ABSTRACT

A key requirement for developing any innovative system in a computing environment is to integrate a sufficiently friendly interface with the average end user. Accurate design of such a user-centered interface, however, means more than just the ergonomics of the panels and displays. It also requires that designers precisely define what information to use and how, where, and when to use it. Facial expression as a natural, non-intrusive and efficient way of communication has been considered as one of the potential inputs of such interfaces.

The work of this thesis aims at designing a robust Facial Expression Recognition (FER) system by combining various techniques from computer vision and pattern recognition. Expression recognition is closely related to face recognition where a lot of research has been done and a vast array of algorithms has been introduced. FER can also be considered as a special case of a pattern recognition problem and many techniques are available. In the designing of an FER system, we can take advantage of these resources and use existing algorithms as building blocks of our system. So a major part of this work is to determine the optimal combination of algorithms. To do this, we first divide the system into 3 modules, i.e. Preprocessing, Feature Extraction and Classification, then for each of them some candidate methods are implemented, and eventually the optimal configuration is found by comparing the performance of different combinations.

Another issue that is of great interest to facial expression recognition systems designers is the classifier which is the core of the system. Conventional classification algorithms assume the image is a single variable function of a underlying class label. However this is not true in face recognition area where the appearance of the face is influenced by multiple factors: identity, expression, illumination and so on.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Project Objective:

The primary goal of this research is to design, implement and evaluate a novel facial expression recognition system using various statistical learning techniques. This goal will be realized through the following objectives:

1. System level design: In this stage, we'll be using existing techniques in related areas as building blocks to design our system.
   a) A facial expression recognition system usually consists of multiple components, each of which is responsible for one task. We first need to review the literature and decide the overall architecture of our system, *i.e.*, how many modules it has, the responsibility of each of them and how they should cooperate with each other.
   b) Implement and test various techniques for each module and find the best combination by comparing their accuracy, speed, and robustness.
2. Algorithm level design: Focus on the classifier which is the core of a recognition system, trying to design new algorithms which hopefully have better performance compared to existing ones.

1.2 Motivation:

In today‟s networked world the need to maintain security of information or physical property is becoming both increasingly important and increasingly difficult. In countries like Nepal the rate of crimes are increasing day by day. No automatic systems are there that can track person‟s activity. If we will be able to track Facial expressions of persons automatically then we can find the criminal easily since facial expressions changes doing different activities. So we decided to make a Facial Expression Recognition System. We are interested in this project after we went through few papers in this area. The papers were published as per their system creation and way of creating the system for accurate and reliable facial expression recognition system. As a result we are highly motivated to develop a system that recognizes facial expression and track one person‟s activity.

1.3 Problem Statement

Human emotions and intentions are expressed through facial expressions and deriving an efficient and effective feature is the fundamental component of facial expression system. Face recognition is important for the interpretation of facial expressions in applications such as intelligent, man-machine interface and communication, intelligent visual surveillance, teleconference and real-time animation from live motion images. The facial expressions are useful for efficient interaction Most research and system in facial expression recognition are limited to six basic expressions (joy, sad, anger, disgust, fear, surprise). It is found that it is insufficient to describe all facial expressions and these expressions are

categorized based on facial actions.Detecting face and recognizing the facial expression is a very complicated task when it is a vital to pay attention to primary components like: face configuration, orientation, location where the face is set.

1.4. Problem Definition :

Human facial expressions can be easily classified into 7 basic emotions: happy, sad, surprise, fear, anger, disgust, and neutral. Our facial emotions are expressed through activation of specific sets of facial muscles. These sometimes subtle, yet complex, signals in an expression often contain an abundant amount of information about our state of mind. Through facial emotion recognition, we are able to measure the effects that content and services have on the audience/users through an easy and low-cost procedure. For example, retailers may use these metrics to evaluate customer interest. Health care providers can provide better service by using additional information about patient‟s emotional state during treatment. Entertainment producers can monitor audience engagement in events to consistently create desired content.

Humans are well - trained in reading the emotions of others, in fact, at just 14 months old, babies can already tell the difference between happy and sad. But can computers do a better job than us in accessing emotional states? To answer the question, We designed a deep learning neural.

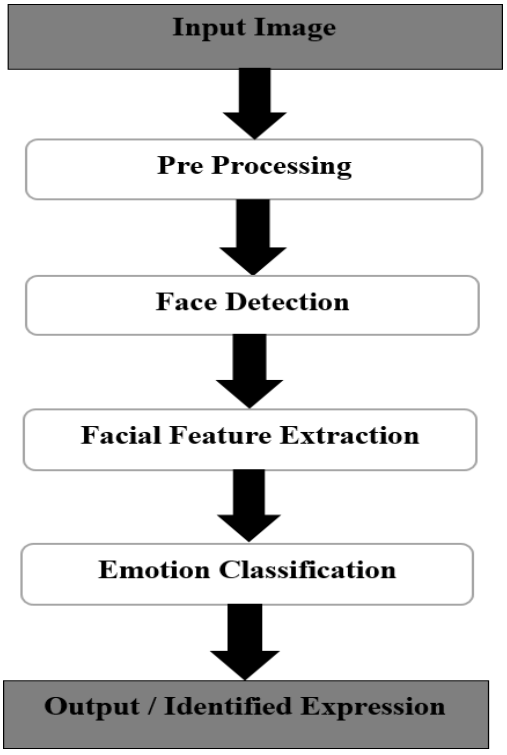

Fig1.1: Flow Chart

CHAPTER 2

LITERATURE STUDY

2.1 Introduction:

As per various literature surveys it is found that for implementing this project four basic steps are required to be performed.

i. Preprocessing

ii. Face registration

iii. Facial feature extraction

iv. Emotion classification

Description about all these processes are given below-

Preprocessing :

Preprocessing is a common name for operations with images at the lowest level of abstraction both input and output are intensity images. Most preprocessing steps that are implemented are –

a. Reduce the noise

b. Convert The Image To Binary/Grayscale.

c. Pixel Brightness Transformation.

d. Geometric Transformation.

As per various literature surveys it is found that for implementing this project four basic steps are required to be performed.

i. **Preprocessing**

ii. Face registration

iii. Facial feature extraction

iv. Emotion classification

Description about all these processes are given below-

2.2 Face Registration :

Face Registration is a computer technology being used in a variety of applications that identifies human faces in digital images. In this face registration step, faces are first located in the image using some set of landmark points called “face localization” or “face detection”. These detected faces are then geometrically normalized to match some template image in a process called “face registration”.

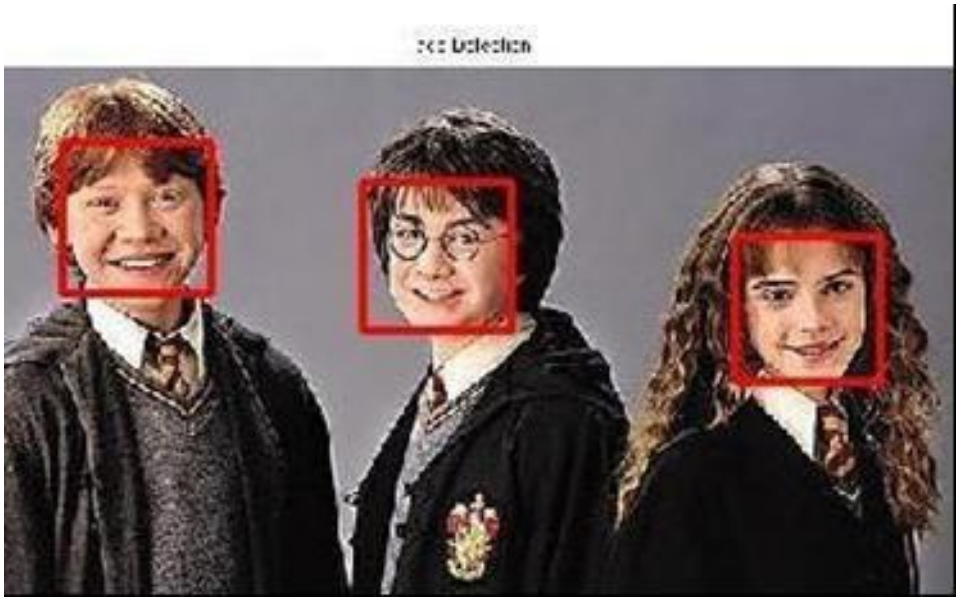

fig.2.1

2.3 Facial Feature Extraction :

Facial Features extraction is an important step in face recognition and is defined as the process of locating specific regions, points, landmarks, or curves/contours in a given 2-D image or a 3D range image. In this feature extraction step, a numerical feature vector is generated from the resulting registered image. Common features that can be extracted area.

Lips

b. Eyes

c. Eyebrows

d. Nose tip

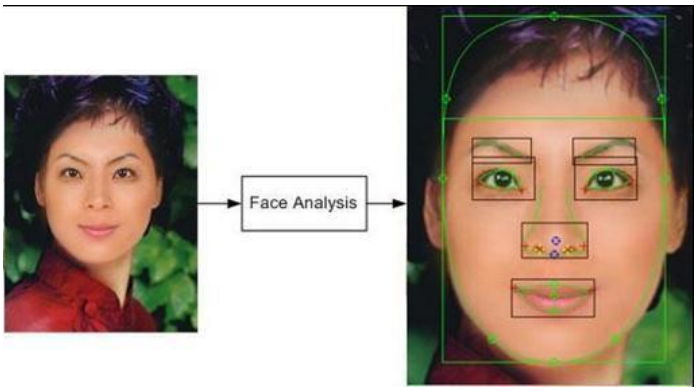

fig.2.2

2.4 Emotion Classification :

In the third step, of classification, the algorithm attempts to classify the given faces portraying one of the seven basic emotions.

