**LOGO DETECTION USING OPENCV**

**ABSTRACT:**

Logo detection is a crucial task in computer vision with numerous applications across various industries, including marketing, brand monitoring, and image retrieval. This abstract focuses on the utilization of OpenCV (Open Source Computer Vision Library) for logo detection within images or video. OpenCV provides a rich set of tools and functions for image processing and computer vision tasks, making it a powerful framework for logo recognition.

The proposed method involves several key steps: image preprocessing to enhance features, feature extraction using techniques like SIFT (Scale-Invariant Feature Transform) or SURF (Speeded-Up Robust Features), logo matching through descriptor comparison or deep learning models like yolo or CNNs (Convolutional Neural Networks),

**OBJECTIVE:**

YOLO (You Only Look Once) is an object detection algorithm that divides an image into a grid and assigns bounding boxes and class predictions to objects within the grid. While YOLO can be used for various object detection tasks, including detecting logos, it requires a well-trained model and labeled dataset for logo detection.

**PROBLEM STATEMENT:**

The problem statement of logo detection using Support Vector Machines (SVM) involves the identification and classification of logos within images.

**INTRODUCTION:**

Logos usually consist of texts, shapes, images, or their combination. Logo detection benefits a wide range of applications in different areas, such as intelligent transportation, social media monitoring, and infringement detection. Meanwhile, some competitions have emerged, such as Robust Logo Detection Grand Challenge and Few-shot Logo Detection. The main task of logo detection is to determine the location of a specific logo in images/videos and identify them. Although it may be regarded as a particular object task, logo detection in real-world images can be pretty challenging since numerous brands may have highly diverse contexts, varied scales, changes in illumination, size, resolution, and even nonrigid deformation. Many previous works on logo detection employ handcrafted features to represent logos and use statistical classifiers for classification. Such methods suffer from complex image preprocessing pipelines and poor robustness when dealing with a much larger number of logos. Recent years have witnessed the rousing success of deep learning since ImageNet Large Scale Visual Recognition Challenge This work was supported by the National Nature Science Foundation of China (No.62072289, 61972378, U193620), CAAI-Huawei MindSpore Open Fund. S. Hou, J. Li, Q. Hou, Y. Zhao and Y. Zheng are School of Information Science and Engineering, Shandong Normal University, Shandong, 250358, China. Email: sujuanhou@sdnu.edu.cn, 2021317140@stu.sdnu.edu.cn, 2019309052@stu.sdnu.edu.cn, yannazhao@sdnu.edu.cn, and zhengyuanjie@gmail.com. W. Min and S. Jiang are with the Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China, and also with University of Chinese Academy of Sciences, Beijing, 100049, China. Email:(ILSVRC) [11]. Deep learning-based solutions with expressive feature representation capability offer better robustness,accuracy, and speed and thus attract increasing attention.

There are more than 100 papers about logo detection from1993 to 2022, and a concise milestone of logo detection We can see that many deep learning-based logo detection strategies have been proposed since 2015. This survey mainly concentrates on deep learning-based solutions specially developed for logo detection.Even though deep learning has dominated the logo research community, a comprehensive and in-depth survey on deep learning-based solutions is lacking. In this survey, we mainly focus on the advances in recent deep learning for logo detection. We provide in-depth analysis and discussion on existing studies in various aspects, covering datasets, pipe lined used, task types, detection strategies, loss functions, their contributions and limitations. We also try to analyze potential research challenges and future research directions for logo detection. We hope our work could provide a novel perspective to promote the understanding of deep learning-based logo detection, foster research on open challenges, and speed up the development of the logo research field.The rest of this paper is organized as follows. In Section II,

we investigate the public logo detection datasets. In Section III,we review and organize the currently available work on logo detection. In Section IV, we introduce the applications of logo detection in real-world scenarios. In Section V, we discuss its challenges and prominent future research directions. Finally,we summarize the whole text in Section VI.

**DOMAIN EXPLANATION:**

**Deep learning:**

 Deep [neural networks](https://www.sciencedirect.com/topics/physics-and-astronomy/neural-networks) are now the state-of-the-art machine learning models across a variety of areas, from image analysis to [natural language processing](https://www.sciencedirect.com/topics/engineering/natural-language-processing), and widely deployed in academia and industry.

These developments have a huge potential for medical imaging technology, medical data analysis, medical diagnostics and healthcare in general, slowly being realized. We provide a short overview of recent advances and some associated challenges in machine learning applied to medical image processing and image analysis.

Long before deep learning was used, traditional machine learning methods were mainly used. Such as Decision Trees, SVM, Naïve Bayes Classifier and Logistic Regression.

These algorithms are also called flat algorithms. Flat here means that these algorithms cannot normally be applied directly to the raw data (such as .csv, images, text, etc.). We need a preprocessing step called Feature Extraction.

The result of Feature Extraction is a representation of the given raw data that can now be used by these classic machine learning algorithms to perform a task. For example, the classification of the data into several categories or classes.

Feature Extraction is usually quite complex and requires detailed knowledge of the problem domain. This preprocessing layer must be adapted, tested and refined over several iterations for optimal results. On the other side are the artificial neural networks of Deep Learning. These do not need the Feature Extraction step. The layers are able to learn an implicit representation of the raw data directly and on their own. Here, a more and more abstract and compressed representation of the raw data is produced over several layers of artificial neural-nets. This compressed representation of the input data is then used to produce the result. The result can be, for example, the classification of the input data into different classes.

