

**Students Attention Monitoring System
Using AI**

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Technology in Information Technology

By

CHARAN V (Reg.No-39120018)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHISALAI,

CHENNAI – 600119

MAY-2023ii



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)



Accredited with —All grade by NAAC
Icopiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119
www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFAIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of (Charan.V Reg.No-39120018) who carried out the Project Phase-2 entitled "Students Attention Monitoring System via AI" under my supervision from January 2023 to May 2023.

Internal Guide

Dr .P.JEYANTHI , M.E., Ph.D.

Head of the Department

Dr. R. SUBHASHINI, M.E., Ph.D.

Submitted for Viva voce Examination held on 10/05/2023

Internal Examiner
External Examiner

I, CHARAN V (Reg.No- 39120018) hereby declare that the Project Phase-1 Report entitled "Students Attention Monitoring System via AI" done by me under the guidance of Dr. P.JEYANTHI ,M.E.,Ph.D. is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 10/05/2023

PLACE: Chennai

V. Charan V.

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing, **Dr. R. Subhashini, M.E., Ph.D.** Head of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.P.JEYANTHI M.E.,Ph.D**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-1 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

Abstract

There have been many studies in which researchers have attempted to classify student attentiveness. Many of these approaches depended on a qualitative analysis and lacked any quantitative analysis. Therefore, this work is focused on bridging the gap between qualitative and quantitative approaches to classify student attentiveness. Thus, this research applies machine learning algorithms (K-means and SVM) to automatically classify students as attentive or inattentive using data from a consumer RGB-D sensor. Results of this research can be used to improve teaching strategies for instructors at all levels and can aid instructors in implementing personalized learning systems, which is a National Academy of Engineering Grand Challenge. This research applies machine learning algorithms to an educational setting. Data from these algorithms can be used by instructors to provide valuable feedback on the effectiveness of their instructional strategies and pedagogies. Instructors can use this feedback to improve their instructional strategies, and students will benefit by achieving improved learning and subject mastery. Ultimately, this will result in the students' increased ability to do

work in their respective areas. Broadly, this work can help advance efforts in many areas of education and instruction. It is expected that improving instructional strategies and implementing personalized learning will help create more competent, capable, and prepared persons available for the future workforce.

Chapter 1

INTRODUCTION

In a traditional learning environment, teachers can understand the learning effect of students through the examination of each course. It is difficult to obtain the specific learning process of each student during the teaching process, and it is also difficult to predict the students' learning performance from the students features and learning behavior data. Therefore, it is also difficult for teachers to master the situation of students and provide timely intervention and help.

With the rapid development of online learning, especially the rise of MOOC in recent years, a large amount of data has gradually accumulated in various online learning platforms. The online behavior data are the data generated by the learners' interaction with the operating platform during the learning process, and is mainly recorded and stored in real time by the learning platform database and other tools. The classification of a large amount of learning behavior data has certain significance for the collection of data. The researcher determines the relevant indicators according to the objectives of the analysis, so as to determine the type of learning behavior from which the data are collected based on the results of the behavior classification, making the data collection process easier and more convenient, and making the

results of the behavior description more accurate. According to the theory of behavioral science, learners conduct learning activities through online learning platforms based on their own needs and generate different learning behaviors. The login behavior is generated based on the requirements of the course plan, the contents of the guide, the contents of the evaluation and the academic achievements, etc. The resource access behavior is generated based on the demand for the resources. The retrieve behavior is generated based on the requirements of the query learning content, the subject tool, the course content, and the like. The forum interaction behavior is generated based on the need to communicate and collaborate with others. The feedback and reflection behavior are generated based on reflecting learners themselves, evaluation curriculum and peers' needs. And the learning task-related behavior is generated based on the requirements of completing the learning task.

According to [Burgos et al. \(2018\)](#), with the increase in the amount of data on the online learning platform, researchers are looking for ways to make those data understandable and meaningful. In order to analyze and dig out more potential educational laws, researchers delve into the theory, framework, tools and practices of learning analysis. In recent years, people have been studying more and more research on learning behavior analysis, especially on predicting students' learning performance. [Valiente et al. \(2015\)](#) point out that with the continuous development of learning analysis research, researchers have begun to study learning predictions using machine learning methods and tools.

However, in recent years, with the widespread use of the internet, especially the rise of the mobile internet, teachers and students, are more inclined to use mobile smart terminals and social applications such as WeChat, WhatsApp, Twitter and Facebook to communicate with each other, discuss learning problems, even submit homework on tablets or mobile phones, rather than traditional tools on LMS or CMS. Because the above data are difficult to collect, researchers have much lower effective data available from online learning platform than before. Therefore, how to analyze online learning behavior and build an effective learning performance prediction platform is a valuable issue under the premise of limited data.

Many studies have been performed to determine the attentiveness of students in an instructional setting. Many of these studies relied on qualitative techniques rather than a quantitative approach to identifying and measuring attentiveness. Some researchers have also investigated quantitative approaches to monitoring student attentiveness. Biometric bracelets are being investigated as an indicator of student attentiveness. Eye and head pose tracking have also been used to determine student attentiveness [5]. Facial expressions have been used to infer student attentiveness for computer network courses [6]. Classifying students as attentive or inattentive can be helpful to the instructor by providing feedback as to which teaching style a particular student responds most favorably to. There are four learning orientations a learner will likely fall into: an innovator; an implementer; a sustainer; or a resistant learner [7]. If an instructor is able to classify a student as attentive or inattentive when the student is exposed to the associated teaching styles of each of these orientations, students

can be separated into course sections that implement their optimal learning style or can be assigned online teaching modules that use the teaching style a particular student will perform optimally with. Researchers have developed personalized e-learning systems based on Genetic algorithms (GA) and case-based reasoning (CBR) [8]. Adaptive user interfaces have also been developed based on personalized learning [9]. These approaches focus on online learning settings. The proposed system in this research can be applied to online as well as classroom instructional settings. This article describes a system that uses a commercial RGBD camera to monitor, count, and record student gestures, postures, facial expressions, and verbalizations in order to produce data for determining student attentiveness. Machine learning algorithms are then used to cluster, label, and classify the data for the purpose of classifying subsequent students as attentive or inattentive. This system is an imperative step towards developing the proposed personalized learning system described in this article.

Chapter 2

LITERATURE SURVEY

Student Behavior Recognition Classroom using Deep Transfer Learning with VGG-16

Tracking numerous students' behavior by observing and questioning them is a difficult task. Therefore, several methods based on automatic facial expression recognition have been proposed to capture and make a summary of students' behavior in the classroom. However, these methods cannot guarantee an effective classification due to the lack of huge datasets in this field. To improve students' behavior identification from video sequences, we propose in this paper a new approach based on deep transfer learning. Our approach pretrains the model on a facial expression dataset. Then, it transfers the model to classify students' behavior. Experimental results confirm that our approach ensures a preferment students' behavior classification.

2 Online learning behavior analysis based on machine learning

The purpose of this paper is to make a correlation analysis between students' online learning behavior features and course grade, and to attempt to build some effective prediction model based on limited data.

The prediction label in this paper is the course grade of students, and the eigenvalues available are student age, student gender, connection time, hits count and days of access. The machine learning model used in this paper is the classical three-layer feedforward neural networks, and the scaled conjugate gradient algorithm is adopted. Pearson correlation analysis method is used to find the relationships between course grade and the student eigenvalues.