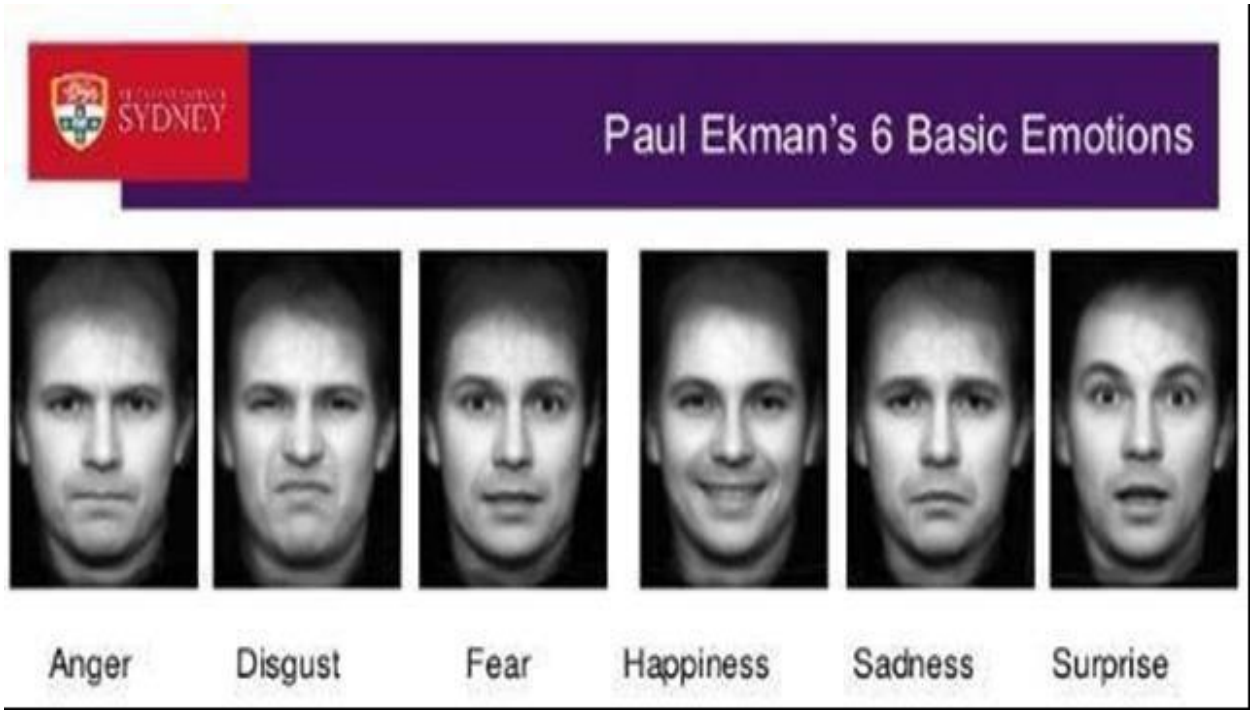

fig.2.3

CHAPTER 3

IMAGE PROCESSING

3.1 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 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 application 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 include Personal identification and Access control, Videophone and Teleconferencing, Forensic application, Human-Computer Interaction, Automated Surveillance, Cosmetology and so on.

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 class such as :

I. Neutral, **II. Angry, III. Disgust IV. Fear, V. Happy, VI . Sadness , VII . Surprise**

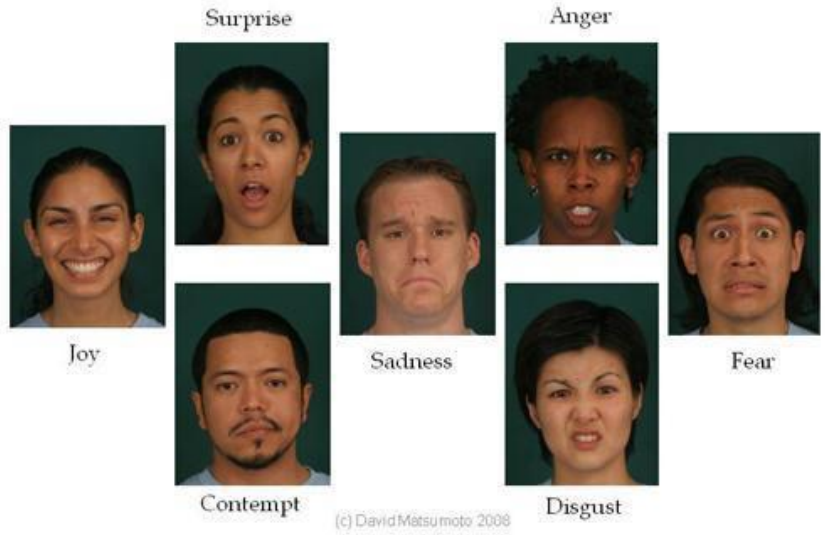

fig.3.1

3.2 DIGITAL IMAGE PROCESSING

Computerized picture preparing is a range portrayed by the requirement for broad test work to build up the practicality of proposed answers for a given issue. A critical trademark hidden the plan of picture preparing frameworks is the huge level of testing and experimentation that

Typically is required before touching base at a satisfactory arrangement. This trademark infors that the capacity to plan approaches and rapidly model hopeful arrangements by and large assumes a noteworthy part in diminishing the cost and time required to land at a suitableframework execution.

WHAT IS DIP?

A picture might be characterized as a two-dimensional capacity f(x, y), where x, y are spatial directions, and the adequacy off at any combine of directions (x, y) is known as the power or dark level of the picture by then. Whenever x, y and the abundance estimation of are all limited discrete amounts, we call the picture a computerized picture. The field of DIP alludes to preparing advanced picture by methods for computerized PC. Advanced picture is made out of a limited number of components, each of which has a specific area and esteem. The components are called pixels.

Vision is the most progressive of our sensor, so it is not amazing that picture play the absolute most imperative part in human observation. Be that as it may, dissimilar to people, who are constrained to the visual band of the EM range imaging machines cover practically the whole EM range, going from gamma to radio waves. They can work likewise on pictures produced by sources that people are not acclimated to partner with picture.

There is no broad understanding among creators in regards to where picture handling stops and other related territories, for example, picture examination and PC vision begin. Now and then a qualification is made by characterizing picture handling as a teach in which both the info and yield at a procedure are pictures. This is constraining and to some degree manufactured limit. The range of picture investigation is in the middle of picture preparing and PC vision.

There are no obvious limits in the continuum from picture handling toward one side to finish vision at the other. In any case, one helpful worldview is to consider three sorts of mechanized procedures in this continuum: low, mid and abnormal state forms. Low-level process includes primitive operations, for example, picture preparing to decrease commotion differentiate upgrade and picture honing.

A low-level process is described by the way that both its sources of info and yields are pictures.

Mid-level process on pictures includes assignments, for example, division, depiction of that Question diminish them to a frame reasonable for PC handling and characterization of individualarticles

A mid-level process is portrayed by the way that its sources of info by and large are pictures however its yields are properties removed from those pictures. At long last more clevated amount handling includes "Understanding an outlet of perceived items, as in picture examination and at the farthest end of the continuum playing out the intellectual capacities typically connected with human vision. Advanced picture handling, as effectively characterized is utilized effectively in a wide scope of regions of outstanding social and monetary esteem.

WHAT IS AN IMAGE?

A picture is spoken to as a two dimensional capacity f(x, y) where x and y are spatial co-ordinates and the adequacy of "T" at any match of directions (x, y) is known as the power of the picture by then.

Fig. 3.2 digital image

Processing on image:

Processing on image can be of three types They are low-level, mid-level, high level.

Low-level Processing:

- Preprocessing to remove noise.
- Contrast enhancement.
- Image sharpening.

Medium Level Processing :

- Segmentation.
- Edge detection
- Object extraction.

High Level Processing:

- Image analysis
- Scene interpretation

Why Image Processing?

Since the digital image is invisible, it must be prepared for viewing on one or more output device(laser printer, monitor at).The digital image can be optimized for the application by enhancing the appearance of the structures within it.

There are three of image processing used. They are

- Image to Image transformation
- Image to Information transformations
- Information to Image transformations

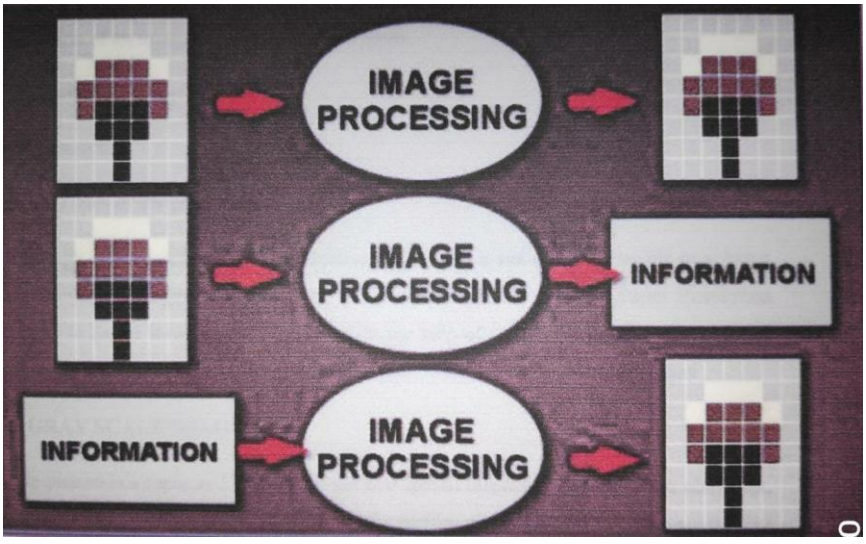

Fig. 3.3 Types of Image Processing

Pixel :

Pixel is the smallest element of an image. Each pixel correspond to any one value. In an 8-bit gray scale image, the value of the pixel between 0 and 255.Each pixel store a value proportional to the light intensity at that particular location. It is indicated in either Pixels per inch or Dots per inch.

Resolution :

The resolution can be defined in many ways. Such as pixel resolution, spatial resolution, temporal resolution, spectral resolution.In pixel resolution, the term resolution refers to the total number of count of pixels in an digital image. For example, If an image has M rows and N columns, then its resolution can be defined as MX N. Higher is the pixel resolution, the higher is the quality of the image.

Resolution of an image is of generally two types.

- Low Resolution image
- High Resolution

Since high resolution is not a cost effective process It is not always possible to achieve high resolution images with low cost. Hence it is desirable Imaging. In Super Resolution imaging, with the help of certain methods and algorithms we can be able to produce high resolution images from the low resolution image from the low resolution images.

3.2.1 GRAY SCALE IMAGE

A gray scale picture is a capacity I (xylem) of the two spatial directions of the picture plane. I(x,y) is the force of the picture force of picture at the point (x, y) on the picture plane. I (xylem) take non-negative expect the picture is limited by a rectangle

3.2.2 COLOR IMAGE

It can be spoken to by three capacities, R (xylem) for red, G (xylem) for green and B (xylem) for blue. A picture might be persistent as for the x and y facilitates and furthermore in adequacy. Changing over such a picture to advanced shape requires that the directions and the adequacy to be digitized. Digitizing the facilitate‟s esteems is called inspecting. Digitizing the adequacy esteems is called quantization.

3.3 RELATED TECHNOLOGY:

R-CNN

R-CNN is a progressive visual object detection system that combines bottom-up region proposals with rich options computed by a convolution neural network.

R-CNN uses region proposal ways to initial generate potential bounding boxes in a picture and then run a classifier on these proposed boxes.

SINGLE SIZE MULTI BOX DETECTOR

SSD discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At the time of prediction the network generates scores for the presence of each object category in each default box and generates adjustments to the box to better match the object shape.

Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes.

ALEXNET

AlexNet is a convolutional neural Network used for classification which has 5 Convolutional layers, 3 fullyconnected layers and 1 softmax layer with 1000 outputs for classification as his architecture.

YOLO

YOLO is real-time object detection. It applies one neural network to the complete image dividing the image into regions and predicts bounding boxes and possibilities for every region.