**DIGITAL IMAGE PROCESSING**

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

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

1. **Basics of Image Processing:-**

**FUNDAMENTALS OF DIGITAL IMAGE**

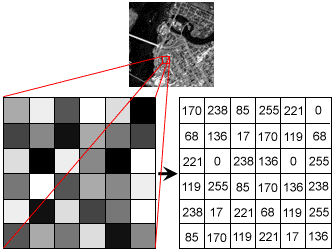
**1.1 IMAGE:**

An image is a two-dimensional picture, which has a similar appearance to some subject usually a physical object or a person.

Image is a two-dimensional, such as a photograph, screen display, and as well as a three-dimensional, such as a statue. They may be captured by optical devices—such as cameras, mirrors, lenses, telescopes, microscopes, etc. and natural objects and phenomena, such as the human eye or water surfaces.

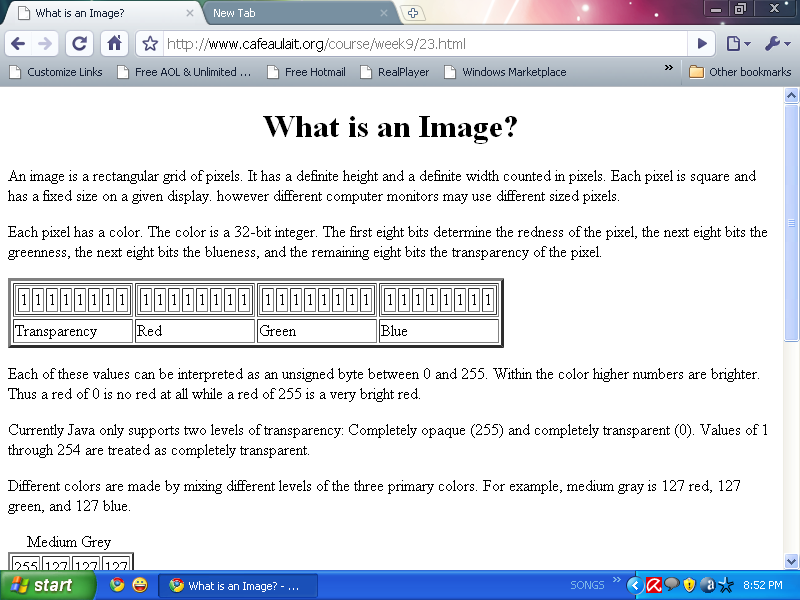
The word image is also used in the broader sense of any two-dimensional figure such as a map, a graph, a pie chart, or an abstract painting. In this wider sense, images can also be rendered manually, such as by drawing, painting, carving, rendered automatically by printing or computer graphics technology, or developed by a combination of methods, especially in a pseudo-photograph.

An image is a rectangular grid of pixels. It has a definite height and a definite width counted in pixels. Each pixel is square and has a fixed size on a given display. However different computer monitors may use different sized pixels. The pixels that constitute an image are ordered as a grid (columns and rows); each pixel consists of numbers representing magnitudes of brightness and color.



**Fig: Gray Scale Image Pixel Value Analysis**

Each pixel has a color. The color is a 32-bit integer. The first eight bits determine the redness of the pixel, the next eight bits the greenness, the next eight bits the blueness, and the remaining eight bits the transparency of the pixel.



**Fig: BIT Transferred for Red, Green and Blue plane (24bit=8bit red;8-bit green;8bit blue)**

**IMAGE FILE SIZES:**

Image file size is expressed as the number of bytes that increases with the number of pixels composing an image, and the color depth of the pixels. The greater the number of rows and columns, the greater the image resolution, and the larger the file. Also, each pixel of an image increases in size when its color depth increases, an 8-bit pixel (1 byte) stores 256 colors, a 24-bit pixel (3 bytes) stores 16 million colors, the latter known as true color.Image compression uses algorithms to decrease the size of a file. High resolution cameras produce large image files, ranging from hundreds of kilobytes to megabytes, per the camera's resolution and the image-storage format capacity. High resolution digital cameras record 12 megapixel (1MP = 1,000,000 pixels / 1 million) images, or more, in true color. For example, an image recorded by a 12 MP camera; since each pixel uses 3 bytes to record true color, the uncompressed image would occupy 36,000,000 bytes of memory, a great amount of digital storage for one image, given that cameras must record and store many images to be practical. Faced with large file sizes, both within the camera and a storage disc, image file formats were developed to store such large images.

**IMAGE FILE FORMATS:**

Image file formats are standardized means of organizing and storing images. This entry is about digital image formats used to store photographic and other images. Image files are composed of either pixel or vector (geometric) data that are rasterized to pixels when displayed (with few exceptions) in a vector graphic display. Including proprietary types, there are hundreds of image file types. The PNG, JPEG, and GIF formats are most often used to display images on the Internet.In addition to straight image formats, Metafile formats are portable formats which can include both raster and vector information. The metafile format is an intermediate format. Most Windows applications open metafiles and then save them in their own native format.

**IMAGE PROCESSING:**

Digital image processing, the manipulation of images by computer, is relatively recent development in terms of man’s ancient fascination with visual stimuli. In its short history, it has been applied to practically every type of images with varying degree of success. The inherent subjective appeal of pictorial displays attracts perhaps a disproportionate amount of attention from the scientists and also from the layman. Digital image processing like other glamour fields, suffers from myths, mis-connect ions, mis-understandings and mis-information. It is vast umbrella under which fall diverse aspect of optics, electronics, mathematics, photography graphics and computer technology. It is truly multidisciplinary endeavor ploughed with imprecise jargon.