A College Student Behavior Analysis and Management Method Based on Machine Learning Technology

A digital campus will generate a large amount of student-related data. How to analyze and apply these data has become the key to improving the management level of students. The analysis of student behavior data can not only assist schools in early warning of dangerous events and strengthen school safety but also can use real data to describe student behavior, thereby providing quantitative data support for scholarship and grant evaluation. This paper takes a university student as the research object, collects various data in the digital campus platform, and uses an adaptive K-means algorithm in the machine learning algorithm to cluster the data. Analyze the behavior of college students from the clustering results, so as to provide a basis for the education management and learning ability improvement of college students. Specifically, the student's study, life, and consumption data are selected as the data to describe the student's behavior at school. This data is input into the adaptive K-means algorithm to obtain different types of student consumption habits, living habits, and learning habits. Through the analysis results, it can be found that the problem of the group of students with low financial ability, the problem of too long online time for students, and the number of books borrowed are too low. According to the characteristics of these problems, teachers and schools are provided with targeted management suggestions. The analysis of student behavior based on machine learning technology provides a reference for the formulation of students' school management policies and provides teachers with information on students' personality characteristics, which is conducive to improving teachers' teaching effects. In short, the management of the results of student behavior analysis can provide a basis for the school to formulate reasonable management policies, thereby promoting precision management and scientific decision-making.

Online learning behavior analysis based on machine learning

Purpose The purpose of this paper is to make a correlation analysis between students' online learning behavior features and course grade, and to attempt to build some effective prediction model based on limited data. **Design/methodology/approach** The prediction label in this paper is the course grade of students, and the eigenvalues available are student age, student gender, connection time, hits count and days of access. The machine learning model used in this paper is the classical three-layer feedforward neural networks, and the scaled conjugate gradient algorithm is adopted. Pearson correlation analysis method is used to find the relationships between course grade and the student eigenvalues. **Findings** Days of access has the highest

correlation with course grade, followed by hits count, and connection time is less relevant to students' course grade. Student age and gender have the lowest correlation with course grade. Binary classification models have much higher prediction accuracy than multi-class classification models. Data normalization and data discretization can effectively improve the prediction accuracy of machine learning models, such as ANN model in this paper. Originality/value This paper may help teachers to find some clue to identify students with learning difficulties in advance and give timely help through the online learning behavior data. It shows that acceptable prediction models based on machine learning can be built using a small and limited data set. However, introducing external data into machine learning models to improve its prediction accuracy is still a valuable and hard issue.

CHAPTER 3

SYSTEM DESIGN

EXISTING SYSTEM

- The learner's behaviors such as login, exit and jump between web pages in the system are trajectory behaviors. Trajectory behaviors are the most extensive type of behavior and the most basic type of learning behaviors. When a learner retrieves a certain type of resource, this behavior belongs to both the resource learning behavior and the trajectory behavior. The difference between the trajectory behavior and the resource learning behavior is that the resource learning behavior contains the body of the search, and the track behavior is only to retrieve the action, not including the main content of the retrieval.

PROPOSED SYSTEM

This article describes a system that uses a commercial camera to monitor, count, and record student gestures, postures, facial expressions, and verbalizations in order to produce data for determining student attentiveness.

Machine learning algorithms are then used to cluster, label, and classify the data for the purpose of classifying subsequent students as attentive or inattentive.

This system is an imperative step towards developing the proposed personalized learning system described in this article.

The investigated as an indicator of student attentiveness . Eye and head pose tracking have also been used to determine student attentiveness . Facial expressions have been used to infer student attentiveness for computer network courses .

Four.1. Advantages

- The results from the system can be used to determine the learning style of a particular student.

An instructor can teach similar material with various teaching styles and record which particular student reacts best to a given style

SYSTEM ARCHITECTURE

BLOCK DIAGRAM

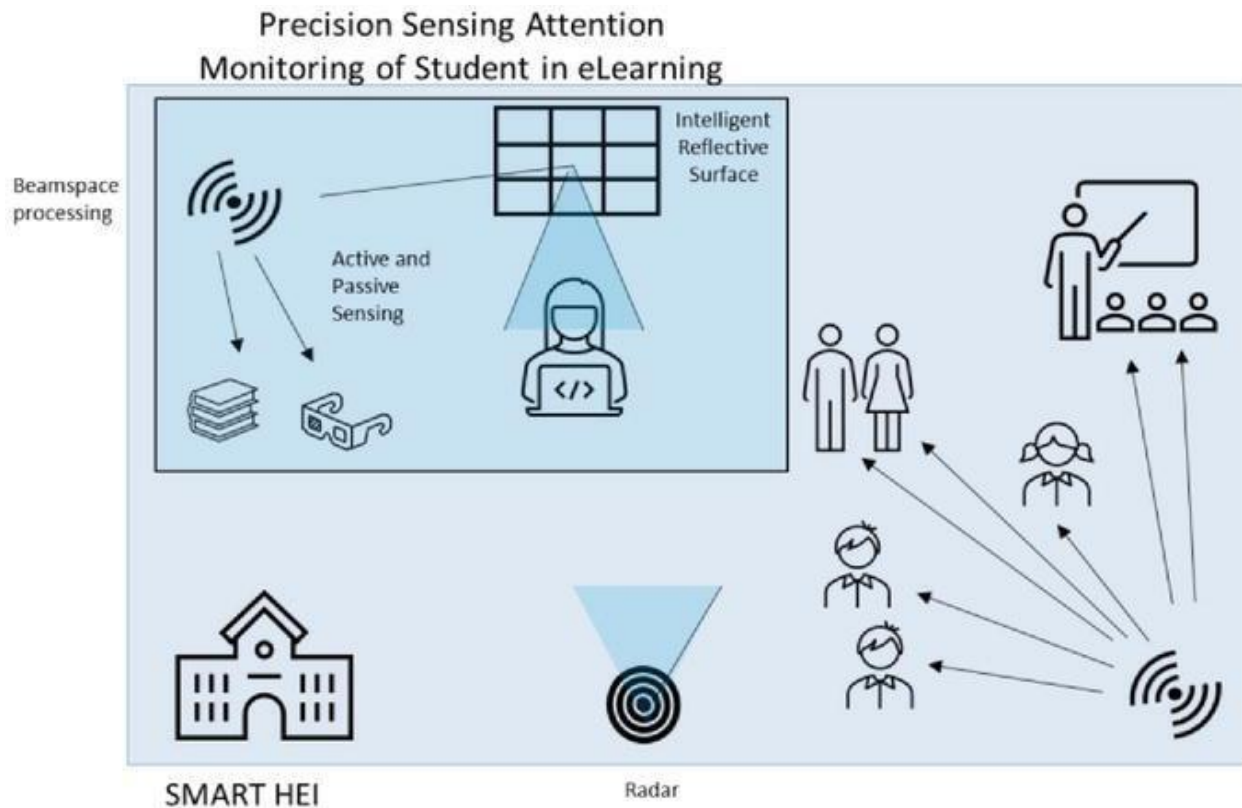


Fig 1 : Block Diagram

REQUIREMENTS:

System: windows 10, 64bit

Software tool: Anaconda Navigator, Spyder

Programming language: Python

MODULES:

Behavior-detecting Algorithm :

There are various behaviors that indicate attentiveness ; and many of these can be automatically observed and recorded using an RGB-D camera. Some examples of these

types of behaviors are: leaning forward ; speaking to the instructor; hand raising; eyebrow raising / lowering .

Database :

The data of each student are saved into the database separately; since each RGB-D sensor is used for one student. After each student is observed and the instructional session is over, the data are recorded into a new row inside of the database. The number of occurrences of each feature (e.g. hand raising or leaning forward) is stored into a different column of that row. .

Data Clustering :

When every student in the study has been observed, the data are clustered into two clusters using the K-means algorithm. The clusters of data are labeled as attentive or inattentive depending on the distance from the origin of the data; the cluster with a centroid closest to the origin is labeled **Data Generation:**

The data used in this analysis were created by randomly generating data within a fixed range of values (from zero to twelve); these values represented the number of times a student was observed exhibiting a specific attentiveness behavior. For illustration purposes, each hypothetical student was given only two features (attentiveness behavior) each: number of hand raises; and number of times eyebrows were raised. Figure 3 depicts the generated data. Each of the points in the figure represent a separate student. Twenty hypothetical students were used in this study.