Predicted probabilities are the basis on which these bounding boxes are weighted. A single neural network predicts bounding boxes and class possibilities directly from full pictures in one evaluation. Since the full detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

VGG

VGG network is another convolution neural network architecture used for image classification.

MOBILENETS

To build lightweight deep neural networks MobileNets are used. It is based on a streamlined architecture that uses depth-wise separable convolutions. MobileNet uses 3×3 depth-wise separable convolutions that uses between 8 times less computation than standard convolution at solely a little reduction accuracy. Applications and use cases including object detection, fine grain classification, face attributes and large scale-localization.

TENSOR FLOW

Tensor flow is an open source software library for high performance numerical computation. It allows simple deployment of computation across a range of platforms (CPUs, GPUs, TPUs) due to its versatile design also from desktops to clusters of servers to mobile and edge devices. Tensor flow was designed and developed by researchers and engineers from the Google Brain team at intervals Google‟s AI organization, it comes with robust support for machine learning and deep learning and the versatile numerical computation core is used across several alternative scientific domains.

To construct, train and deploy Object Detection Models TensorFlow is used that makes it easy and also it provides a collection of Detection Models pre-trained on the COCO dataset, the Kitti dataset, and the Open Images dataset. One among the numerous Detection Models is that the combination of Single Shot Detector (SSDs) and Mobile Nets architecture that is quick, efficient and doesn't need huge computational capability to accomplish the object Detection.

FACIAL RECOGNITION

“Deep Face” is a deep learning facial recognition system developed to identify human faces in a digital image. Designed and developed by a group of researchers in Facebook. Google also has its own facial recognition system in Google Photos, which automatically separates all the photos according to the person in the image.

There are various components involved in Facial Recognition or authors could say it focuses on various aspects like the eyes, nose, mouth and the eyebrows for recognizing a faces.

CHAPTER 4

DEEP LEARNING

4.1 INTRODUCTION

Deep learning is a machine learning technique. It teaches a computer to filter inputs through layers to learn how to predict and classify information. Observations can be in the form of images, text, or sound. The inspiration for deep learning is the way that the human brain filters information. Its purpose is to mimic how the human brain works to create some real magic. In the human brain, there are about 100 billion neurons. Each neuron connects to about 100,000 of its neighbors. We‟re kind of recreating that, but in a way and at a level that works for machines. In our brains, a neuron has a body, dendrites, and an axon. The signal from one neuron travels down the axon and transfers to the dendrites of the next neuron. That connection where the signal passes is called a synapse. Neurons by themselves are kind of useless. But when you have lots of them, they work together to create some serious magic. That‟s the idea behind a deep learning algorithm! You get input from observation and you put your input into one layer. That layer creates an output which in turn becomes the input for the next layer, and so on. This happens over and over until your final output signal! The neuron (**node**) gets a signal or signals (**input values**), which pass through the neuron. That neuron delivers the **output signal**.

Think of the input layer as your senses: the things you see, smell, and feel, for example. These are independent variables for one single observation. This information is broken down into numbers and the bits of binary data that a computer can use. You‟ll need to either standardize or normalize these variables so that they‟re within the same range. They use many layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output of the previous layer for its input. What they learn forms a hierarchy of concepts. In this hierarchy, each level learns to transform its input data into a more and more abstract and composite representation. That means that for an image, for example, the input might be a matrix of pixels. The first layer might encode the edges and compose the pixels. The next layer might compose an arrangement of edges. The next layer might encode a nose and eyes. The next layer might recognize that the image contains a face, and so on.

What happens inside the neuron?

The input node takes in information in a numerical form. The information is presented as an activation value where each node is given a number. The higher the number, the greater the activation. Based on the connection strength (weights) and transfer function, the activation value passes to the next node. Each of the nodes sums the activation values that it receives (it calculates the **weighted sum**) and modifies that sum based on its transfer function. Next, it applies an activation function. An activation

function is a function that‟s applied to this particular neuron. From that, the neuron understands if it needs to pass along a signal or not.

Each of the synapses gets assigned weights, which are crucial to **Artificial Neural Networks** (ANNs). Weights are how ANNs learn. By adjusting the weights, the ANN decides to what extent signals get passed along. When you‟re training your network, you‟re deciding how the weights are adjusted.

The activation runs through the network until it reaches the output nodes. The output nodes then give us the information in a way that we can understand. Your network will use a cost function to compare the output and the actual expected output. The model performance is evaluated by the cost function. It‟s expressed as the difference between the actual value and the predicted value.

There are many different cost functions you can use, you‟re looking at what the error you have in your network is. You‟re working to minimize loss function. (In essence, the lower the loss function, the closer it is to your desired output). The information goes back, and the neural network begins to learn with the goal of minimizing the cost function by tweaking the weights. This process is called **backpropagation**.

In **forward propagation**, information is entered into the input layer and propagates forward through the network to get our output values. We compare the values to our expected results. Next, we calculate the errors and propagate the info backward. This allows us to train the network and update the weights. (Backpropagation allows us to adjust all the weights simultaneously.) During this process, because of the way the algorithm is structured, you‟re able to adjust all of the weights simultaneously. This allows you to see which part of the error each of your weights in the neural network is responsible for.

When you‟ve adjusted the weights to the optimal level, you‟re ready to proceed to the testing phase!

How does an artificial neural network learn?

There are two different approaches to get a program to do what you want. First, there‟s the specifically guided and hard-programmed approach. You tell the program exactly what you want it to do. Then there are **neural networks**. In neural networks, you tell your network the inputs and what you want for the outputs, and then you let it learn on its own.

By allowing the network to learn on its own, you can avoid the necessity of entering in all of the rules. You can create the architecture and then let it go and learn. Once it‟s trained up, you can give it a new image and it will be able to distinguish output.

4.2 Feedforward and feedback networks

A **feedforward** network is a network that contains inputs, outputs, and hidden layers. The signals can only travel in one direction (forward). Input data passes into a layer where calculations are performed. Each processing element computes based upon the weighted sum of its inputs. The new values become the new input values that feed the next layer (feed-forward). This continues through all the layers and determines the output. Feedforward networks are often used in, for example, data mining.

A **feedback network** (for example, a recurrent neural network) has feedback paths. This means that they can have signals traveling in both directions using loops. All possible connections between neurons are allowed. Since loops are present in this type of network, it becomes a non-linear dynamic system which changes continuously until it reaches a state of equilibrium. Feedback networks are often used in optimization problems where the network looks for the best arrangement of interconnected factors.

4.3 Weighted Sum

Inputs to a neuron can either be features from a training set or outputs from the neurons of a previous layer. Each connection between two neurons has a unique synapse with a unique weight attached. If you want to get from one neuron to the next, you have to travel along the synapse and pay the “toll” (weight). The neuron then applies an activation function to the sum of the weighted inputs from each incoming synapse. It passes the result on to all the neurons in the next layer. When we talk about updating weights in a network, we‟re talking about adjusting the weights on these synapses.

A neuron‟s input is the sum of weighted outputs from all the neurons in the previous layer. Each input is multiplied by the weight associated with the synapse connecting the input to the current neuron. If there are 3 inputs or neurons in the previous layer, each neuron in the current layer will have 3 distinct weights: one for each synapse.

In a nutshell, the activation function of a node defines the output of that node.

The activation function (or transfer function) translates the input signals to output signals. It maps the output values on a range like 0 to 1 or -1 to 1. It‟s an abstraction that represents the rate of action potential firing in the cell. It‟s a number that represents the likelihood that the cell will fire. At it‟s simplest, the function is binary: **yes** (the neuron fires) or **no** (the neuron doesn‟t fire). The output can be either 0 or 1 (on/off or yes/no), or it can be anywhere in a range. If you were using a function that maps a range between 0 and 1 to determine the likelihood that an image is a cat, for example, an output of 0.9 would show a 90% probability that your image is, in fact, a cat.

4.4 Activation function

In a nutshell, the activation function of a node defines the output of that node.

The activation function (or transfer function) translates the input signals to output signals. It maps the output values on a range like 0 to 1 or -1 to 1. It‟s an abstraction that represents the rate of action potential firing in the cell. It‟s a number that represents the likelihood that the cell will fire. At it‟s simplest, the function is binary: **yes** (the neuron fires) or **no** (the neuron doesn‟t fire). The output can be either 0 or 1 (on/off or yes/no), or it can be anywhere in a range.

What options do we have? There are many activation functions, but these are the four very common ones:

Thresholdfunction

This is a step function. If the summed value of the input reaches a certain threshold the function passes on 0. If it‟s equal to or more than zero, then it would pass on 1. It‟s a very rigid, straightforward, yes or no function.

Sigmoid function

This function is used in logistic regression. Unlike the threshold function, it‟s a smooth, gradual progression from 0 to 1. It‟s useful in the output layer and is used heavily for linear regression.

Hyperbolic Tangent Function

This function is very similar to the sigmoid function. But unlike the sigmoid function which goes from 0 to 1, the value goes below zero, from -1 to 1. Even though this isn‟t a lot like what happens in a brain,

this function gives better results when it comes to training neural networks. Neural networks sometimes get “stuck” during training with the sigmoid function. This happens when there‟s a lot of strongly negative input that keeps the output near zero, which messes with the learning process.

Rectifier function

This might be the most popular activation function in the universe of neural networks. It‟s the most efficient and biologically plausible. Even though it has a kink, it‟s smooth and gradual after the kink at 0. This means, for example, that your output would be either “no” or a percentage of “yes.” This function doesn‟t require normalization or other complicated calculations.

The field of artificial intelligence is essential when machines can do tasks that typically require human intelligence. It comes under the layer of machine learning, where machines can acquire skills and learn from past experience without any involvement of human. Deep learning comes under machine learning where artificial neural networks, algorithms inspired by the human brain, learn from large amounts of data. The concept of deep learning is based on humans‟ experiences; the deep learning algorithm would perform a task continuously so that it can improve the outcome. Neural networks have various (deep) layers that enable learning. Any drawback that needs “thought” to work out could be a drawback deep learning can learn to unravel.

CHAPTER 5

CONVOLUTIONAL NEURAL NETWORKS

5.1 INTRODUCTION TO CONVOLUTIONAL NEURAL NETWORKS (CNN)

Artificial Neural Networks

The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living neurons and dendrites.

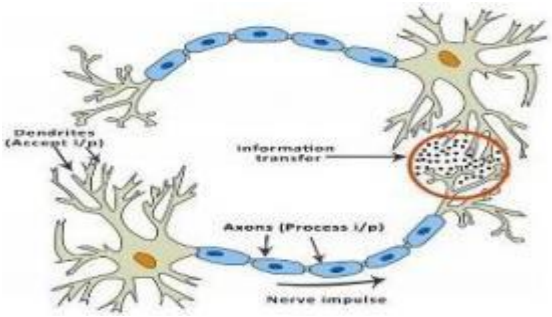

fig: 5.1

The human brain is composed of 86 billion nerve cells called neurons. They are connected to other thousand cells by Axons. Stimuli from external environment or inputs from sensory organs are accepted by dendrites. These inputs create electric impulses, which quickly travel through the neural network. A neuron can then send the message to other neuron to handle the issue or does not send it forward.