Several factor combine to indicate a lively future for digital image processing. A major factor is the declining cost of computer equipment. Several new technological trends promise to further promote digital image processing. These include parallel processing mode practical by low cost microprocessors, and the use of charge coupled devices (CCDs) for digitizing, storage during processing and display and large low cost of image storage arrays.

**Image Acquisition:**

**Image Acquisition** is to acquire a digital image. To do so requires an image sensor and the capability to digitize the signal produced by the sensor. The sensor could be monochrome or color TV camera that produces an entire image of the problem domain every 1/30 sec. the image sensor could also be line scan camera that produces a single image line at a time. In this case, the objects motion past the line.



**Fig: Digital camera**

Scanner produces a two-dimensional image. If the output of the camera or other imaging sensor is not in digital form, an analog to digital converter digitizes it. The nature of the sensor and the image it produces are determined by the application.



**Fig: Mobile based Camera**

**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 interesting an image. A familiar example of enhancement is when we increase the contrast of an image because “it looks better.” It is important to keep in mind that enhancement is a very subjective area of image processing.

  
**Fig: Image enhancement process for Gray Scale Image and Colour Image using Histogram Bits**

**1.5.3 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.



Fig: Noise image🡪 Image Enhancement

Enhancement, on the other hand, is based on human subjective preferences regarding what constitutes a “good” enhancement result. For example, contrast stretching is considered an enhancement technique because it is based primarily on the pleasing aspects it might present to the viewer, where as removal of image blur by applying a deblurring function is considered a restoration technique.

**Color image processing:**

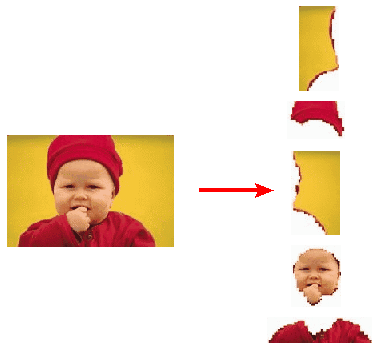
The use of color in image processing is motivated by two principal factors. First, color is a powerful descriptor that often simplifies object identification and extraction from a scene. Second, humans can discern thousands of color shades and intensities, compared to about only two dozen shades of gray. This second factor is particularly important in manual image analysis.

****

**Fig: gray Scale image 🡪 Colour Image**

**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.



**Fig: Image Segment Process**

On the other hand, weak or erratic segmentation algorithms almost always guarantee eventual failure. In general, the more accurate the segmentation, the more likely recognition is to succeed.

Digital image is defined as a two dimensional function f(x, y), where x and y are spatial (plane) coordinates, and the amplitude of f at any pair of coordinates (x, y) is called intensity or grey level of the image at that point. The field of digital image processing refers to processing digital images by means of a digital computer. The digital image is composed of a finite number of elements, each of which has a particular location and value. The elements are referred to as picture elements, image elements, pels, and pixels. Pixel is the term most widely used.

image compression:

digital image compression addresses the problem of reducing the amount of data required to represent a digital image. the underlying basis of the reduction process is removal of redundant data. from the mathematical viewpoint, this amounts to transforming a 2d pixel array into a statically uncorrelated data set. the data redundancy is not an abstract concept but a mathematically quantifiable entity. if n1 and n2 denote the number of information-carrying units in two data sets that represent the same information, the relative data redundancy  [2] of the first data set (the one characterized by n1) can be defined as,

* 
* where  called as compression ratio [2]. it is defined as
* = 

in image compression, three basic data redundancies can be identified and exploited: coding redundancy, interpixel redundancy, and phychovisal redundancy. image compression is achieved when one or more of these redundancies are reduced or eliminated. the image compression is mainly used for image transmission and storage. image transmission applications are in broadcast television; remote sensing via satellite, air-craft, radar, or sonar; teleconferencing; computer communications; and facsimile transmission. image storage is required most commonly for educational and business documents, medical images that arise in computer tomography (ct), magnetic resonance imaging (mri) and digital radiology, motion pictures, satellite images, weather maps, geological surveys, and so on.

image compression model

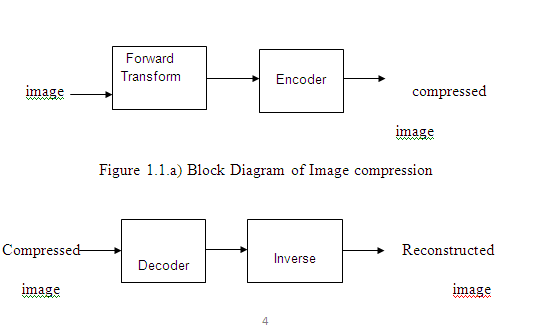


fig:1.1b) decompression process for image

image compression types

there are two types’ image compression techniques.