K-means Algorithm :

The K-means algorithm was used to cluster the data. Algorithm 1 illustrates the basic procedure of the algorithm. In this research, the algorithm converged in four iterations as illustrated

Support Vector Machine Algorithm :

After the data were clustered and labeled using the Kmeans algorithm, they were classified using the SVM algorithm. Algorithm 2 illustrates the basic algorithm. The variable Θ represents the parameters and f is chosen based on the type of kernel function used. In this research, a Gaussian Kernel was chosen

CHAPTER 4

Image Processing in Python: Algorithms Tools, and

Methods You Should Know

Images define the world, each image has its own story, it contains a lot of crucial information that can be useful in many ways. This information can be obtained with the help of the technique known as **Image Processing**.

It is the core part of computer vision which plays a crucial role in many real-world examples like robotics, self-driving cars, and object detection. Image processing allows us to transform and manipulate thousands of images at a time and extract useful insights from them. It has a wide range of applications in almost every field.

Python is one of the widely used programming languages for this purpose. Its amazing libraries and tools help in achieving the task of image processing very efficiently.

Through this article, you will learn about classical algorithms, techniques, and tools to process the image and get the desired output.

Let's get into it!

What is image processing?

As the name says, image processing means processing the image and this may include many different techniques until we reach our goal.

The final output can be either in the form of an image or a corresponding feature of that image. This can be used for further analysis and decision making.

But what is an image?

An image can be represented as a 2D function $F(x,y)$ where x and y are spatial coordinates. The amplitude of F at a particular value of x,y is known as the intensity of an image at that point. If x,y , and the amplitude value is finite then we call it a digital image. It is an array of pixels arranged in columns and rows. Pixels are the elements of an image that contain information about intensity and color. An image can also be represented in 3D where x,y , and z become spatial coordinates. Pixels are arranged in the form of a matrix. This is known as an **RGB image**.

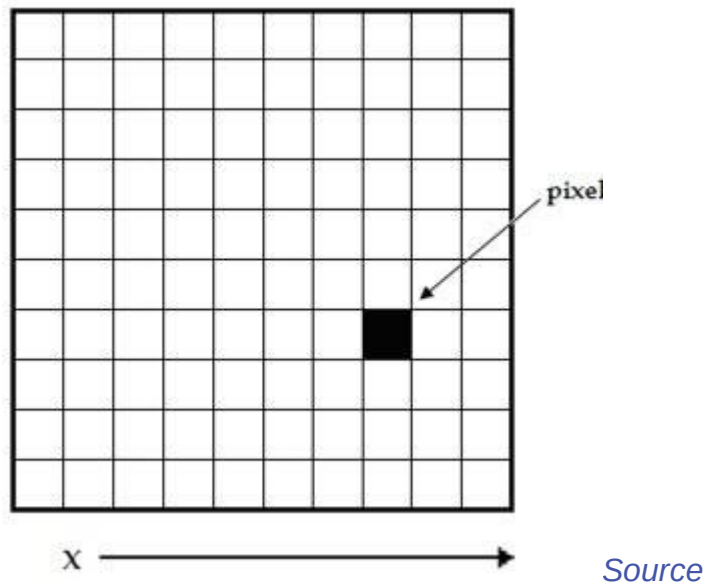
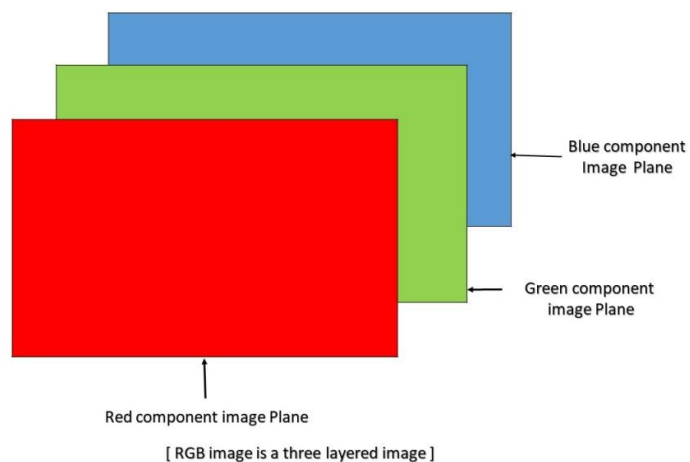


Fig 2



Source

Fig 3

There are various types of images:

- RGB image: It contains three layers of 2D image, these layers are Red, Green, and Blue channels.
- Grayscale image: These images contain shades of black and white and contain only a single channel.

Classic image processing algorithms **1.**

Morphological Image Processing

image processing tries to remove the imperfections from the binary images because binary regions produced by simple thresholding can be distorted by noise. It also helps in smoothing Morphological the image using opening and closing operations.

Morphological operations can be extended to grayscale images. It consists of non-linear operations related to the structure of features of an image. It depends on the related ordering of pixels but on their numerical values. This technique analyzes an image using a small template known as **structuring element** which is placed on different possible locations in the image and is compared with the corresponding neighbourhood pixels. A structuring element is a small matrix with 0 and 1 values.

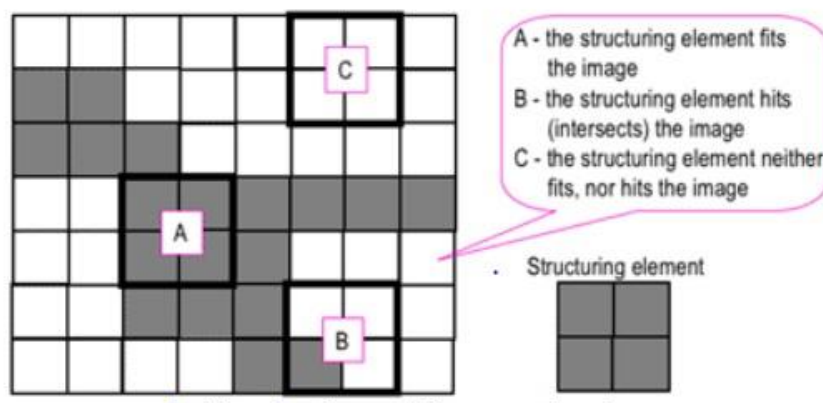
Let's see the two fundamental operations of morphological image processing, **Dilation and Erosion**:

- **dilation** operation adds pixels to the boundaries of the object in an image
- **erosion** operation removes the pixels from the object boundaries.

The number of pixels removed or added to the original image depends on the size of the structuring element.

At this point you may be thinking “what is a structuring element?” Let me explain:

Structuring element is a matrix consisting of only 0's and 1's that can have any arbitrary shape and size. It is positioned at all possible locations in the image and it is compared with the corresponding neighbourhood of pixels.



[Source](#)

Fig 4

The square structuring element 'A' fits in the object we want to select, the 'B' intersects the object and 'C' is out of the object.

The zero-one pattern defines the configuration of the structuring element. It's according to the shape of the object we want to select. The center of the structuring element identifies the pixel being processed.