ANNs are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value. Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values.

Neural network:

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. Thus a neural network is either a biological neural network, made up of real biological neurons, or an artificial neural network, for solving artificial intelligence (AI) problem. The connections of the biological neuron are modeled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred as a linear

combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be - 1 and 1.

These artificial networks may be used for predictive modeling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks,

which can derive conclusions from a complex and seemingly unrelated set of information.

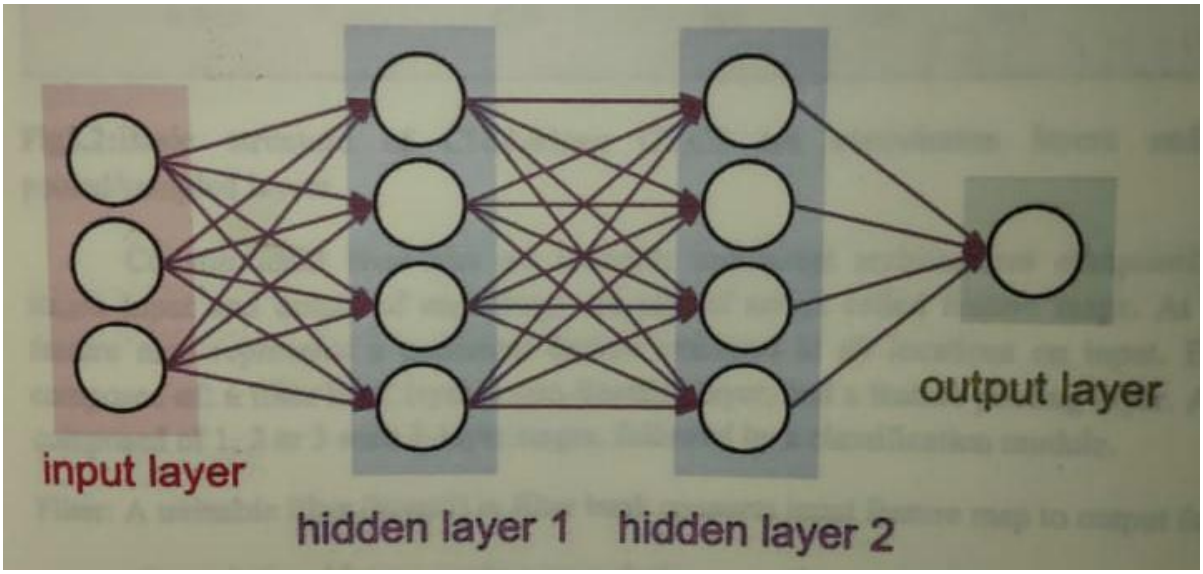

fig. 5.2 A simple neural network

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship.

5.2 CONVOLUTIONAL NEURAL NETWORKS:

Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity.

Convolutional Neural Networks (CNNs) are analogous to traditional ANNs in that they are comprised of neurons that self-optimise through learning. Each neuron will still receive an input and perform a operation (such as a scalar product followed by a non-linear function) - the basis of countless ANNs. From the input raw image vectors to the final output of the class score, the entire of the network will still express a single

perceptive score function (the weight). The last layer will contain loss functions associated with the classes, and all of the regular tips and tricks developed for traditional ANNs still apply.

The only notable difference between CNNs and traditional ANNs is that CNNs are primarily used in the field of pattern recognition within images. This allows us to encode image-specific features into the architecture, making the network more suited for image-focused tasks - whilst further reducing the parameters required to set up the model. One of the largest limitations of traditional forms of ANN is that they tend to struggle with the computational complexity required to compute image data. Common machine learning benchmarking datasets such as the MNIST database of handwritten digits are suitable for most forms of ANN, due to its relatively small image dimensionality of just 28 × 28. With this dataset a single neuron in the first hidden layer will contain 784 weights (28×28×1 where 1 bare in mind that MNIST is normalised to just black and white values), which is manageable for most forms of ANN. If you consider a more substantial coloured image input of 64 × 64, the number of weights on just a single neuron of the first layer increases substantially to 12, 288. Also take into account that to deal with this scale of input, the network will also need to be a lot larger than one used to classify colour-normalised MNIST digits, then you will understand the drawbacks of using such models.

5.2.1 CNN ARCHITECTURE:

CNNs are feedforward networks in that information flow takes place in one direction only, from their inputs to their outputs. Just as artificial neural networks (ANN) are biologically inspired, so are CNNs. The visual cortex in the brain, which consists of alternating layers of simple and complex cells (Hubel & Wiesel, 1959, 1962), motivates their architecture.

CNN architectures come in several variations; however, in general, they consist of convolutional and pooling (or subsampling) layers, which are grouped into modules. Either one or more fully connected layers, as in a standard feedforward neural network, follow these modules. Modules are often stacked on top of each other to form a deep model. It illustrates typical CNN architecture for a toy image classification task. An image is input directly to the network, and this is followed by several stages of convolution and pooling. Thereafter, representations from these operations feed one or more fully connected layers.

Finally, the last fully connected layer outputs the class label. Despite this being the most popular base architecture found in the literature, several architecture changes have been proposed in recent years with the objective of improving image classification accuracy or reducing computation costs. Although for the remainder of this section, we merely fleetingly introduce standard CNN architecture.

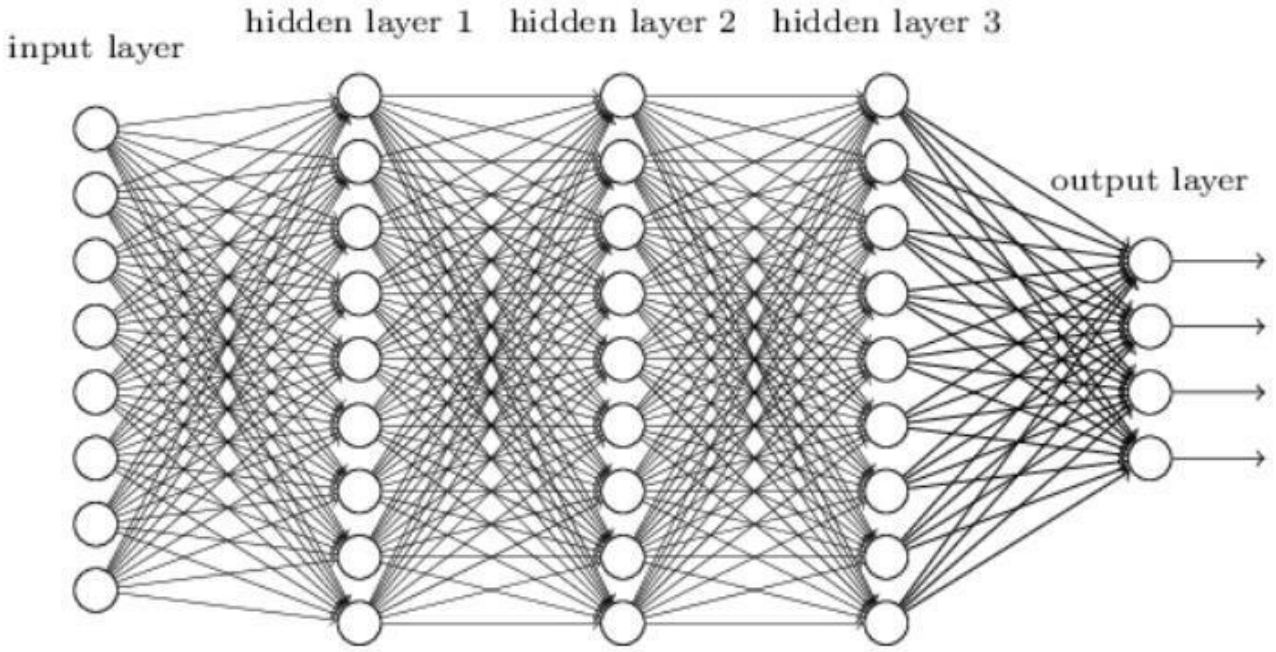

fig: 5.3

5.3 OVERALL ARCHITECTURE:

CNNs are comprised of three types of layers. These are convolutional layers, pooling layers and fully-connected layers. When these layers are stacked, a CNN architecture has been formed. A simplified CNN architecture for MNIST classification is illustrated in Figure 2. input 0 9 convolution w/ReLu pooling output fully-connected w/ ReLu fully-connected ... Fig. 2: An simple CNN architecture, comprised of just five layers The basic functionality of the example CNN above can be broken down into four key areas. 1. As found in other forms of ANN, the input layer will hold the pixel values of the image. 2. The convolutional layer will determine the output of neurons of which are connected to local regions of the input through the calculation of the scalar product between their weights and the region connected to the input volume. The rectified linear unit (commonly shortened to ReLu) aims to apply an "elementwise" activation function such as sigmoid to the output of the activation produced by the previous layer. 3. The pooling layer will then simply perform downsampling along the spatial dimensionality of the given input, further reducing the number of parameters within that activation. 4. The fully-connected layers will then perform the same duties found in standard ANNs and attempt to produce class scores from the activations, to be used for classification. It is also suggested that ReLu may be used between these layers, as to improve performance. Through this simple method of transformation, CNNs are able to transform the original input layer by layer using convolutional and downsampling techniques to produce class scores for classification and regression purposes.

However, it is important to note that simply understanding the overall architecture of a CNN architecture will not suffice. The creation and optimisation of these models can take quite some time, and can be quite confusing. We will now explore in detail the individual layers, detailing their hyperparameters and connectivities.

5.4 CONVOLUTIONAL LAYERS:

The convolutional layers serve as feature extractors, and thus they learn the feature representations of their input images. The neurons in the convolutional layers are arranged into feature maps. Each neuron in a feature map has a receptive field, which is connected to a neighborhood of neurons in the previous layer via a set of trainable weights, sometimes referred to as a filter bank. Inputs are convolved with the learned weights in order to compute a new feature map, and the convolved results are sent through a nonlinear activation function.

All neurons within a feature map have weights that are constrained to be equal; however, different feature maps within the same convolutional layer have different weights so that several features can be extracted at each location.

As the name implies, the convolutional layer plays a vital role in how CNNs operate. The layers parameters focus around the use of learnable kernels.

These kernels are usually small in spatial dimensionality, but spreads along the entirety of the depth of the input. When the data hits a convolutional layer, the layer convolves each filter across the spatial dimensionality of the input to produce a 2D activation map. These activation maps can be visualised.

As we glide through the input, the scalar product is calculated for each value in that kernel. From this the network will learn kernels that ‟fire‟ when they see a specific feature at a given spatial position ofthe input. These are commonly known as activations.