1. lossy image compression

2. lossless image compression

compression ratio:

1. lossy image compression :

lossy compression provides higher levels of data reduction but result in a less than perfect reproduction of the original image. it provides high compression ratio. lossy image compression is useful in applications such as broadcast television, videoconferencing, and facsimile transmission, in which a certain amount of error is an acceptable trade-off for increased compression performance. originally, pgf has been designed to quickly and progressively decode lossy compressed aerial images. a lossy compression mode has been preferred, because in an application like a terrain explorer texture data (e.g., aerial orthophotos) is usually mid-mapped filtered and therefore lossy mapped onto the terrain surface. in addition, decoding lossy compressed images is usually faster than decoding lossless compressed images.

in the next test series we evaluate the lossy compression efficiency of pgf. one of the best competitors in this area is for sure jpeg 2000. since jpeg 2000 has two different filters, we used the one with the better trade-off between compression efficiency and runtime. on our machine the 5/3 filter set has a better trade-off than the other. however, jpeg 2000 has in both cases a remarkable good compression efficiency for very high compression ratios but also a very poor encoding and decoding speed. the other competitor is jpeg. jpeg is one of the most popular image file formats.

it is very fast and has a reasonably good compression efficiency for a wide range of compression ratios. the drawbacks of jpeg are the missing lossless compression and the often missing progressive decoding. fig. 4 depicts the average rate-distortion behavior for the images in the kodak test set when fixed (i.e., nonprogressive) lossy compression is used. the psnr of pgf is on average 3% smaller than the psnr of jpeg 2000, but 3% better than jpeg.

these results are also qualitative valid for our pgf test set and they are characteristic for aerial ortho-photos and natural images. because of the design of pgf we already know that pgf does not reach the compression efficiency of jpeg 2000. however, we are interested in the trade-off between compression efficiency and runtime. to report this trade-off we show in table 4 a comparison between jpeg 2000 and pgf and in fig. 5 (on page 8) we show for the same test series as in fig. 4 the corresponding average decoding times in relation to compression ratios.table 4 contains for seven different compression ratios (mean values over the compression ratios of the eight images of the kodak test set) the corresponding average encoding and decoding times in relation to the average psnr values. in case of pgf the encoding time is always slightly longer than the corresponding decoding time. the reason for that is that the actual encoding phase (cf. subsection 2.4.2) takes slightly longer than the corresponding decoding phase. for six of seven ratios the psnr difference between jpeg 2000 and pgf is within 3% of the psnr of jpeg 2000. only in the first row is the difference larger (21%), but because a psnr of 50 corresponds to an almost perfect image quality the large psnr difference corresponds with an almost undiscoverable visual difference. the price they pay in jpeg 2000 for the 3% more psnr is very high. the creation of a pgf is five to twenty times faster than the creation of a corresponding jpeg 2000 file, and the decoding of the created pgf is still five to ten times faster than the decoding of the jpeg 2000 file. this gain in speed is remarkable, especially in areas where time is more important than quality, maybe for instance in real-time computation.

in fig. 5 we see that the price we pay in pgf for the 3% more psnr than jpeg is low: for small compression ratios (< 9) decoding in pgf takes two times longer than jpeg and for higher compression ratios (> 30) it takes only ten percent longer than jpeg. these test results are characteristic for both natural images and aerial ortho-photos. again, in the third test series we only use the ‘lena’ image. we run our lossy coder with six different quantization parameters and measure the psnr in relation to the resulting compression ratios. the results (ratio: psnr) are:

2.lossless image compression :

lossless image compression is the only acceptable amount of data reduction. it provides low compression ratio while compared to lossy. in lossless image compression techniques are composed of two relatively independent operations: (1) devising an alternative representation of the image in which its interpixel redundancies are reduced and (2) coding the representation to eliminate coding redundancies.

Lossless image compression is useful in applications such as medical imaginary, business documents and satellite images.table 2 summarizes the lossless compression efficiency and table 3 the coding times of the pgf test set. for winzip we only provide average runtime values, because of missing source code we have to use an interactive testing procedure with runtimes measured by hand. all other values are measured in batch mode.

In table 2 it can be seen that in almost all cases the best compression ratio is obtained by jpeg 2000, followed by pgf, jpeg-ls, and png. this result is different to the result in [sea+00], where the best performance for a similar test set has been reported for jpeg-ls. pgf performs between 0.5% (woman) and 21.3% (logo) worse than jpeg 2000. on average it is almost 15% worse. the two exceptions to the general trend are the ‘compound’ and the ‘logo’ images. both images contain for the most part black text on a white background. for this type of images, jpeg-ls and in particular winzip and png provide much larger compression ratios. however, in average png performs the best, which is also reported in [sea+00].

These results show, that as far as lossless compression is concerned, pgf performs reasonably well on natural and aerial images. in specific types of images such as ‘compound’ and ‘logo’ pgf is outperformed by far in png.

Table 3 shows the encoding (enc) and decoding (dec) times (measured in seconds) for the same algorithms and images as in table 2. jpeg 2000 and pgf are both symmetric algorithms, while winzip, jpeg-ls and in particular png are asymmetric with a clearly shorter decoding than encoding time. jpeg 2000, the slowest in encoding and decoding, takes more than four times longer than pgf. this speed gain is due to the simpler coding phase of pgf.

jpeg-ls is slightly slower than pgf during encoding, but slightly faster in decoding images.

WinZip and png decode even more faster than jpeg-ls, but their encoding times are also worse. pgf seems to be the best compromise between encoding and decoding times.

our pgf test set clearly shows that pgf in lossless mode is best suited for natural images and aerial ortho photos. pgf is the only algorithm that encodes the three mega byte large aerial ortho photo in less than second without a real loss of compression efficiency. for this particular image the efficiency loss is less than three percent compared to the best. these results should be underlined with our second test set, the kodak test set.

**CLASSIFICATION OF IMAGES:**

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

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

**BINARY IMAGE:**

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

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

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

**GRAY SCALE IMAGE**

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

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

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

**COLOUR IMAGE:**

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

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

From the above figure, colors are coded on three bytes representing their decomposition on the three primary colors. It sounds obvious to a mathematician to immediately interpret colors as vectors in a three dimension space where each axis stands for one of the primary colors. Therefore we will benefit of most of the geometric mathematical concepts to deal with our colors, such as norms, scalar product, projection, rotation or distance.