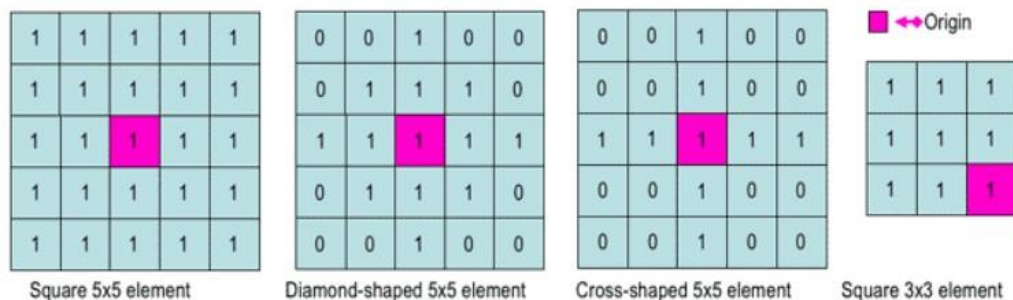
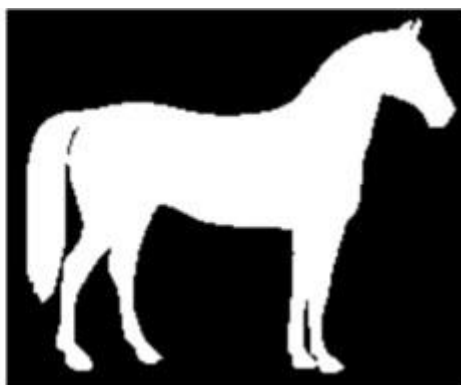
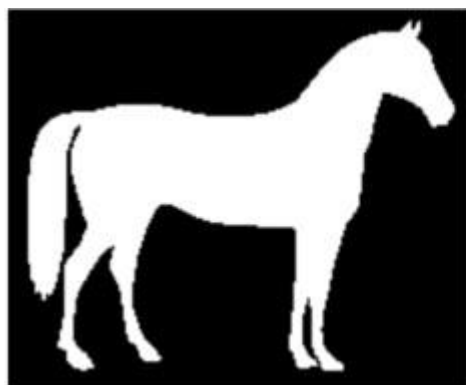
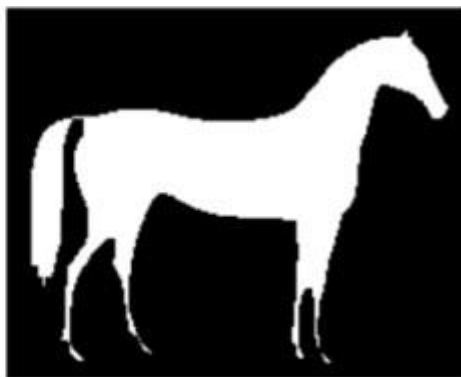
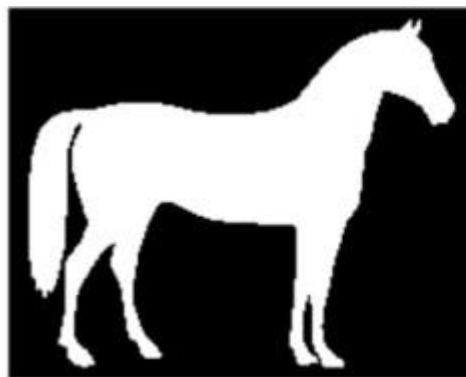


Fig 5



Dilation | [Source](#)

Fig 6



Erosion | [Source](#)

Fig 7

2. Gaussian Image Processing

Gaussian blur which is also known as gaussian smoothing, is the result of blurring an **image** by a **Gaussian** function.

It is **used to reduce image noise and reduce details**. The visual effect of this blurring technique is similar to looking at an image through the translucent screen. It is sometimes used in computer vision for image enhancement at different scales or as a data augmentation technique in deep learning.

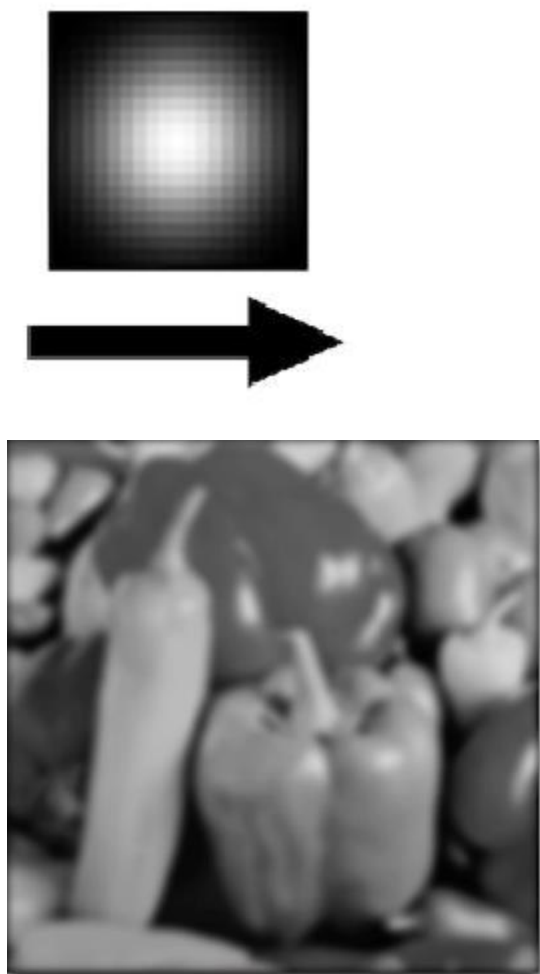
The basic gaussian function looks like:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

In practice, it is best to take advantage of the Gaussian blur's separable property by dividing the process into two passes. In the first pass, a one-dimensional kernel is used to blur the image in only the horizontal or vertical direction. In the second pass, the same one-dimensional kernel is used to blur in the remaining direction. The resulting effect is the same as convolving with a two-dimensional kernel in a single pass. Let's see an example to understand what gaussian filters do to an image.

If we have a filter which is normally distributed, and when its applied to an image, the results look like this:





Original

Filter

Result

[Source](#)

You can see that some of the edges have little less detail. The filter is giving more weight to the pixels at the center than the pixels away from the center. Gaussian filters are low-pass filters i.e. weakens the high frequencies. It is commonly [used in edge detection](#).

3. Fourier Transform in image processing

Fourier transform breaks down an image into sine and cosine components.

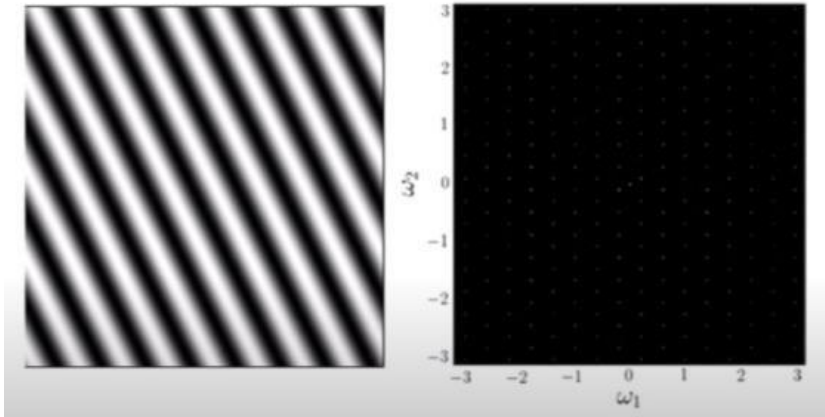
It has multiple applications like image reconstruction, image compression, or image filtering.

Since we are talking about images, we will take discrete fourier transform into consideration.

Let's consider a sinusoid, it comprises of three things:

- Magnitude – related to contrast
- Spatial frequency – related to brightness
- Phase – related to color information

The image in the frequency domain looks like this:



[*fig 8*](#)

The formula for 2D discrete fourier transform is:

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})}$$

In the above formula, $f(x,y)$ denotes the image.

The inverse fourier transform converts the transform back to image. The formula for 2D inverse discrete fourier transform is:

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(\frac{ux}{M} + \frac{vy}{N})}$$

4. Edge Detection in image processing

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness.

This could be very beneficial in extracting useful information from the image because most of the shape information is enclosed in the edges. Classic edge detection methods work by detecting discontinuities in the brightness.

It can rapidly react if some noise is detected in the image while detecting the variations of grey levels. Edges are defined as the local maxima of the gradient.