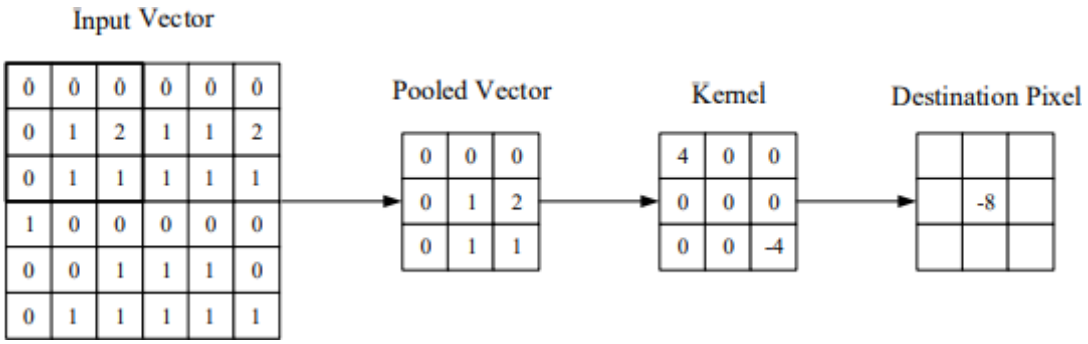

fig: 5.4 Visual representation of a convolutional layer

The centre element of the kernel is placed over the input vector, of which is then calculated and replaced with a weighted sum of itself and any nearby pixels.

Every kernel will have a corresponding activation map, of which will be stacked along the depth dimension to form the full output volume from the convolutional layer.

As we alluded to earlier, training ANNs on inputs such as images results in models of which are too big to train effectively. This comes down to the fullyconnected manner of standard ANN neurons, so to mitigate against this every neuron in a convolutional layer is only connected to small region of the input volume. The dimensionality of this region is commonly referred to as the receptive field size of the neuron. The magnitude of the connectivity through the depth is nearly always equal to the depth of the input.

For example, if the input to the network is an image of size 64 × 64 × 3 (aRGBcoloured image with a dimensionality of 64 × 64) and we set the receptive field size as 6 × 6, we would have a total of 108 weights on each neuron within the convolutional layer. (6 × 6 × 3 where 3 is the magnitude of connectivity across the depth of the volume) To put this into perspective, a standard neuron seen in other forms of ANN would contain 12, 288 weights each.

Convolutional layers are also able to significantly reduce the complexity of the model through the optimisation of its output. These are optimised through three hyperparameters, the depth, the stride and setting zero-padding.

The depth of the output volume produced by the convolutional layers can be manually set through the number of neurons within the layer to a the same region of the input. This can be seen with other forms of ANNs, where the all of the neurons in the hidden layer are directly connected to every single neuron beforehand. Reducing this hyperparameter can significantly minimise the total number of neurons of the network, but it can also significantly reduce the pattern recognition capabilities of the model.

We are also able to define the stride in which we set the depth around the spatial dimensionality of the input in order to place the receptive field. For example if we were to set a stride as 1, then we would have a heavily overlapped receptive field producing extremely large activations. Alternatively, setting the stride to a greater number will reduce the amount of overlapping and produce an output of lower spatial dimensions.

Zero-padding is the simple process of padding the border of the input, and is an effective method to give further control as to the dimensionality of the output volumes.

It is important to understand that through using these techniques, we will alter the spatial dimensionality of the convolutional layers output.

Despite our best efforts so far we will still find that our models are still enormous if we use an image input of any real dimensionality. However, methods have been developed as to greatly curtail the overall number of parameters within the convolutional layer.

Parameter sharing works on the assumption that if one region feature is useful to compute at a set spatial region, then it is likely to be useful in another region. If we constrain each individual activation map within the output volume to the same weights and bias, then we will see a massive reduction in the number of parameters being produced by the convolutional layer.

As a result of this as the backpropagation stage occurs, each neuron in the output will represent the overall gradient of which can be totalled across the depth - thus only updating a single set of weights, as opposed to every single one.

Pooling Layers

The purpose of the pooling layers is to reduce the spatial resolution of the feature maps and thus achieve spatial invariance to input distortions and translations. Initially, it was common practice to use average pooling aggregation layers to propagate the average of all the input values, of a small neighbourhood of an image to the next layer. However, in more recent models, max pooling aggregation layers propagate the maximum value within a receptive field to the next layer.

Pooling layers aim to gradually reduce the dimensionality of the representation, and thus further reduce the number of parameters and the computational complexity of the model.

The pooling layer operates over each activation map in the input, and scales its dimensionality using the “MAX” function. In most CNNs, these come in the form of max-pooling layers with kernels of a

dimensionality of 2 × 2 applied with a stride of 2 along the spatial dimensions of the input. This scales the activation map down to 25% of the original size - whilst maintaining the depth volume to its standard size.

Due to the destructive nature of the pooling layer, there are only two generally observed methods of max-pooling. Usually, the stride and filters of the pooling layers are both set to 2 × 2, which will allow the layer to extend through the entirety of the spatial dimensionality of the input. Furthermore overlapping pooling may be utilised, where the stride is set to 2 with a kernel size set to 3. Due to the destructive nature of pooling, having a kernel size above 3 will usually greatly decrease the performance of the model.

It is also important to understand that beyond max-pooling, CNN architectures may contain general-pooling. General pooling layers are comprised of pooling neurons that are able to perform a multitude of common operations including L1/L2-normalisation, and average pooling. However, this tutorial will primarily focus on the use of max-pooling.

Fully Connected Layers

Several convolutional and pooling layers are usually stacked on top of each other to extract more abstract feature representations in moving through the network. The fully connected layers that follow these layers interpret these feature representations and perform the function of high-level reasoning. . For classification problems, it is standard to use the softmax operator on top of a DCNN. While early success was enjoyed by using radial basis functions (RBFs), as the classifier on top of the convolutional towers found that replacing the softmax operator with a support vector machine (SVM) leads to improved classification accuracy.

The fully-connected layer contains neurons of which are directly connected to the neurons in the two adjacent layers, without being connected to any layers within them. This is analogous to way that neurons are arranged in traditional forms of ANN.

Despite the relatively small number of layers required to form a CNN, there is no set way of formulating a CNN architecture. That being said, it would be idiotic to simply throw a few of layers together and expect it to work. Through reading of related literature it is obvious that much like other forms of ANNs, CNNs tend to follow a common architecture. This common architecture is illustrated in Figure 2, where convolutional layers are stacked, followed by pooling layers in a repeated manner before feeding forward to fully-connected layers.

Convolutional Neural Networks differ to other forms of Artifical Neural Network in that instead of focusing on the entirety of the problem domain, knowledge about the specific type of input is exploited. This in turn allows for a much simpler network architecture to be set up.

This paper has outlined the basic concepts of Convolutional Neural Networks, explaining the layers required to build one and detailing how best to structure the network in most image analysis tasks.

Research in the field of image analysis using neural networks has somewhat slowed in recent times. This is partly due to the incorrect belief surrounding the level of complexity and knowledge required to begin modelling these superbly powerful machine learning algorithms. The authors hope that this paper has in some way reduced this confusion, and made the field more accessible to beginners.

Training

CNNs and ANN in general use learning algorithms to adjust their free parameters in order to attain the desired network output. The most common algorithm used for this purpose is backpropagation. Backpropagation computes the gradient of an objective function to determine how to adjust a network‟s parameters in order to minimize errors that affect performance. A commonly experienced problem with training CNNs, and in particular DCNNs, is overfitting, which is poor performance on a held-out test set after the network is trained on a small or even large training set. This affects the model‟s ability to generalize on unseen data and is a major challenge for DCNNs that can be assuaged by regularization.

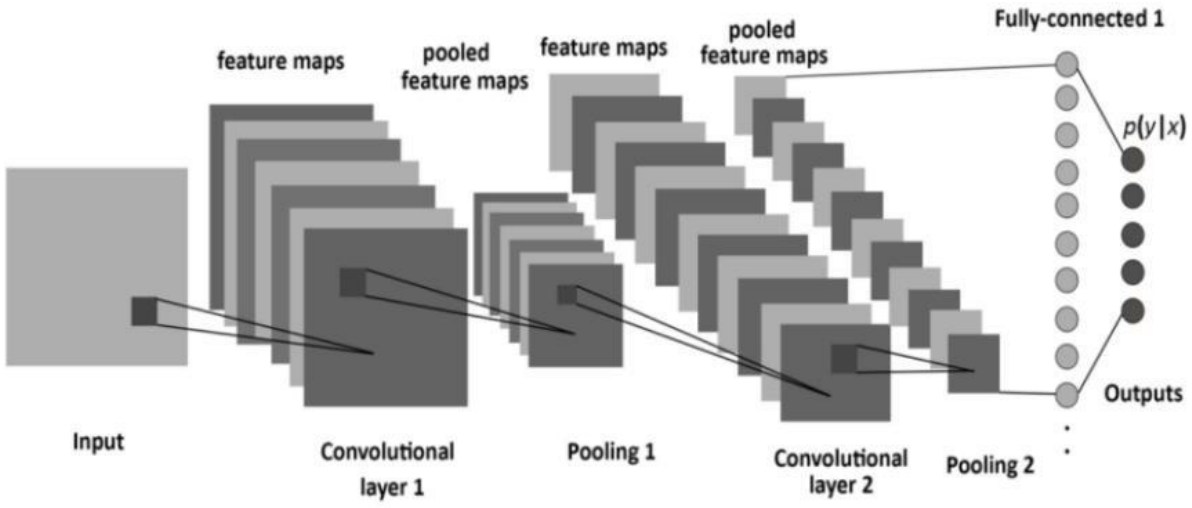

fig: 5.5

Caffe Model

Caffe is a framework of Deep Learning and it was made used for the implementation and to access the following things in an object detection system.

•Expression: Models and optimizations are defined as plaintext schemas in the caffe model unlike others which use codes for this purpose.

•Speed: for research and industry alike speed is crucial for state-of-the-art models and massive data [11].

•Modularity: Flexibility and extension is majorly required for the new tasks and different settings.

•Openness: Common code, reference models, and reproducibility are the basic requirements of scientific and applied progress.

Types of Caffe Models

Open Pose

The first real-time multi-person system is portrayed by OpenPose which can collectively sight human body, hand, and facial keypoints (in total 130 keypoints) on single pictures.

Fully Convolutional Networks for Semantic Segmentation

In the absolutely convolutional networks (FCNs) Fully Convolutional Networks are the reference implementation of the models and code for the within the PAMI FCN and CVPR FCN papers.

Cnn-vis

Cnn-vis is an open-source tool that lets you use convolutional neural networks to generate images. It has taken inspiration from the Google's recent Inceptionism blog post.

Speech Recognition

Speech Recognition with the caffe deep learning framework.