**LITERATURE SURVEY:**

X. Liu, P. H. Tu, J. Rittscher, A. Perera and N. Krahnstoever, "Detecting and counting people in surveillance applications,"A new video-based people counting method has been proposed in this research that incorporates a number of new features. We can be certain that the items being tallied are solely individuals since we can see their faces. When the face scales drastically shift, a scale-invariant Kalman filter is presented. Face occlusions are also handled using a kernelbased object tracking algorithm combined with it. Finally, we've suggested a method for counting individuals based on the automated categorization of possible face trajectories's. The Earth Mover's Distance measures the degree of similarity between the spots in the histogram. By eliminating incorrect trajectories, not only can the system's counting accuracy be improved, but the automated categorization process can also eliminate the need for humans to develop counting rules based on empirical evidence. An accuracy of roughly 93 percent was achieved by evaluating our experimental data, which exhibited excellent performance on these many areas. It is our goal to adapt our technology to more complicated environments, such as outdoors, where light varies greatly. In addition, online learning and the adoption of a more robust classifier will enhance the categorization of face trajectory to better match varied settings. N. R. Borkar and S. Kuwelkar, "Real-time implementation of face recognition system,A Face Recognition System Based on PCA and LDA is presented in this study. Accuracy of 97% has been achieved by utilising the raspberry pi 3 module in conjunction with these two approaches. To be employed in identification systems, the Raspberry pi 3 module is a low-cost, lightweight, and tiny component. I've learned a lot about facial recognition algorithms with this project, and I'm glad I had the chance to do so. This study has also shown me that improving the accuracy of a Face recognition system by integrating many approaches. In the future, robots might be made more human-like by incorporating this face-recognition technology. A. Das, M. Wasif Ansari and R. Basak, "Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV,"At the beginning of this article, we briefly discussed the purpose of our research. The model's learning and performance challenges were then shown. Sophisticated methodologies and ML tools have been used to reach a reasonable level of accuracy. It may be used to a wide range of uses. Considering the Covid-19 situation, wearing a mask may become a necessity in the near future.. Masks are required by several public service providers in order to use their services. The public health care system will greatly benefit from the model's implementation. In the future, it will be possible to tell whether someone is wearing the mask correctly if the technology is included. A more advanced version of the model might determine whether or not a mask is susceptible to viral infection, i.e. whether or not it is surgical, N95, or not.

**LITERATURE SURVEY:**

Literature [survey](http://www.blurtit.com/q876299.html) is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then next step is to determine which operating system and language can be used for developing the tool. Once the [programmers](http://www.blurtit.com/q876299.html) start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from [book](http://www.blurtit.com/q876299.html) or from websites. Before building the system the above considerations are taken into account for developing the proposed system.

A literature review is a body of text that aims to review the critical points of current knowledge including substantive findings as well as theoretical and methodological contributions to a particular topic. Literature reviews are [secondary sources](http://en.wikipedia.org/wiki/Secondary_sources), and as such, do not report any new or original experimental work. Also, a literature review can be interpreted as a review of an abstract accomplishment.

Most often associated with academic-oriented literature, such as a [thesis](http://en.wikipedia.org/wiki/Thesis), a literature review usually precedes a research proposal and results section. Its main goal is to situate the current study within the body of literature and to provide context for the particular reader.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SNO | TITLE | AUTHOR | YEAR | TECHNOLOGY |
| 1 | Mutual Enhancement for Detection of Multiple Logos in Sports Videos | Y. Liao, X. Lu, C. Zhang, Y. Wang | 2017 | Feature  extraction  Image  analysis  image texture  object  recognition  supervised learning. |
| 2 | Automatic graphic logo detection via Fast Region-based Convolutional Networks | G. Oliveira,  X. Frazão, A. Pimentel, and B. Ribeiro. | 2016 | Image processing  Machine learning |
| 3 | A hierarchical scheme for vehicle make and model recognition from frontal images of vehicles,” IEEE Transactions on Intelligent Transportation Systems. | L. Lu and H. Huang, | 2018 | Computer vision  Data privacy  Object detection |
| 4 | Deep learning for logo recognition | Y. Zhang et al | 2017 | Object recognition  Artificial intelligence  reinforcement |
| 5 | Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition | K. He, X. Zhang, S. Ren, and J. Sun | 2020 | Feature extraction  Image fusion  Logo detection |

**EXISTING SYSTEM:**

* SUPPORT VECTOR MACHINE
* PRINCIPAL COMPONENT ANALYSIS

**SVM (Support Vector Machine):**

More formally, a support-vector machine constructs a hyper plane or set of hyper planes in a [high-](https://en.wikipedia.org/wiki/High-dimensional_space) or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyper plane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier. Whereas the original problem may be stated in a finite-dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was propose that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products of pairs of input data vectors may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function selected to suit the problem. The hyper planes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant, where such a set of vectors is an orthogonal (and thus minimal) set of vectors that defines a hyper plane.

**Principal Component analysis**

            PCA is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called **principal components**. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to (i.e., uncorrelated with) the preceding components. Principal components are guaranteed to be independent only if the data set is jointly normally distributed. PCA is sensitive to the relative scaling of the original variables. Depending on the field of application, it is also named the discrete Karhunen–Loève transform (KLT), the Hotelling transform or proper orthogonal decomposition (POD).

**DISADVANTAGES:**

* In this doesn’t work in real time due to many customized processing steps involved.
* Hence, considering Deep CNNs is quite intuitive, given the recent success in basic computer vision problem.