The most common edge detection algorithm is **sobel edge detection algorithm**. Sobel detection operator is made up of 3*3 convolutional kernels. A simple kernel Gx and a 90 degree rotated kernel Gy. Separate measurements are made by applying both the kernel separately to the image.

$$Gx = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * Image\ matrix$$

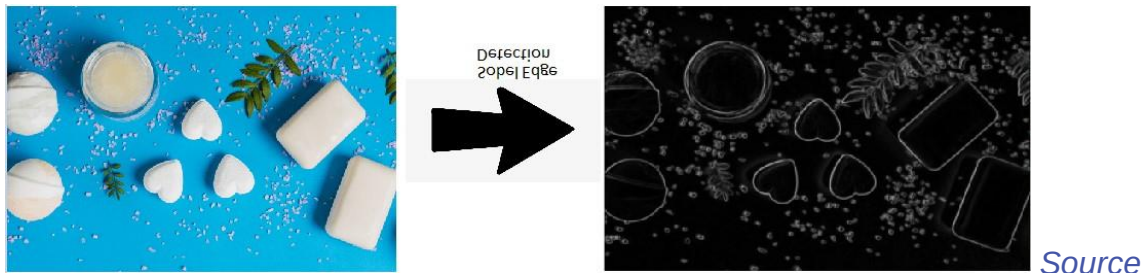
And,

$$Gy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * Image\ matrix$$

* denotes the 2D signal processing convolution operation.

Resulting gradient can be calculated as:

$$G = \sqrt{Gx^2 + Gy^2}$$



[fig 9](#)

5. Wavelet Image Processing

We saw a Fourier transform but it is only limited to the frequency. Wavelets take both time and frequency into the consideration. This transform is apt for non-stationary signals.

We know that edges are one of the important parts of the image, while applying the traditional filters it's been noticed that noise gets removed but image gets blurry. The wavelet transform

is designed in such a way that we get good frequency resolution for low frequency components. Below is the 2D wavelet transform example:



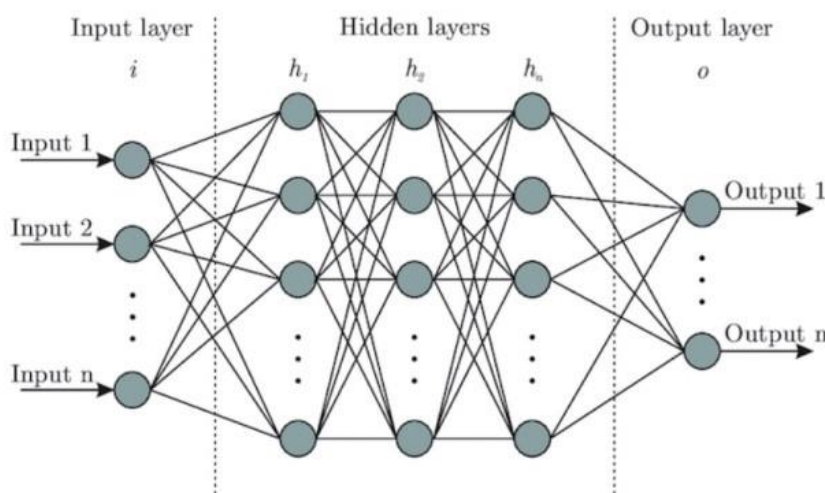
Fig 10

Image processing using Neural Networks

Neural Networks are multi-layered networks consisting of neurons or nodes. These neurons are the core processing units of the neural network. They are designed to act like human brains. They take in data, train themselves to recognize the patterns in the data and then predict the output.

A basic neural network has three layers:

1. Input layer
2. Hidden layer
3. Output layer



Basic neural network | [Source](#)

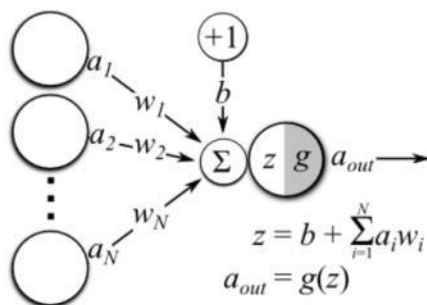
The input layers receive the input, the output layer predicts the output and the hidden layers

do most of the calculations. The number of hidden layers can be modified according to the requirements. There should be atleast one hidden layer in a neural network.

The basic working of the neural network is as follows:

1. Let's consider an image, each pixel is fed as input to each neuron of the first layer, neurons of one layer are connected to neurons of the next layer through channels.
2. Each of these channels is assigned a numerical value known as weight.
3. The inputs are multiplied by the corresponding weights and this weighted sum is then fed as input to the hidden layers.
4. The output from the hidden layers is passed through an activation function which will determine whether the particular neuron will be activated or not.
5. The activated neurons transmits data to the next hidden layers. In this manner, data is propagated through the network, this is known as Forward Propagation.
6. In the output layer, the neuron with the highest value predicts the output. These outputs are the probability values.
7. The predicted output is compared with the actual output to obtain the error. This information is then transferred back through the network, the process is known as Backpropagation.
8. Based on this information, the weights are adjusted. This cycle of forward and backward propagation is done several times on multiple inputs until the network predicts the output correctly in most of the cases.
9. This ends the training process of the neural network. The time taken to train the neural network may get high in some cases.

In the below image, **ai's** is the set of inputs, **wi's** are the weights, **z** is the output and **g** is any activation function.



Operations in a single neuron | [Source](#)

Here are some guidelines to prepare data for image processing. □ More data

needs to be fed to the model to get the better results.

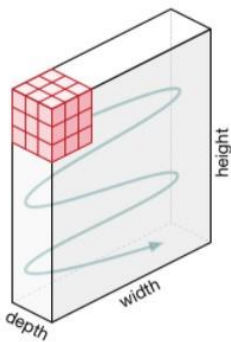
- Image dataset should be of high quality to get more clear information, but to process them you may require deeper neural networks.
- In many cases RGB images are converted to grayscale before feeding them into a neural network.

Types of Neural Network

Convolutional Neural Network

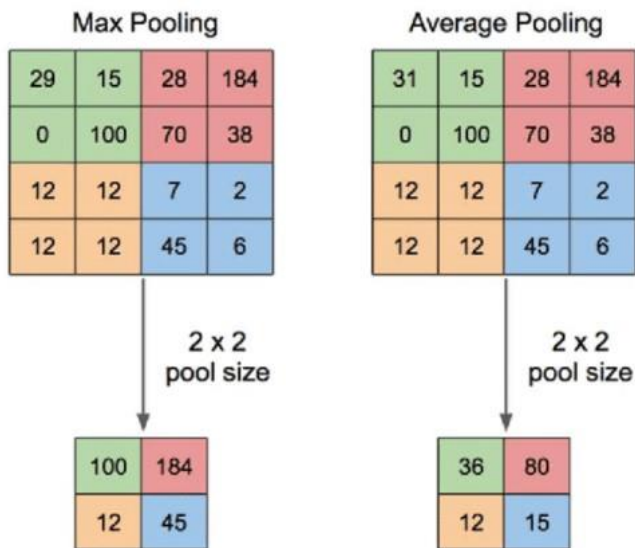
A convolutional neural network, ConvNets in short has three layers:

- **Convolutional Layer (CONV):** They are the core building block of CNN, it is responsible for performing convolution operation. The element involved in carrying out the convolution operation in this layer is called the **Kernel/Filter (matrix)**. The kernel makes horizontal and vertical shifts based on the **stride rate** until the full image is traversed.



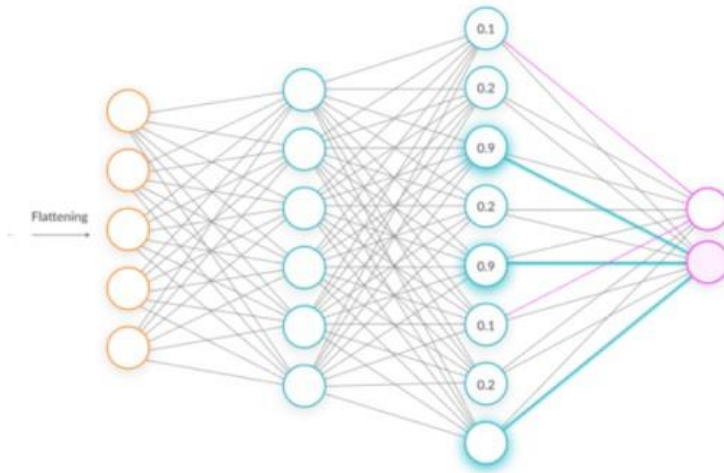
Movement of the kernel | [Source](#)

- **Pooling Layer (POOL):** This layer is responsible for dimensionality reduction. It helps to decrease the computational power required to process the data. There are two types of Pooling: Max Pooling and Average Pooling. Max pooling returns the maximum value from the area covered by the kernel on the image. Average pooling returns the average of all the values in the part of the image covered by the kernel.