DeconvNet

Learning Deconvolution Network for Semantic Segmentation.

Coupled Face Generation

This is the open source repository for the Coupled Generative Adversarial Network (CoupledGAN or CoGAN) work.These models are compatible with Caffe master, unlike earlier FCNs that required a pre-release branch (note: this reference edition of the models remains ongoing and not all of the models have yet been ported to master).

Codes for Fast Image Retrieval

To create the hash-like binary codes it provides effective framework for fast image retrieval.

SegNet and Bayesian SegNet

SegNet is real-time semantic segmentation architecture for scene understanding.

Deep Hand

It gives pre-trained CNN models.

DeepYeast

Deep Yeast may be an 11-layer convolutional neural network trained on biaural research pictures of yeast cells carrying fluorescent proteins with totally different subcellular localizations.

Python VS other languages for Object Detection: Object detection may be a domain-specific variation of the machine learning prediction drawback. Intel‟s OpenCV library that is implemented in C/C++ has its interfaces offered during a} very vary of programming environments like C#, Matlab, Octave, R, Python and then on. Why Python codes are much better option than other language codes for object detection are more compact and readable code.

Python uses zero-based indexing.

Dictionary (hashes) support provided.

Simple and elegant Object-oriented programming

Free and open

Multiple functions can be package in one module

More choices in graphics packages and toolsets Supervised learning also plays an important role.

The utility of unsupervised pre-training is usually evaluated on the premise of what performance is achieved when supervised fine-tuning. This paper reviews and discusses the fundamentals of learning

as well as supervised learning for classification models, and also talks about the mini batch stochastic gradient descent algorithm that is used to fine-tune many of the models.

Object Classification in Moving Object Detection Object classification works on the shape, motion, color and texture. The classification can be done under various categories like plants, objects, animals, humans etc. The key concept of object classification is tracking objects and analysing their features.

Shape-Based

A mixture of image-based and scene based object parameters such as image blob (binary large object) area, the as pectration of blob bounding box and camera zoom is given as input to this detection system. Classification is performed on the basis of the blob at each and every frame. The results are kept in the histogram.

Motion-Based

When an easy image is given as an input with no objects in motion, this classification isn't required. In general, non- rigid articulated human motion shows a periodic property; therefore this has been used as a powerful clue for classification of moving objects. based on this useful clue, human motion is distinguished from different objects motion. ColorBased- though color isn't an applicable live alone for police investigation and following objects, but the low process value of the colour primarily based algorithms makes the coloura awfully smart feature to be exploited. As an example, the color-histogram based technique is employed for detection of vehicles in period. Color bar chart describes the colour distribution in a very given region that is powerful against partial occlusions.

Texture-Based

The texture-based approaches with the assistance of texture pattern recognition work just like motion-based approaches. It provides higher accuracy, by exploitation overlapping native distinction social control however might need longer, which may be improved exploitation some quick techniques. I. proposed WORK Authors have applied period object detection exploitation deep learning and OpenCV to figure to work with video streams and video files. This will be accomplished using the highly efficient open computer vision. Implementation of proposed strategy includes caffe-model based on Google Image Scenery; Caffe offers the model definitions, optimization settings, pre-trained weights[4]. Prerequisite includes Python 3.7, OpenCV 4 packages and numpy to complete this task of object detection. NumPy is the elementary package for scientific computing with Python. It contains among other things: a strong N-dimensional array object, subtle (broadcasting) functions tools for integrating C/C++ and fortran code, helpful linear algebra, Fourier transform, and random number

capabilities. Numpy works in backend to provide statistical information of resemblance of object with the image scenery caffemodel database. Object clusters can be created according to fuzzy value provided by NumPy. This project can detect live objects from the videos and images.

LEARNING FEATURE HIERARCHY:

Learn hierarchy all the way from pixels classifier One layer extracts features from output of previous layer, train all layers jointly

Zero-One Loss

The models given in these deep learning tutorials are largely used for classification. The major aim of training a classifier is to reduce the amount of errors (zero-one loss) on unseen examples

Negative Log-Likelihood Loss

Optimizing it for large models (thousands or millions of parameters) is prohibitively expensive (computationally) because the zero-one loss isn't differentiable. In order to achieve this maximization of the log-likelihood is done on the classifier given all the labels in a training set [14].The likelihood of the correct class and number of right predictions is not the equal, but they are pretty similar from the point of view of a randomly initialized classifier. As the likelihood and zero-one loss are different objectives but we should always see that they are co-related on the validation set but sometimes one will rise while the other falls, or vice-versa.

Stochastic Gradient Descent

Ordinary gradient descent is an easy rule within which we repeatedly create tiny steps downward on an error surface defined by a loss function of some parameters. For the aim of normal gradient descent we take into account that the training data is rolled into the loss function. Then the pseudo code of this algorithm can be represented as Stochastic gradient descent (SGD) works according to similar principles as random gradient descent (SGD) operates on the basis of similar principles as normal gradient descent. It quickly proceeds by estimating the gradient from simply a few examples at a time instead of complete training set. In its purest kind, we use simply one example at a time to estimate the gradient.

Caffe is a deep learning framework or else we can say a library it's made with expression speed and modularity in mind they will put by Berkeley artificial intelligence research and created by young King Gia there are many deep learning or machine learning frameworks for computer vision like tensorflow ,Tiano, Charis and SVM[2]. But why exactly we implement edition cafe there as on is its expressive architecture we can easily switch between CPU and GPU while training on GPU machine

modules and optimization for Our problem is defined by configuration without hard coding. It supports extensible code since cafes are open source library. It is four foot by over twenty thous and developers and github since its birth it offers coding platform in extensible languages like Python and C++. The next reason is speed for training the neural networks speed is the primary constraint. Caffe can process over million images in a single day with the standard media GPU that is milliseconds per image. Whereas the same dataset of million images can take weeks for Tiana and Kara's Caffe is the fastest convolution neural network present community as mentioned earlier since its open source library huge number of research arepowered by cafe and every single day something new is coming out of it.

CHAPTER 6

SOFTWARE REQUIREMENT

6.1 Introduction

As the project is developed in python, we have used Anaconda for Python 3.6.5 and Spyder.

Anaconda

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

Spyder

Spyder (formerly Pydee) is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates NumPy, SciPy, Matplotlib and I Python, as well as other open source software. It is released under the MIT license.

Spyder is extensible with plug-in, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint and Rope. It is available cross-platform through Anaconda, on Windows with Win Python and Python (x,y), on macOS through MacPorts, and on major Linux distributions such as Arch Linux, Debian, Fedora, Gentoo Linux, openSUSE and Ubuntu.

Features include:

1. editor with syntax highlighting and introspection for code completion
2. support for multiple Python consoles (including I Python)
3. the ability to explore and edit variables from a GUI

Available plug-in include:

1. Static Code Analysis with Plinth
2. Code Profiling
3. Conda Package Manager with Condo

6.2 Interfaces

Hardware Interfaces

1. Processor : Intel CORE i5 processor with minimum 2.9 GHz
2. RAM : Minimum 4 GB.
3. Hard Disk : Minimum 500 GB

Software Interfaces

1. Microsoft Word 2003
2. Database Storage : Microsoft Excel
3. Operating System : Windows10

6.3. PLANNING :

The steps we followed while developing this project are-:

1. Analysis of the problem statement.
2. Gathering of the requirement specification
3. Analysation of the feasibility of the project.
4. Development of a general layout.
5. Going by the journals regarding the previous related works on this field.
6. Choosing the method for developing the algorithm.
7. Analyzing the various pros and cons.
8. Starting the development of the project
9. Installation of software like ANACONDA.
10. Developing an algorithm.
11. Analysation of algorithm by guide.
12. Coding as per the developed algorithm in PYTHON.

We developed this project as per the iterative waterfall model:

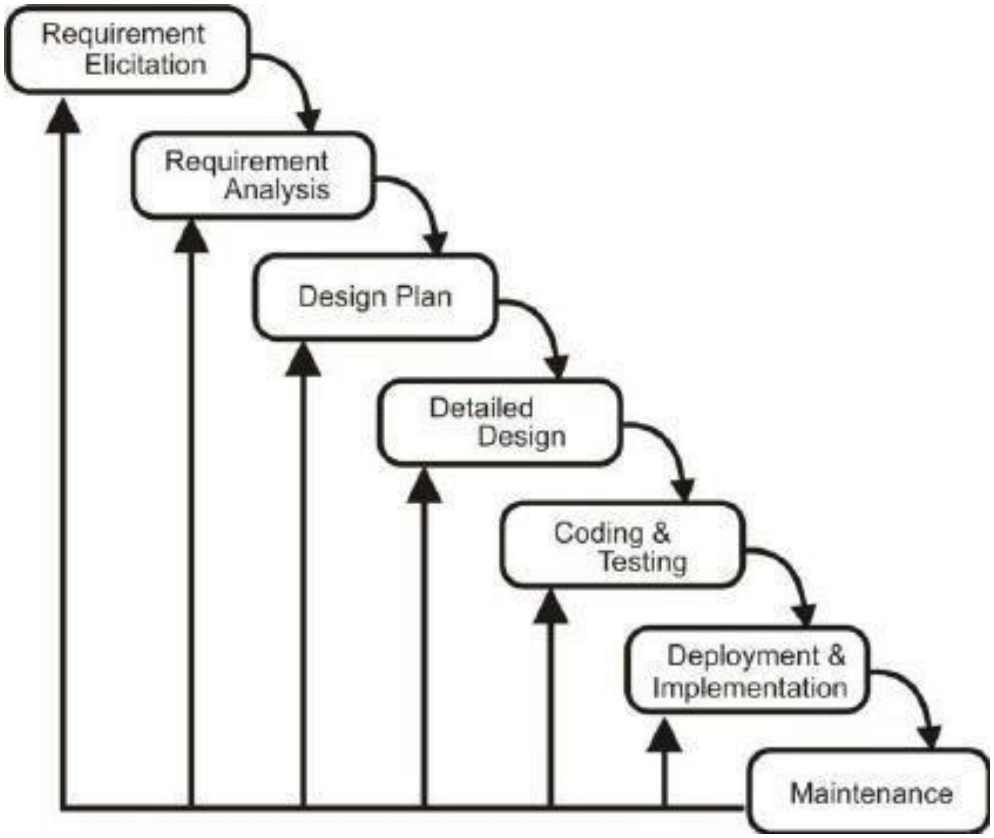

fig.6.1

6.4 The Library & Packages :

Open CV :

Open CV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. Open CV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, Open CV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red

eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

Open CV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. The library is used extensively in companies, research groups and by governmental bodies.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. Open CV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available.

A full-featured CUDA and Open CV interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those Algorithms. Open CV is written natively in C++ and has a template interface that works seamlessly with STL containers.