**PROPOSEDSYSTEM:**

* Pre-processing
* Blob detection
* Yolo

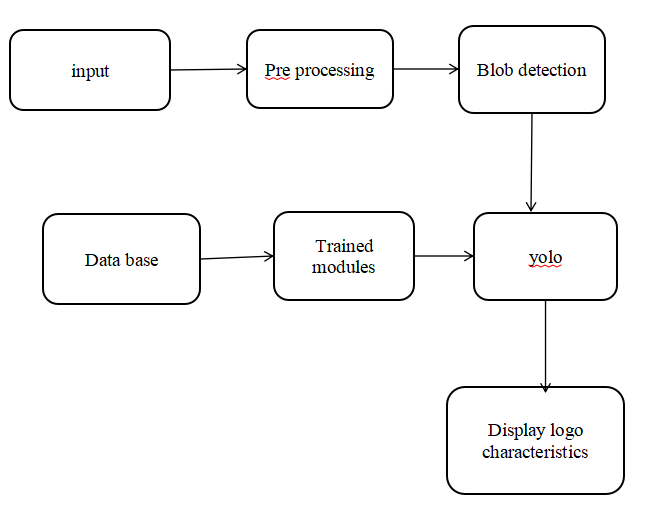
**ADVANTAGES:**

* It will get high accuracy results
* Recognize logo correctly

**APPLICATIONS:**

* Retail
* Marketing and Advertising
* Sports and Entertainment
* Smart Cities

**BLOCK DIAGRAM:**



**DESCRIPTION MODULES:**

**IMAGE ACQUISITION:**

 Image acquisition is the first step in image processing. This step is also known as pre-processing in image processing. It involves retrieving the image from a source, usually a hardware-based source This can be done via hardware systems such as cameras, encoders, sensors, etc.

**PREPROCESSING:**

Some powerful image pre-processing techniques include noise reduction, contrast enhancement, image resizing, color correction, segmentation, feature extraction, etc. It is an essential step in image analysis that helps enhance the data in images and reduce clutter.

**BLOB DETECTION:**

In computer vision, blob detection methods are aimed at detecting regions in a digital image that differ in properties, such as brightness or color, compared to surrounding regions.

**YOLO:**

YOLO is an abbreviation for the term ‘You Only Look Once’. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects.

This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously.

The YOLO algorithm consists of various variants. Some of the common ones include tiny YOLO and YOLOv3.

**UML DIAGRAMS:**

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

* actors
* business processes
* (logical) components
* activities
* programming language statements
* database schemas, and
* Reusable software components.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

**Sequence Diagram:**

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

* **Activity Diagrams-:**
* Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency.In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**Usecase diagram:**

* UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.
* UML was created by Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.
* OMG is continuously putting effort to make a truly industry standard.
* UML stands for **U**nified **M**odeling **L**anguage.
* UML is a pictorial language used to make software blue prints

**Collaboration**

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing  
**Class diagram**

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling.[1] The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

In the diagram, classes are represented with boxes that contain three compartments:

The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.

The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.

The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

**Component diagram**

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

Thus from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

The purpose of the component diagram can be summarized as −

* Visualize the components of a system.
* Construct executables by using forward and reverse engineering.
* Describe the organization and relationships of the components.

**ER Diagram**

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases.

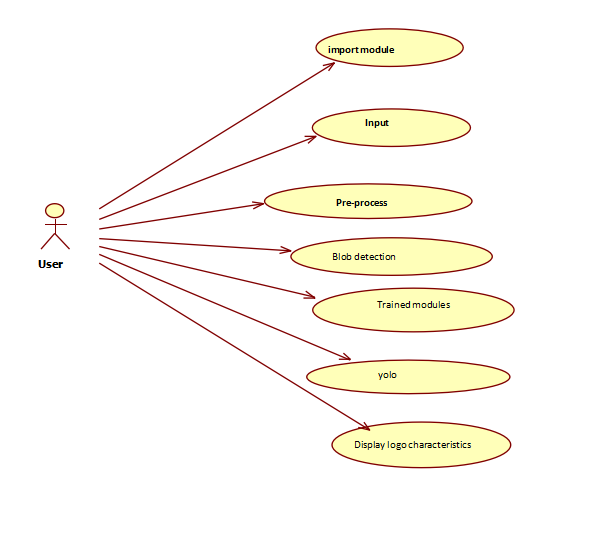
ER diagrams are used to sketch out the design of a database.

**Data flow diagram**

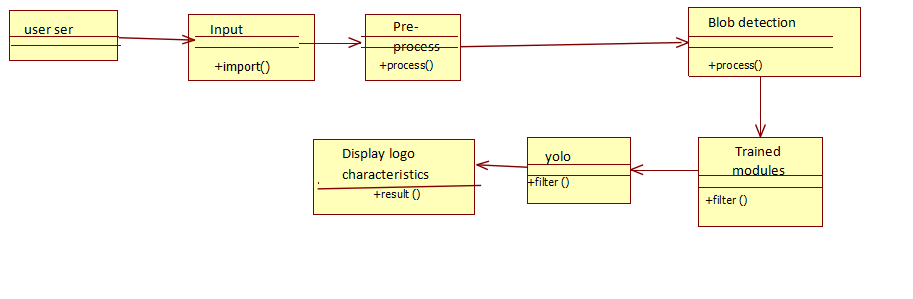
Also known as DFD, Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow