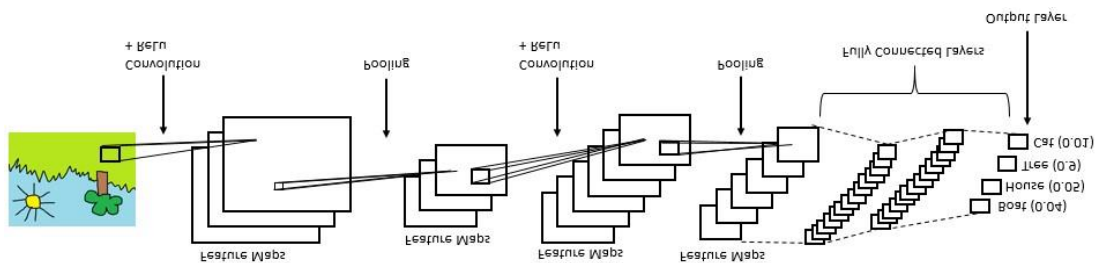
Pooling operation | [Source](#)

- **Fully Connected Layer (FC):** The **fully connected layer (FC)** operates on a flattened input where each input is connected to all neurons. If present, **FC layers** are usually found towards the end of **CNN** architectures.



Fully connected layers | [Source](#)

CNN is mainly used in extracting features from the image with help of its layers. CNNs are widely used in image classification where each input image is passed through the series of layers to get a probabilistic value between 0 and 1.



[Source](#)

Generative Adversarial Networks

Generative models use an unsupervised learning approach (there are images but there are no labels provided).

GANs are composed of two models **Generator** and **Discriminator**. *Generator* learns to make fake images that look realistic so as to fool the discriminator and *Discriminator* learns to distinguish fake from real images (it tries not to get fooled).

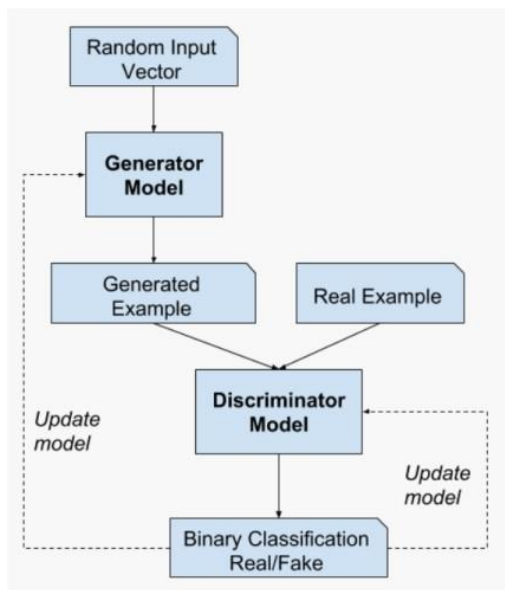
Generator is not allowed to see the real images, so it may produce poor results in the starting phase while the discriminator is allowed to look at real images but they are jumbled with the fake ones produced by the generator which it has to classify as real or fake.

Some noise is fed as input to the generator so that it's able to produce different examples every single time and not the same type image. Based on the scores predicted by the discriminator,

the generator tries to improve its results, after a certain point of time, the generator will be able to produce images that will be harder to distinguish, at that point of time, the user gets satisfied with its results. Discriminator also improves itself as it gets more and more realistic images at each round from the generator.

Popular types of GANs are Deep Convolutional GANs(DCGANs), Conditional GANs(cGANs), StyleGANs, CycleGAN, DiscoGAN, GauGAN and so on.

GANs are great for image generation and manipulation. Some applications of GANs include : Face Aging, Photo Blending, Super Resolution, Photo Inpainting, Clothing Translation.



[Source](#)

Image processing tools

1. OpenCV

It stands for Open Source Computer Vision Library. This library consists of around 2000+ optimised algorithms that are useful for computer vision and machine learning. There are several ways you can use opencv in image processing, a few are listed below:

- Converting images from one color space to another i.e. like between BGR and HSV, BGR and gray etc.
- Performing thresholding on images, like, simple thresholding, adaptive thresholding etc.
- Smoothing of images, like, applying custom filters to images and blurring of images. □ Performing morphological operations on images.
- Building image pyramids.
- Extracting foreground from images using GrabCut algorithm. □ Image segmentation using watershed algorithm.

Refer to [this link](#) for more details.

2. Scikit-image

It is an open-source library used for image preprocessing. It makes use of machine learning with built-in functions and can perform complex operations on images with just a few functions.

It works with numpy arrays and is a fairly simple library even for those who are new to python. Some operations that can be done using scikit image are :

- To implement thresholding operations use **try_all_threshold()** method on the image. It will use seven global thresholding algorithms. This is in the **filters** module.
- To implement edge detection use **sobel()** method in the **filters** module. This method requires a 2D grayscale image as an input, so we need to convert the image to grayscale.
- To implement gaussian smoothing use **gaussian()** method in the **filters** module.
- To apply histogram equalization, use **exposure** module, to apply normal histogram equalization to the original image, use **equalize_hist()** method and to apply adaptive equalization, use **equalize_adapthist()** method.
- To rotate the image use **rotate()** function under the **transform** module.
- To rescale the image use **rescale()** function from the **transform** module.
- To apply morphological operations use **binary_erosion()** and **binary_dilation()** function under the **morphology** module.

3. PIL/pillow

PIL stands for Python Image Library and **Pillow** is the friendly PIL fork by Alex Clark and Contributors. It's one of the powerful libraries. It supports a wide range of image formats like PPM, JPEG, TIFF, GIF, PNG, and BMP.

It can help you perform several operations on images like rotating, resizing, cropping, grayscaling etc. Let's go through some of those operations

To carry out manipulation operations there is a module in this library called **Image**.

- To load an image use the **open()** method. □ To display an image use **show()** method.
- To know the file format use **format** attribute □ To know the size of the image use **size** attribute
- To know about the pixel format use **mode** attribute.
- To save the image file after desired processing, use **save()** method. Pillow saves the image file in *png* format.
- To resize the image use **resize()** method that takes two arguments as width and height.
- To crop the image, use **crop()** method that takes one argument as a box tuple that defines position and size of the cropped region.
- To rotate the image use **rotate()** method that takes one argument as an integer or float number representing the degree of rotation.

- To flip the image use **transform()** method that take one argument among the following: Image.FLIP_LEFT_RIGHT, Image.FLIP_TOP_BOTTOM, Image.ROTATE_90, Image.ROTATE_180, Image.ROTATE_270.

READ ALSO

4. NumPy

With this library you can also perform simple image techniques, such as flipping images, extracting features, and analyzing them.

Images can be represented by numpy multi-dimensional arrays and so their type is **NdArrays**. A color image is a numpy array with 3 dimensions. By slicing the multi-dimensional array the RGB channels can be separated.

Below are some of the operations that can be performed using NumPy on the image (image is loaded in a variable named **test_img** using imread).

- To flip the image in a vertical direction, use **np.flipud(test_img)**.
- To flip the image in a horizontal direction, use **np.fliplr(test_img)**.
- To reverse the image, use **test_img[::-1]** (the image after storing it as the numpy array is named as <img_name>).
- To add filter to the image you can do this:

Example: **np.where(test_img > 150, 255, 0)**, this says that in this picture if you find anything with 150, then replace it with 255, else 0.