Open CV's application areas include :

1. 2D and 3D feature toolkits

2. Emotion estimation

3. Facial recognition system

4. Gesture recognition

5. Human–computer interaction (HCI)

6. Mobile robotics

7. Motion understanding

8. Object identification

9. Segmentation and recognition

10. Stereo sis stereo vision: depth perception from 2 cameras

11.Structure from motion (SFM)

12. Motion tracking

13.Augmented reality.

To support some of the above areas, Open CV includes a statistical machine learning library that contains :

1. Boosting

2. Decision tree learning

3. Gradient boosting trees

4.Expectation-maximization algorithm

5.k-nearest neighbor algorithm

6.Naïve Bays classifier

7.Artificial neural networks

8. Random forest

9. Random forest

10. Support vector machine (SVM)

11. Deep neural networks (DNN)

Numpy :

NumPy is an acronym for "Numeric Python“ or "Numerical Python". It is an open source extension module for Python, which provides fast precompiled functions for mathematical and numerical routines. Furthermore, NumPy enriches the programming language Python with powerful data structures for efficient computation of multi-dimensional arrays and matrices. The implementation is even aiming at huge matrices and arrays. Besides that the module supplies a large library of high -level mathematical functions to operate on these matrices and arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

1. A powerful N-dimensional array object

2. Sophisticated (broadcasting) functions

3. Tools for integrating C/C++ and Fortran code

4. Useful linear algebra, Fourier Transform, and random number capabilities.

Numpy Array :

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of non negative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

SciPy :

SciPy (Scientific Python) is often mentioned in the same breath with NumPy. SciPy extends the capabilities of NumPy with further useful functions for minimization regression, Fourier-transformation and many others.

NumPy is based on two earlier Python modules dealing with arrays. One of these Is Numeric. Numeric is like NumPy a Python module for high-performance, Numeric computing, but it is obsolete nowadays. Another predecessor of NumPy is Numarray, which is a complete rewrite of Numeric but is deprecated as well. NumPy is a merger of those two, i.e. it is build on the code of Numeric and the features of Numarray.

6.5 PYTHON Alternative To MATLAB:

Python in combination with Numpy, Scipy and Matplotlib can be used as a Replacement for MATLAB. The combination of NumPy, SciPy and Matplotlib is a free (meaning both "free" as in "free beer" and "free‟ as in "freedom") alternative to MATLAB. Even though MATLAB has a huge number of additional tool boxes available, NumPy has the advantage that Python is a more modern and complete programming language and – as we have said already before - is open source. SciPy adds even more MATLAB-like functionalities to Python. Python is rounded out in the direction of MATLAB with the module Matplotlib, which provides MATLAB-like plotting functionality.

Keras :

Keras is a high-level neural networks API, written in Python and capable of running on top of Tensor Flow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. 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).

TensorFlow :

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

Guiding Principles

User Friendliness : Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it Minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.

Modularity : A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as little restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can combine to create new models.

Easy Extensibility : New modules are simple to add (as new classes and functions), and existing modules provide sample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.

Work with Python : No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

SYS :

System-specific parameters and functions. This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter . The sys module provides information about constants, functions and methods of the Python interpreter. dir(system) gives a summary of the available constants, functions and methods. Another possibility is the help() function. Using help (sys) provides valuable detail information.

Sigmoid Function :

The sigmoid function in a neural network will generate the end point (activation) of inputs multiplied by their weights. For example, let‟s say we had two columns (features) of input data and one hidden node (neuron) in our neural network.

Each feature would be multiplied by its corresponding weight value and then added together and passed through the sigmoid (just like a logistic regression). To take that simple example and turn it into a neural network we just add more hidden units. In addition to adding more hidden units, we add a path from every input feature to each of those hidden units where it is multiplied by its corresponding weight. Each hidden unit takes the sum of its inputs weights and passes that through the sigmoid resulting in that unit's activation.

Properties of Sigmoid Function :

1. The sigmoid function returns a real-valued output.

2. sigmoid function take any range real number and returns the output value which falls in the range of **0 to 1.** The first derivative of the sigmoid function will be non-negative or non positive.

3. Non-Negative: If a number is greater than or equal to zero.

4. Non-Positive: If a number is less than or equal to Zero**.**

Softmax Function :

Softmax function calculates the probabilities distribution of the event over „n‟ different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs.

The main advantage of using Softmax is the output probabilities range. The range will 0 to 1, and the sum of all the probabilities will be equal to one. If the softmax function used for multi-classification model it returns the probabilities of each class and the target class will have the high probability.

The formula computes the exponential (e-power) of the given input value and the sum of exponential values of all the values in the inputs. Then the ratio of the exponential of the input value and the sum of exponential values is the output of the softmax function.

Properties Of Softmax Function :

1. The calculated probabilities will be in the range of 0 to 1.

2. The sum of all probabilities is equal to 0.

6.6 Haar Features :

Haar feature is similar to Karnals , which is generally used to detect edge. All human faces share some similar features, like eye region is darker than upper check region , noise region is brighter than eye region. By this match able features, their location and size will help us to detect a face.

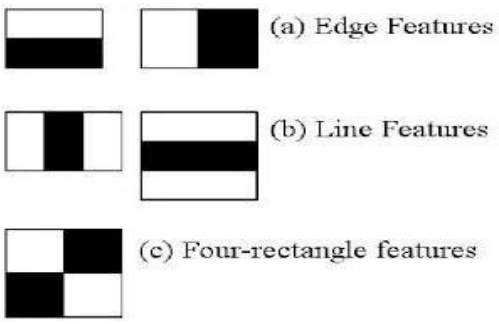

fig.6.2

Here are some Haar feature , using those we can say there is a face or not. Haar feature signifies that black region is represented by +1 and white region is represented by -1 . It uses a 24X24 window for an image. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle. Now all possible sizes and locations of each kernel is used to calculate plenty of features. For each feature calculation, we need to find sum of pixels under white and black rectangles. For 24X24

window, there will be 160000+ Haar features, which is a huge number. To solve this, they introduced the integral images. It simplifies calculation of sum of pixels, how Large may be the number of pixels, to an operation involving just four pixels.

Integral Images :

The basic idea of integral image is that to calculate the area. So, we do not need to sum up all the pixel values rather than we have to use the corner values and then a simple calculation is to be done. The integral image at location x , y contains the sum of the pixels above and to the left of x , y, inclusive :

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

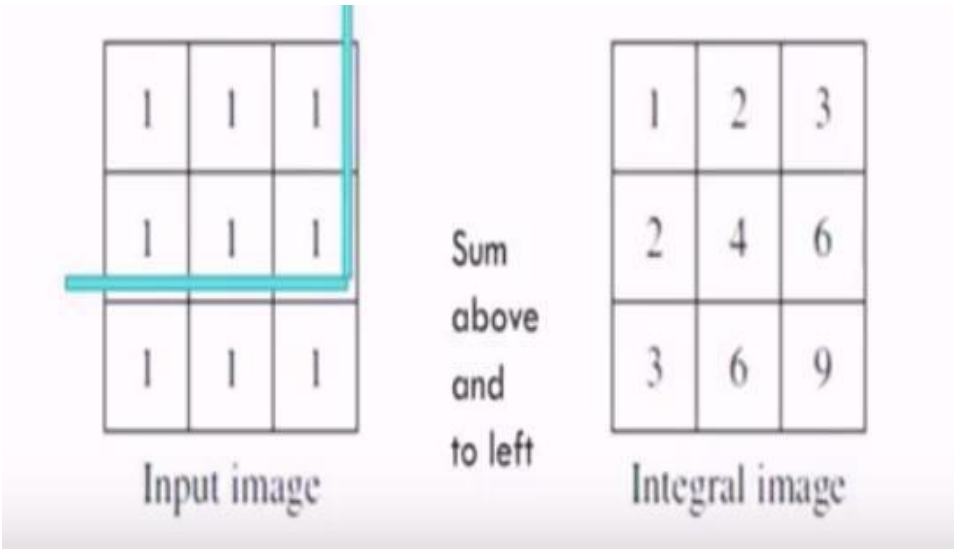

fig.6.3

The sum of the pixels within rectangle D can be computed with four array references:

1. The value of the integral image at location 1 is the sum of the pixels in Rectangle A. The value at location 2 is A + B, at location 3 is A + C, and at location 4 is A + B + C + D.

2. The sum within D can be computed as 4 + 1 − (2 + 3).

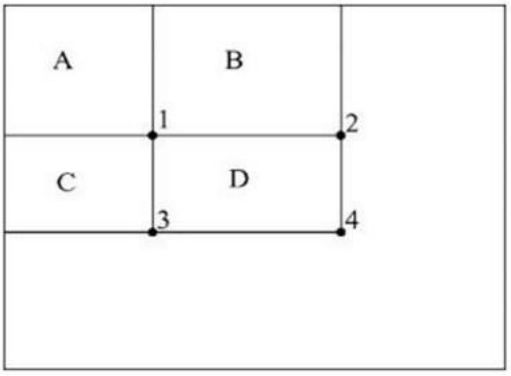

fig.6.4

The Process of Canny edge detection algorithm can be broken down to 5 different steps: 1.Apply Gaussian filter to smooth the image in order to remove the noise

2. Find the intensity gradients of the image.

3. Apply non-maximum suppression to get rid of spurious response to edge detection.

4. Apply double threshold to determine potential edges.

5. Track edge by hystersis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

The Canny algorithm contains a number of adjustable parameters, which can affect the computation time and effectiveness of the algorithm. The Canny algorithm is adaptable to various environments. Its parameters allow it to be tailored to recognition of edges of differing characteristics depending on the particular requirements of a given implementation

Robert Operator

The Roberts cross operator is used in image processing and computer vision for edge detetction. As a differential operator ,the idea behind the Roberts cross operator is to approximate the gradient of an image through discrete differentiation which is achieved by computing the sum of the squares of the differences between diagonally adjacent pixels. According to Roberts, an edge detector should have the following properties:

The produced edges should be well-defined, the background should contribute as little noise as possible, and the intensity of edges should correspond as close as possible to what a human would perceive. The results of this operation will highlight changes in intensity in a diagonal

direction. One of the most appealing aspects of this operation is its simplicity; the kernel is small and contains only integers.

Hard Exudates Extraction

Automated early detection of the presence of exudates can assist ophthalmologists to prevent the spread of the disease more efficiently. Hence, detection of exudates is an important diagnostic task.

Step 1: Read input color image.

Step 2: Green component of input image is extracted.

Step 3: Morphological Bottom hat operations performed on the green component of the image.

Step 4: Morphological Top hat operation is performed on the green component of the image.

Step 5: Subtract resultant image from step 4 with step 3.

Step 6: Optic disc elimination and hard exudates detection.

Adaboost :

Adaboost is used to eliminate the redundant feature of Haar. A very small number of these features can be combined to form an effective classifier. The main challenge is to find these features. A variant of AdaBoost is used both to select the features and to train the classifier.