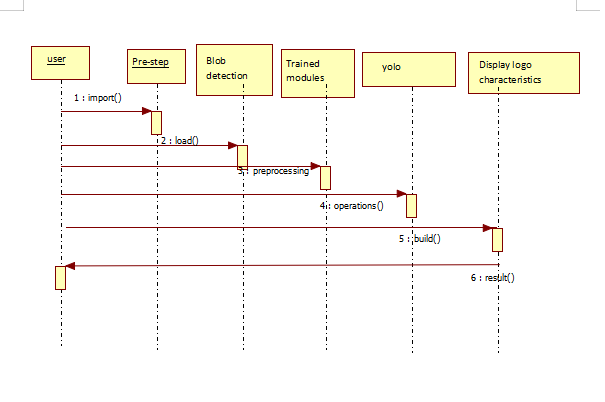
**USE CASE:**



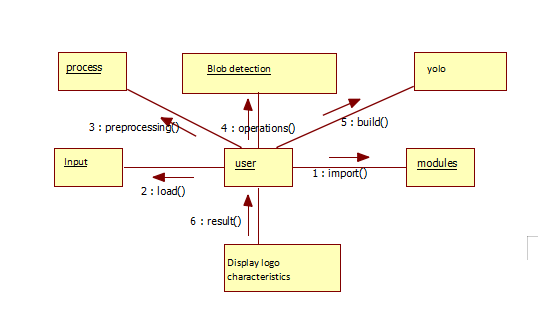
**CLASS DIAGRAM:-**



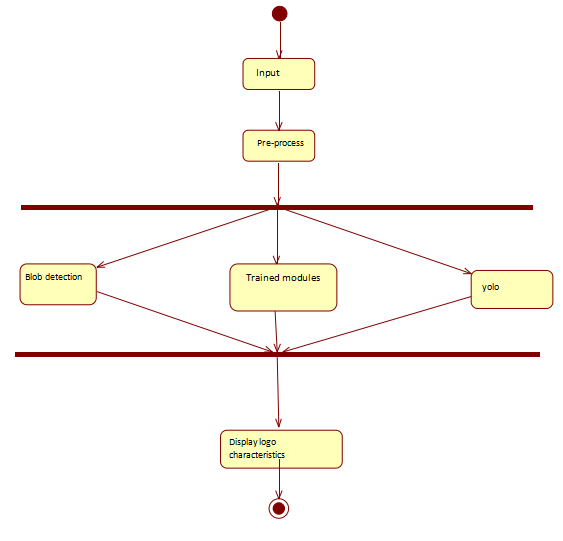
**SEQUENCE DIAGRAM:**



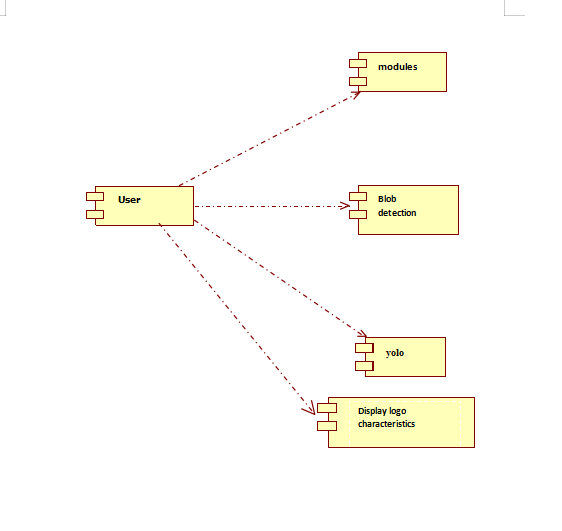
**COLLABORATION:**

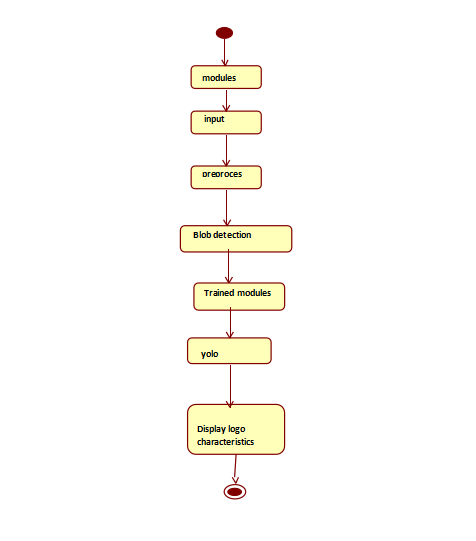


**ACTIVITY DIAGRAM:**



**COMPONENT:**





**SOFTWARE AND HARDWARE REQUIREMENTS:**

**SOFTWARE REQUIREMENT:**

* PYTHON  IDLE 3.9.7

**PYTHON**

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

**PYTHON FEATURES:**

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

**PYTHON LIBRARIES**

Python's large [standard library](https://en.wikipedia.org/wiki/Standard_library), commonly cited as one of its greatest strengths, provides tools suited too many tasks. For Internet-facing applications, many standard formats and protocols such as [MIME](https://en.wikipedia.org/wiki/MIME) and [HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) are supported. It

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

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

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

**DEPLOYMENT ENVIRONMENT**

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

which the user enters statements sequentially and receives results immediately.

Other shells, including [IDLE](https://en.wikipedia.org/wiki/IDLE_(Python)) and [IPython](https://en.wikipedia.org/wiki/IPython), add further abilities such as auto-completion, session state retention and [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting).