- You can also display the RGB channels separately. It can be done using this code snippet:

To obtain a red channel, do **test_img[:, :, 0]**, to obtain a green channel, do **test_img[:, :, 1]** and to obtain a blue channel, do **test_img[:, :, 2]**.

Python:

PYTHON (PROGRAMMING LANGUAGE)

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented,

imperative, functional programming, and procedural styles. It has a large and comprehensive standard library. Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation

ADVANTAGES OF IMAGE PROCESSING:

The purpose of image processing is divided into 5 groups.

They are:

VISUALIZATION - Observe the objects that are not visible.

IMAGE SHARPENING AND RESTORATION - To create a better image.

IMAGE RETRIEVAL - Seek for the image of interest.

MEASUREMENT OF PATTERN – Measures various objects in an image.

IMAGE RECOGNITION – Distinguish the objects in an image.

FUNDAMENTAL STEPS IN DIGITAL IMAGE PROCESSING:

1. IMAGE ACQUISITION:

This is the first step or process of the fundamental steps of digital image processing. Image acquisition could be as simple as being given an image that is already in digital form. Generally, the image acquisition stage involves pre-processing, such as scaling etc.

2. IMAGE ENHANCEMENT:

Image enhancement is among the simplest and most appealing areas of digital image processing. Basically, the idea behind enhancement techniques is to bring out detail that is obscured, or simply to highlight certain features of interest in an image. Such as, changing brightness & contrast etc.

3. IMAGE RESTORATION:

Image restoration is an area that also deals with improving the appearance of an image.

However, unlike enhancement, which is subjective, image restoration is objective, in the sense that restoration techniques tend to be based on mathematical or probabilistic models of image degradation.

4. COLOR IMAGE PROCESSING

Color image processing is an area that has been gaining its importance because of the significant increase in the use of digital images over the Internet. This may include color modelling and processing in a digital domain etc.

5. WAVELETS AND MULTIREOLUTION PROCESSING

Wavelets are the foundation for representing images in various degrees of resolution. Images subdivision successively into smaller regions for data compression and for pyramidal representation.

6. COMPRESSION

Compression deals with techniques for reducing the storage required to save an image or the bandwidth to transmit it. Particularly in the uses of internet it is very much necessary to compress data.

7. MORPHOLOGICAL PROCESSING

Morphological processing deals with tools for extracting image components that are useful in the representation and description of shape.

8. SEGMENTATION

Segmentation procedures partition an image into its constituent parts or objects. In general, autonomous segmentation is one of the most difficult tasks in digital image processing. A rugged segmentation procedure brings the process a long way toward successful solution of imaging problems that require objects to be identified individually.

9. REPRESENTATION AND DESCRIPTION

Representation and description almost always follow the output of a segmentation stage, which usually is raw pixel data, constituting either the boundary of a region or all the points in the region itself. Choosing a representation is only part of the solution for transforming raw data

into a form suitable for subsequent computer processing. Description deals with extracting attributes that result in some quantitative information of interest or are basic for differentiating one class of objects from another.

10. OBJECT RECOGNITION

Recognition is the process that assigns a label, such as, “vehicle” to an object based on its descriptors

How do I run a Python program under Windows?

This is not necessarily a straightforward question. If you are already familiar with running programs from the Windows command line then everything will seem obvious; otherwise, you might need a little more guidance.

Unless you use some sort of integrated development environment, you will end up *typing* Windows commands into what is variously referred to as a “DOS window” or “Command prompt window”. Usually you can create such a window from your search bar by searching for `cmd`. You should be able to recognize when you have started such a window because you will see a Windows “command prompt”, which usually looks like this:

```
C:\>
```

The letter may be different, and there might be other things after it, so you might just as easily see something like:

```
D:\YourName\Projects\Python>
```

depending on how your computer has been set up and what else you have recently done with it. Once you have started such a window, you are well on the way to running Python programs.

You need to realize that your Python scripts have to be processed by another program called the Python *interpreter*. The interpreter reads your script, compiles it into bytecodes, and then executes the bytecodes to run your program. So, how do you arrange for the interpreter to handle your Python?

First, you need to make sure that your command window recognises the word “py” as an instruction to start the interpreter. If you have opened a command window, you should try entering the command `py` and hitting return:

```
C:\Users\YourName> py
```

You should then see something like:

```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information. >>>
```

You have started the interpreter in “interactive mode”. That means you can enter Python statements or expressions interactively and have them executed or evaluated while you wait. This is one of Python’s strongest features. Check it by entering a few expressions of your choice and seeing the results:

```
>>>
```

```
>>> print("Hello")
Hello
>>> "Hello" * 3
'HelloHelloHello'
```

Many people use the interactive mode as a convenient yet highly programmable calculator. When you want to end your interactive Python session, call the `exit()` function or hold the Ctrl key down while you enter a Z, then hit the “Enter” key to get back to your Windows command prompt.

You may also find that you have a Start-menu entry such as Start ▸ Programs ▸ Python 3.x ▸ Python (command line) that results in you seeing the `>>>` prompt in a new window. If so, the window will disappear after you call the `exit()` function or enter the Ctrl-Z character; Windows is running a single “python” command in the window, and closes it when you terminate the interpreter.

Now that we know the `py` command is recognized, you can give your Python script to it. You’ll have to give either an absolute or a relative path to the Python script. Let’s say your Python script is located in your desktop and is named `hello.py`, and your command prompt is nicely opened in your home directory so you’re seeing something similar to:

```
C:\Users\YourName>
```

So now you'll ask the `py` command to give your script to Python by typing `py` followed by your script path:

```
C:\Users\YourName> py Desktop\hello.py  
hello
```

How do I make Python scripts executable?

On Windows, the standard Python installer already associates the `.py` extension with a file type (Python.File) and gives that file type an open command that runs the interpreter (`D:\Program Files\Python\python.exe "%1" %*`). This is enough to make scripts executable from the command prompt as `'foo.py'`. If you'd rather be able to execute the script by simple typing `'foo'` with no extension you need to add `.py` to the `PATHEXT` environment variable.

Why does Python sometimes take so long to start?

Usually Python starts very quickly on Windows, but occasionally there are bug reports that Python suddenly begins to take a long time to start up. This is made even more puzzling because Python will work fine on other Windows systems which appear to be configured identically.

The problem may be caused by a misconfiguration of virus checking software on the problem machine. Some virus scanners have been known to introduce startup overhead of two orders of magnitude when the scanner is configured to monitor all reads from the filesystem. Try checking the configuration of virus scanning software on your systems to ensure that they are indeed configured identically. McAfee, when configured to scan all file system read activity, is a particular offender.

How do I make an executable from a Python script?

See [How can I create a stand-alone binary from a Python script?](#) for a list of tools that can be used to make executables.

Is a `*.pyd` file the same as a DLL?

Yes, `.pyd` files are dll's, but there are a few differences. If you have a DLL named `foo.pyd`, then it must have a function `PyInit_foo()`. You can then write Python `"import foo"`, and Python will search for `foo.pyd` (as well as `foo.py`, `foo.pyc`) and if it finds it, will attempt to call `PyInit_foo()` to initialize it. You do not link your `.exe` with `foo.lib`, as that would cause Windows to require the DLL to be present.

Note that the search path for `foo.pyd` is `PYTHONPATH`, not the same as the path that Windows uses to search for `foo.dll`. Also, `foo.pyd` need not be present to run your program, whereas if

you linked your program with a dll, the dll is required. Of course, `foo.pyd` is required if you want to say `import foo`. In a DLL, linkage is declared in the source code with `__declspec(dllexport)`. In a `.pyd`, linkage is defined in a list of available functions.

How can I embed Python into a Windows application?

Embedding the Python interpreter in a Windows app can be summarized as follows:

1. Do `_not_` build Python into your `.exe` file directly. On Windows, Python must be a DLL to handle importing modules that are themselves DLL's. (This is the first key undocumented fact.) Instead, link to `pythonNN.dll`; it is typically installed in `C:\Windows\System`. `NN` is the Python version, a number such as "33" for Python 3.3.