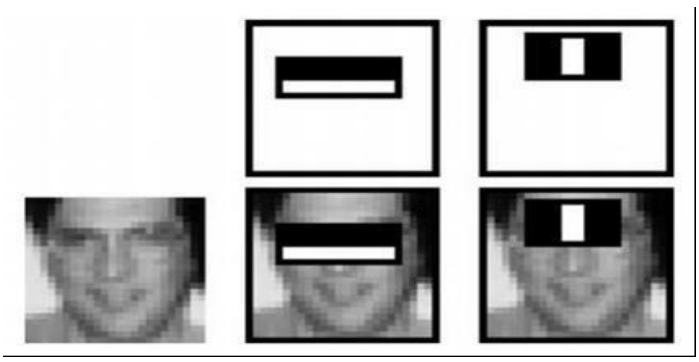

fig.6.5

The second feature is used for detecting the noise bridge, but it is irrelevant for upper lips as upper lips has more or less constant feature. So, we can easily eliminate it. Using adaboost we can determine which are relevant out of 160000+ feature. After finding all the features, a

weighted value is added to it which is which is used to evaluate a given window is a face or not.

$F(x) = a_1f_1(x) + a_2f_2(x) + a_3f_3(x) + a_4f_4(x) + a_5f_5(x) + \ldots.$

F(x) is strong classifier and f(x) is weak classifier.

Weak classifier always provide binary value i.e. 0 and 1. If the feature is present the value will be 1, otherwise value will be 0. Generally 2500 classifiers are used to make a strong classifier. Here selected features are said to be okay if it perform better than the random guessing i.e. it has to detect more than half of cases.

Suppose, we have a n input image of features are to be evaluated. Taking all 2500 features in a linear way it checks weather there is any threshold or not and then decide it is a face or not 640X480 resolution. Then we need to move 24X24 window through out the image and for each window 2500.

But instead of using all 2500 features for 24X24 times we will use cascade. Out of 2500 features , 1st 10 features are classified in one classifier, next 20-30 features are in next classifier, then next 100 in another classifier. So, like this we will increase the complexity.

The advantage is we can eliminate non face from 1st step instead of going through all 2500 features for 24X24 window. Suppose we have an image. If the image pass through 1st stage where 10 classifiers are stored, it may be a face. Then the image will go for 2nd stage checking. It the image does not pass the 1st stage, we can easily eliminate that.

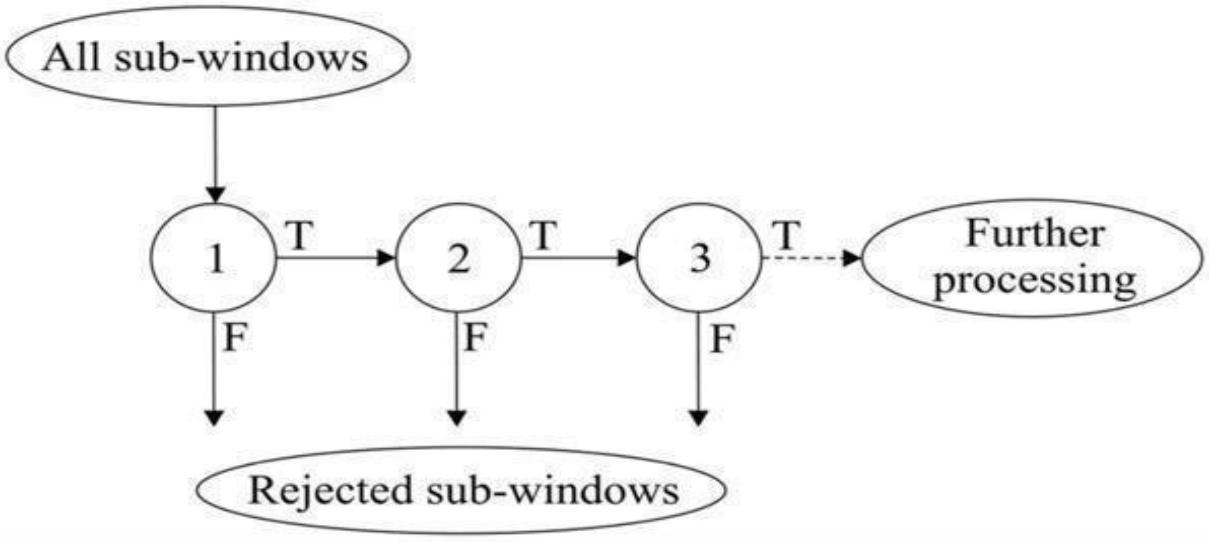

fig.6.6

Cascading is smaller, most efficient classifier . It is very easy to non face areas using cascading.

Haar Cascade Classifier in OpenCv

The algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, haar features shown in below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

Now all possible sizes and locations of each kernel is used to calculate plenty of features. (Just imagine how much computation it needs? Even a 24x24 window results over 160000 features). For each feature calculation, we need to find sum of pixels under white and black rectangles. To solve this, they introduced the integral images. It simplifies calculation of sum of pixels, how large may be the number of pixels, to an operation involving just four pixels. Nice, isn‟t it? It makes things super-fast.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant. So how do we select the best features out of 160000+ features? It is achieved by Adaboost.

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. But obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that best classifies the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then again same process is done. New error rates are calculated. Also new weights. The process is continued until required accuracy or error rate is achieved or required number of features are found).

Final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can‟t classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain).

So now you take an image. Take each 24x24 window. Apply 6000 features to it. Check if it is face or not. Wow.. Wow..Isn‟t it a little inefficient and time consuming? Yes, it is. Authors have a good solution for that.

In an image, most of the image region is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot. Don‟t process it again. Instead focus on region where there can be a face. This way, we can find more time to check a possible face region

For this they introduced the concept of Cascade of Classifiers. Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. (Normally first few stages will contain very less number of features). If a window fails the first stage, discard it. We don‟t consider remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region. Haar-like features are digital image features used in object recognition. They owe their name to their intuitive similarity with Haar wavelets and were used in the first real-time face detector.

Historically, working with only image intensities (i.e., the RGB pixel values at each and every pixel of image) made the task of feature calculation computationally expensive. A publication by Papageorgiou et al. discussed working with an alternate feature set based on Haar wavelets instead of the usual image intensities. Paul Viola and Michael Jones adapted the idea of using Haar wavelets and developed the so-called Haar-like features.

A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. This difference is then used to categorize subsections of an image. For example, with a human face, it is a common observation that among all faces the region of the eyes is darker than the region of the cheeks. Therefore, a common Haar feature for face detection is a set of two adjacent rectangles that lie above the eye and the cheek region. The position of these rectangles is defined relative to a detection window that acts like a bounding box to the target object (the face in this case).

In the detection phase of the Viola–Jones object detection framework, a window of the target size is moved over the input image, and for each subsection of the image the Haar-like feature is calculated. This difference is then compared to a learned threshold that separates non-objects from objects. Because such a Haar-like feature is only a weak learner or classifier (its detection quality is slightly better than random guessing) a large number of Haar-like features are necessary to describe an object with sufficient accuracy.

In the Viola–Jones object detection framework, the Haar-like features are therefore organized in something called a classifier cascade to form a strong learner or classifier.

The key advantage of a Haar-like feature over most other features is its calculation speed. Due to the use of integral images, a Haar-like feature of any size can be calculated in constant time (approximately 60 microprocessor instructions for a 2-rectangle feature).

CHAPTER 7

RESULTS

Input

Output

Neutral

Happy

CONCLUSION:

In this case, when the model predicts incorrectly, the correct label is often the second most likely emotion.The facial expression recognition system presented in this research work contributes a resilient face recognition model based on the mapping of behavioral characteristics with the physiological biometric characteristics. The physiological characteristics of the human face with relevance to various expressions such as happiness, sadness, fear, anger, surprise and disgust are associated with geometrical structures which restored as base matching template for the recognition system.The behavioral aspect of this system relates the attitude behind different expressions as property base. The property bases are alienated as exposed and hidden category in genetic algorithmic genes. The gene training set evaluates the expressional uniqueness of individual faces and provide a resilient expressional recognition model in the field of biometric security.

The design of a novel asymmetric cryptosystem based on biometrics having features like hierarchical group security eliminates the use of passwords and smart cards as opposed to earlier cryptosystems. It requires a special hardware support like all other biometrics system. This research work promises a new direction of research in the field of asymmetric biometric cryptosystems which is highly desirable in order to get rid of passwords and smart cards completely. Experimental analysis and study show that the hierarchical security structures are effective in geometric shape identification for physiological traits.

FUTURE SCOPE :

It is important to note that there is no specific formula to build a neural network that would guarantee to work well. Different problems would require different network architecture and a lot of trail and errors to produce desirable validation accuracy. This is the reason why neural nets are often perceived as "black box algorithms.".

In this project we got an accuracy of almost 70% which is not bad at all comparing all the previous models. But we need to improve in specific areas like-

1. number and configuration of convolutional layers

2. number and configuration of dense layers

3. dropout percentage in dense layers

But due to lack of highly configured system we could not go deeper into dense neural network as the system gets very slow and we will try to improve in these areas in future.

We would also like to train more databases into the system to make the model more and more accurate but again resources becomes a hindrance in the path and we also need to improve in several areas in future to resolve the errors and improve the accuracy.

Having examined techniques to cope with expression variation, in future it may be investigated in more depth about the face classification problem and optimal fusion of color and depth information. Further study can be laid down in the direction of allele of gene matching to the geometric factors of the facial expressions. The genetic property evolution framework for facial expressional system can be studied to suit the requirement of different security models such as criminal detection, governmental confidential security breaches etc

REFERENCES:

[1] D. C. Ali Mollahosseini and M. H. Mahoor. Going deeper in facial expression recognition using deep neural networks. IEEE

Winter Conference on Applications of Computer Vi- sion, 2016.

[2] S.-Y. D. Bo-Kyeong Kim, Jihyeon Roh and S.-Y. Lee. Hi- erarchical committee of deep convolutional neural networks for robust

facial expression recognition. Journal on Multi- modal User Interfaces, pages 1–17, 2015.

[3] F. Chollet. keras. https://github.com/fchollet/ keras, 2015.

[4] [4] P. Ekman and W. V. Friesen.Emotional facial action coding system. Unpublished manuscript, University of California at San

Francisco, 1983.

[5] B. Graham. Fractional max-pooling. 2015.

[6] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. JMLR

Proceedings, 2015

[7] M. S.-S. S. M. B. Z. L. X. S. B. P. J. Zhang, S. Ma and R. Mech. Salient object subitizing. IEEE Conference on Computer

Vision and Pattern Recognition (CVPR), 2015..

[8] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional

architecture for fast feature embedding. arXiv preprint arXiv:1408.5093, 2014

[9] Lab41. Misc, caffekeras util. https://github.com/ Lab41/Misc/blob/master/blog/kerasvgg/ caffekeras_util.py, 2015