As well as standard desktop [integrated development environments](https://en.wikipedia.org/wiki/Integrated_development_environment), there are [Web browser](https://en.wikipedia.org/wiki/Web_browser)-based IDEs; [Sage Math](https://en.wikipedia.org/wiki/SageMath) (intended for developing science and math-related Python programs); [Python Anywhere](https://en.wikipedia.org/wiki/PythonAnywhere), a browser-based IDE and hosting environment; and Canopy IDE, a commercial Python IDE emphasizing scientific computing.

**OPENCV-PYTHON**

 Python is a general purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability. Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation. And the support of Numpy makes the task more easier. Numpy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this. So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

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

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

**ARRAY ATTRIBUTES**

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

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

**SCALARS**

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

**METHODS**

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

**DATA TYPE OBJECTS (DTYPE)**

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

**ANACONDA NAVIGATOR**

         Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, mac OS and Linux.

**Navigator:**

      In order to run, many scientific packages depend on specific versions of other          packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator.

APPLICATIONS

The following applications are available by default in Navigator:

* JupyterLab
* Jupyter Notebook
* QTConsole
* Spyder
* VSCode
* Glueviz
* Orange 3 App
* Rodeo
* RStudio

Advanced conda users can also build your own Navigator applications

**Python**

Python is a general-purpose, versatile and popular programming language. It's great as a first language because it is concise and easy to read, and it is also a good language to have in any programmer's stack as it can be used for everything from web development to soitware development and scientific applications.   
  
It has simple easy-to-use syntax, making it the perfect language for someone trying to learn computer programming for the first time.  **Features of Python**   
  
A simple language which is easier to learn, Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax. If you are a newbie, it's a great choice to start your journey with Python.

* **Free and open source**

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software’s written in it, you can even make changes to the Python's source code. Python has a large community constantly improving it in each iteration.

* **Portability**  
  You can move Python programs from one platform to another, and run it without any changes.   
  It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.
* **Extensible and Embeddable**   
  Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code. This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.
* **A high-level, interpreted language**

Unlike  C/C++,  you don’t have to worry about daunting tasks like memory management, garbage collection and so on.   
Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower level operations.

* **Large standard libraries to solve common tasks**   
  Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server You can use MySQLdb library using import MySQL db Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.
* **Object-oriented**

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.   
With OOP, you are able to divide these complex problems into smaller sets by creating object.

**HARDWARE REQUIREMENT :**

* **Monitor**
* **Hdd: 1tb**
* **Ram: 4gb ram 64bit**

**CONCLUSION:**

The use of deep learning for logo detection has shown promising results and significant advancements in computer vision applications. Deep learning models, particularly convolutional neural networks (CNNs), have been widely employed due to their ability to learn complex patterns and features from image data.

**FUTURE SCOPE:**

As AI and machine learning continue to advance, the accuracy, speed, and applications of logo detection are likely to expand further, offering innovative solutions across diverse industries and everyday life. However, ethical considerations regarding privacy, copyright, and data usage will also need careful attention during the development and implementation of such technologies.

**REFERENCE:**

[1] L. Lu and H. Huang, “A hierarchical scheme for vehicle make and model recognition from frontal images of vehicles,” IEEE Transactions on Intelligent Transportation Systems, 2018. doi: 10.1109/TITS.2018.2835471.

 [2] Simone Bianco, Marco Buzzelli, Davide Mazzini, and Raimondo Schettini, “Deep learning for logo recognition,” Computer Vision and Pattern Recognition, May 2017. doi: 10.1016/j.neucom.2017.03.051.

 [3] L. Huan, W. Li, and Q. Yujian, “Vehicle Logo Retrieval Based on Hough Transform and Deep Learning,” IEEE International Conference on Computer Vision Workshops (ICCVW) 2017, Venice, 2017, pp. 967–973. doi: 10.1109/ICCVW.2017.118.

 [4] R. Chen, M. Hawes, O. Isupova, L. Mihaylova, and H. Zhu, “Online vehicle logo recognition using Cauchy prior logistic regression,” 20th International Conference on Information Fusion (Fusion) 2017, Xi’an, 2017, pp. 1–8. doi: 10.23919/ICIF.2017.8009720.

[5] T. Sridevi, P. Swapna, and K. Harinath, “Vehicle Identification Based on the Model,” IEEE 7th International Advance Computing Conference (IACC) 2017, Hyderabad, 2017, pp. 566–571. doi: 10.1109/IACC.2017.0122.

 [6] Y. Xia, J. Feng, and B. Zhang, “Vehicle Logo Recognition and attributes prediction by multi-task learning with CNN,” 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD) 2016, Changsha, 2016, pp. 668–672. doi: 10.1109/FSKD.2016.7603254

 [7] F. Tafazzoli and H. Frigui, “Vehicle make and model recognition using local features and logo detection,” International Symposium on Signal, Image, Video and Communications (ISIVC) 2016, Tunis, 2016, pp. 353–358. doi: 10.1109/ISIVC.2016.7894014

 [8] Cong Chen, Xiaobo Lu, Shengqin Jiang, and Jiaji Song, “An effective vehicle logo recognition method for road surveillance images,” 2nd IEEE International Conference on Computer and Communications (ICCC) 2016, Chengdu, 2016, pp. 728–732. doi: 10.1109/CompComm.2016.7924798.

[9] Ming Liang and Xiaolin Hu, “Recurrent convolutional neural network for object recognition,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015, Boston, MA, 2015, pp. 3367–3375. doi: 10.1109/CVPR.2015.7298958.

[10] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao, “Recurrent convolutional neural networks for text classification,” Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, Texas, 25–30 Jan. 2015, pp. 2267–2273.