You can link to Python in two different ways. Load-time linking means linking against `pythonNN.lib`, while run-time linking means linking against `pythonNN.dll`. (General note: `pythonNN.lib` is the so-called "import lib" corresponding to `pythonNN.dll`. It merely defines symbols for the linker.)

Run-time linking greatly simplifies link options; everything happens at run time. Your code must load `pythonNN.dll` using the Windows `LoadLibraryEx()` routine. The code must also use access routines and data in `pythonNN.dll` (that is, Python's C API's) using pointers obtained by the Windows `GetProcAddress()` routine. Macros can make using these pointers transparent to any C code that calls routines in Python's C API.

Borland note: convert `pythonNN.lib` to OMF format using `Coff2Omf.exe` first.

2. If you use SWIG, it is easy to create a Python "extension module" that will make the app's data and methods available to Python. SWIG will handle just about all the grungy details for you. The result is C code that you link *into* your `.exe` file (!) You do `_not_` have to create a DLL file, and this also simplifies linking.
3. SWIG will create an `init` function (a C function) whose name depends on the name of the extension module. For example, if the name of the module is `leo`, the `init` function will be called `initleo()`. If you use SWIG shadow classes, as you should, the `init` function will be called `initleoc()`. This initializes a mostly hidden helper class used by the shadow class.

The reason you can link the C code in step 2 into your `.exe` file is that calling the initialization function is equivalent to importing the module into Python! (This is the second key undocumented fact.)

4. In short, you can use the following code to initialize the Python interpreter with your extension module.

```
5. #include "python.h" 6. ...
7. Py_Initialize(); // Initialize Python.
8. initmyAppc(); // Initialize (import) the helper class.
9. PyRun_SimpleString("import myApp"); // Import the shadow class.
```

10. There are two problems with Python's C API which will become apparent if you use a compiler other than MSVC, the compiler used to build pythonNN.dll.

Problem 1: The so-called "Very High Level" functions that take FILE * arguments will not work in a multi-compiler environment because each compiler's notion of a struct FILE will be different. From an implementation standpoint these are very `_low_` level functions.

Problem 2: SWIG generates the following code when generating wrappers to void functions:

```
Py_INCREF(Py_None);
_resultobj = Py_None; return
_resultobj;
```

Alas, `Py_None` is a macro that expands to a reference to a complex data structure called `_Py_NoneStruct` inside pythonNN.dll. Again, this code will fail in a multi-compiler environment. Replace such code by:

```
return Py_BuildValue("");
```

It may be possible to use SWIG's `%typemap` command to make the change automatically, though I have not been able to get this to work (I'm a complete SWIG newbie).

11. Using a Python shell script to put up a Python interpreter window from inside your Windows app is not a good idea; the resulting window will be independent of your app's windowing system. Rather, you (or the `wxPythonWindow` class) should create a "native" interpreter window. It is easy to connect that window to the Python interpreter. You can redirect Python's i/o to `_any_` object that supports read and write, so all you need is a Python object (defined in your extension module) that contains `read()` and `write()` methods. How do I keep editors from inserting tabs into my Python source?

The FAQ does not recommend using tabs, and the Python style guide, PEP 8, recommends 4 spaces for distributed Python code; this is also the Emacs python-mode default.

Under any editor, mixing tabs and spaces is a bad idea. MSVC is no different in this respect, and is easily configured to use spaces: Take Tools ▸ Options ▸ Tabs, and for file type "Default" set "Tab size" and "Indent size" to 4, and select the "Insert spaces" radio button.

Python raises `IndentationError` or `TabError` if mixed tabs and spaces are causing problems in leading whitespace. You may also run the `tabnanny` module to check a directory tree in batch mode.

How do I check for a keypress without blocking?

Use the `msvcrt` module. This is a standard Windows-specific extension module. It defines a function `kbhit()` which checks whether a keyboard hit is present, and `getch()` which gets one character without echoing it.

OPENCV

OpenCV (*Open source computer vision*) is a [library of programming functions](#) mainly aimed at real-time [computer vision](#). Originally developed by [Intel](#), it was later supported by [Willow Garage](#) then Itseez (which was later acquired by Intel). The library is [cross-platform](#) and free for use under the [open-source BSD license](#).

OpenCV supports the [deep learning](#) frameworks [TensorFlow](#), [Torch/PyTorch](#) and [Caffe](#)

Conclusion:

We confirmed and tested cohort and non-cohort detection using distinctive deep learning knowledge of classifier algorithms. The overall performance of those models is measured and as compared the use of numerous parameters which include accuracy, reproducibility and don't forget. We finished 89.66% accuracy with InceptionResNetV2 and DenseNet201, that's better than other fashions.

In this device, we explored some widespread problems that have an effect on the generality of any model for automated pavement evaluation. Image variation is decided by the scale of the image, the space from the hoop to the digicam perspective, and the lights effect. We educated an item detection model at the Kaggle Lane dataset. We used a quicker RNN with Inception V2 because the base model for item detection. To take a look at the generality of the version, we used a spread of conditions, inclusive of small photograph sizes, unique picture types, and lights consequences. We have attempted to identify elements that can have an effect on the overall pattern of publicity. These elements encompass photograph length, digicam source, light stages, and digicam interference. To look into those factors, we performed four experiments. In each experiment we study one situation.

Result screenshots:

Code:

References:

- V. Snider, "Use of self-monitoring of attention with Id students: Research and application," *Learning Disability Quarterly*, pp. 139–151, 2019.
- [2] E. E. McDowell et al., "A multivariate study of teacher immediacy, teaching effectiveness, and student attentiveness at the junior high and senior high levels.," 2018.
- [3] R. P. Grobe and T. J. Pettibone, "Effect of instructional pace on student attentiveness," *The Journal of Educational Research*, pp. 131–134, 2018.
- [4] S. Simon, "Biosensors to monitor students attentiveness.." Retrieved June 5, 2012, from <http://www.reuters.com/article/2012/06/13/>, June 2018.
- [5] S. Asteriadis, P. Tzouveli, K. Karpouzis, and S. Kollias, "Estimation of behavioral user state based on eye gaze and head pose-application in an e-learning environment," *Multimedia Tools and Applications*, vol. 41, no. 3, pp. 469–493, 2016.
- [6] H.-R. Chen, "Assessment of learners' attention to e-learning by monitoring facial expressions for computer network courses," *Journal of Educational Computing Research*, vol. 47, no. 4, pp. 371–385, 2016.
- [7] M. Martinez, "What is personalized learning? are we there yet," *ELearning Developer's Journal*, 2018. [8] M.-J. Huang, H.-S. Huang, and M.-Y. Chen, "Constructing a personalized elearning system based on genetic algorithm and casebased reasoning approach," *Expert Systems with Applications*, vol. 33, no. 3, pp. 551–564, 2007.
- [9] J. Liu, C. K. Wong, and K. K. Hui, "An adaptive user interface based on personalized learning," *Intelligent Systems, IEEE*, vol. 18, no. 2, pp. 52– 57, 2003.
- [10] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability.*, pp. 281–297, University of California Press., 1967. [11] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [12] "Kinect for Windows." <http://www.microsoft.com/enus/kinectforwindows/>, 2013.
- [13] A. Guye-Vuillème, T. Capin, S. Pandzic, N. Thalmann, and D. Thalmann, "Nonverbal communication interface for collaborative virtual environments," *Virtual Reality*, vol. 4, no. 1, pp. 49–59, 1999.
- [14] H. Wagner, C. MacDonald, and A. Manstead, "Communication of individual emotions by spontaneous facial expressions," *Journal of Personality and Social Psychology*, vol. 50(4), pp. 737–743, 1986.

- [15] G. Gan, C. Ma, and J. Wu, Data Clustering: Theory, Algorithms, and Applications (ASIAM Series on Statistics and Applied Probability). Society for Industrial and Applied Mathematics, 2007.