is in e.g. the integer) 3. Byte order of the data (little-endian or big-endian) 4. If the data type is structured, an aggregate of other data types, (e.g., describing an array item consisting of an integer and a float), (a) what are the names of the “fields” of the structure, by which they can be accessed, (b) what is the data-type of each field, and (c) which part of the memory block each field takes. 5. If the data type is a sub-array, what is its shape and data type. To describe the type of scalar data, there are several built-in scalar types in Numpy for various precision of integers, floating-point numbers, etc. An item extracted from an array, e.g., by indexing, will be a Python object whose type is the scalar type associated with the data type of the array. Note that the scalar types are not dtype objects, even though they can be used in place of one whenever a data type specification is needed in Numpy. Structured data types are formed by creating a data type whose fields contain other data types. Each field has a name by which it can be accessed. The parent data type should be of sufficient size to contain all its fields; the parent is nearly always based on the void type which allows an arbitrary item size. Structured data types may also contain nested structured sub-array data types in their fields. Finally, a data type can describe items that are themselves arrays of items of another data type. These sub-arrays must, however, be of a fixed size. If an array is created using a data-type describing a sub-array, the dimensions of the sub-array are appended to the shape of the array when the array is created. Sub-arrays in a field of a structured type behave differently, see Field Access. Sub-arrays always have a C-contiguous memory layout

**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.

**OpenCV-Python**

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) 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.

**OpenCV-Python Tutorials**

OpenCV introduces a new set of tutorials which will guide you through various functions available in OpenCV-Python. **This guide is mainly focused on OpenCV 3.x version** (although most of the tutorials will work with OpenCV 2.x also).

A prior knowledge on Python and Numpy is required before starting because they won’t be covered in this guide. **Especially, a good knowledge on Numpy is must to write optimized codes in OpenCV-Python.**

This tutorial has been started by *Abid Rahman K.* as part of Google Summer of Code 2013 program, under the guidance of *Alexander Mordvintsev*.

**OpenCV Needs You !!!**

Since OpenCV is an open source initiative, all are welcome to make contributions to this library. And it is same for this tutorial also.

So, if you find any mistake in this tutorial (whether it be a small spelling mistake or a big error in code or concepts, whatever), feel free to correct it.

And that will be a good task for freshers who begin to contribute to open source projects. Just fork the OpenCV in github, make necessary corrections and send a pull request to OpenCV. OpenCV developers will check your pull request, give you important feedback and once it passes the approval of the reviewer, it will be merged to OpenCV. Then you become a open source contributor. Similar is the case with other tutorials, documentation etc.

As new modules are added to OpenCV-Python, this tutorial will have to be expanded. So those who knows about particular algorithm can write up a tutorial which includes a basic theory of the algorithm and a code showing basic usage of the algorithm and submit it to OpenCV.

Remember, we **together** can make this project a great success !!!

**Contributors**

Below is the list of contributors who submitted tutorials to OpenCV-Python.

1. Alexander Mordvintsev (GSoC-2013 mentor)
2. Abid Rahman K. (GSoC-2013 intern)

**Additional Resources**

1. A Quick guide to Python - [A Byte of Python](http://swaroopch.com/notes/python/)
2. [Basic Numpy Tutorials](http://wiki.scipy.org/Tentative_NumPy_Tutorial)
3. [Numpy Examples List](http://wiki.scipy.org/Numpy_Example_List)
4. [OpenCV Documentation](http://docs.opencv.org/)
5. [OpenCV Forum](http://answers.opencv.org/questions/)

# Install OpenCV-Python in Windows

## Goals

**In this tutorial**

* We will learn to setup OpenCV-Python in your Windows system.

*Below steps are tested in a Windows 7-64 bit machine with Visual Studio 2010 and Visual Studio 2012. The screenshots shows VS2012.*

## Installing OpenCV from prebuilt binaries

1. Below Python packages are to be downloaded and installed to their default locations.

1.1. [Python-2.7.x](http://python.org/ftp/python/2.7.5/python-2.7.5.msi).

1.2. [Numpy](http://sourceforge.net/projects/numpy/files/NumPy/1.7.1/numpy-1.7.1-win32-superpack-python2.7.exe/download).

1.3. [Matplotlib](https://downloads.sourceforge.net/project/matplotlib/matplotlib/matplotlib-1.3.0/matplotlib-1.3.0.win32-py2.7.exe) (*Matplotlib is optional, but recommended since we use it a lot in our tutorials*).

1. Install all packages into their default locations. Python will be installed to **C:/Python27/**.
2. After installation, open Python IDLE. Enter import numpy and make sure Numpy is working fine.
3. Download latest OpenCV release from [sourceforge site](http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.6/OpenCV-2.4.6.0.exe/download) and double-click to extract it.
4. Goto **opencv/build/python/2.7** folder.
5. Copy **cv2.pyd** to **C:/Python27/lib/site-packeges**.
6. Open Python IDLE and type following codes in Python terminal.
7. >>> import cv2
8. >>> print cv2.\_\_version\_\_

If the results are printed out without any errors, congratulations !!! You have installed OpenCV-Python successfully.

## Building OpenCV from source

1. Download and install Visual Studio and CMake.

1.1. [Visual Studio 2012](http://go.microsoft.com/?linkid=9816768)

1.2. [CMake](http://www.cmake.org/files/v2.8/cmake-2.8.11.2-win32-x86.exe)

1. Download and install necessary Python packages to their default locations

2.1. [Python 2.7.x](http://python.org/ftp/python/2.7.5/python-2.7.5.msi)

2.2. [Numpy](http://sourceforge.net/projects/numpy/files/NumPy/1.7.1/numpy-1.7.1-win32-superpack-python2.7.exe/download)

2.3. [Matplotlib](https://downloads.sourceforge.net/project/matplotlib/matplotlib/matplotlib-1.3.0/matplotlib-1.3.0.win32-py2.7.exe) (*Matplotlib is optional, but recommended since we use it a lot in our tutorials.*)

**Conclusion and Future Scope**

The aim of the project is to face recognition and face emotion by using deep learning technique. In this we proposed real time video surveillance, what human face expresses it in front of camera and they were recognising the face.

**Future scope**:

In future we have increase the accuracy rate based on facial expression.

**Chapter 9:**

**Reference**

[1] P. Póo Argüelles, “Parálisis cerebral infantil,” 2008.

[2] I. C. Valdovinos Lopez, “Evaluación diagnóstica del niño con parálisis cerebral en el tercer nivel de atención,” 2009.

[3] J. M. Valdez, Medicinaௗ: organo de la Sociedad Argentina de Investigación Clínica. La Sociedad.

[4] G. Sumathy and A. Renjith, “A survey of vision and speech stimulation for cerebral palsy rehabilitation,” in 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014, pp. 1315–1319.

[5] M. González, D. Mulet, E. Perez, C. Soria, and V. Mut, “Vision based interface: an alternative tool for children with cerebral palsy,” in 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology, 2010, pp. 5895–5898.

[6] H. Rahmati, O. M. Aamo, O. Stavdahl, R. Dragon, and L. Adde, “Videobased early cerebral palsy prediction using motion segmentation,” in 2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2014, pp. 3779–3783.

[7] M. Betke, J. Gips, and P. Fleming, “The Camera Mouse: visual tracking of body features to provide computer access for people with severe disabilities,” IEEE Trans. Neural Syst. Rehabil. Eng., vol. 10, no. 1, pp. 1–10, Mar. 2002.

[8] C. Stauffer and W. E. L. Grimson, “Learning patterns of activity using real-time tracking,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 8, pp. 747–757, 2000.