**Insurance Amount Prediction Based on Accidental Car Damage Level using AI**

**ABSTRACT:**

Predicting the insurance amount for accidental car damage is a critical process in the insurance industry, ensuring fair and prompt claim settlements. This project introduces an innovative approach that combines Generative Adversarial Networks (GANs) and deep learning techniques to predict insurance payouts based on the severity of car damage. The system employs a two-part GAN architecture: the generator creates synthetic damaged car images with varying levels of severity, while the discriminator distinguishes between real and synthetic images, improving the model's ability to recognize nuanced damage patterns. Additionally, the generated images are used to augment the dataset, improving the performance of the predictive model. The extracted features from these images, alongside structured data such as car make, model, and accident details, are then fed into a regression model to estimate the insurance amount. This AI-driven approach enhances prediction accuracy, helps in generalizing the model for unseen data, and reduces the reliance on large labeled datasets by leveraging the power of GANs to generate diverse accident scenarios. The proposed system improves the efficiency and fairness of insurance claim assessments while reducing fraud and ensuring faster processing.

**OBJECTIVE:**

The objective of this project is to develop an AI-based system that predicts insurance claim amounts based on the severity of accidental car damage. By leveraging deep learning techniques, the model ensures accurate and efficient estimation, aiding insurers in fair claim assessments.

**PROBLEM STATEMENT:**

The traditional method of assessing car accident damage for insurance claims is often time-consuming, subjective, and prone to human error. Manual evaluation can lead to inconsistencies in claim amounts, resulting in disputes between insurers and policyholders. An AI-driven system can automate the assessment process by analyzing car damage images and predicting insurance amounts accurately. This approach enhances efficiency, reduces processing time, and ensures fair compensation based on damage severity. Implementing deep learning models improves prediction accuracy, leading to a more transparent and reliable insurance claim process.

**INTRODUCTION :**

In the insurance industry, accurately predicting the claim amount based on the severity of car damage is essential for fair, transparent, and efficient claim processing. Traditional methods of assessment are time-consuming, prone to human error, and often rely on limited data, which can lead to inaccuracies. With the advent of artificial intelligence, particularly deep learning, there is an opportunity to enhance the prediction process by automating damage assessment and insurance amount estimation. This project proposes the use of Generative Adversarial Networks (GANs) combined with Convolutional Neural Networks (CNNs) to analyze car damage images and predict the corresponding insurance claim amounts. GANs help generate synthetic images of car damage to augment training data, while CNNs extract important features related to damage severity. A regression model is then used to predict the insurance payout based on these extracted features. This AI-driven approach aims to streamline the claim process, improve accuracy, reduce fraud, and provide consistent insurance evaluations.

**DOMAIN OVERVIEW:**

**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.

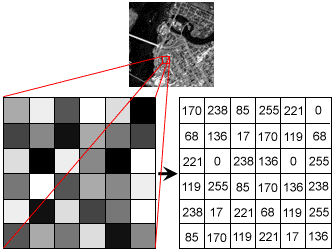
Image is a two-dimensional, such as a photograph, screen display, and as well as a three-dimensional, such as a statue. They may be captured by optical devices—such as cameras, mirrors, lenses, telescopes, microscopes, etc. and natural objects and phenomena, such as the human eye or water surfaces.

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.

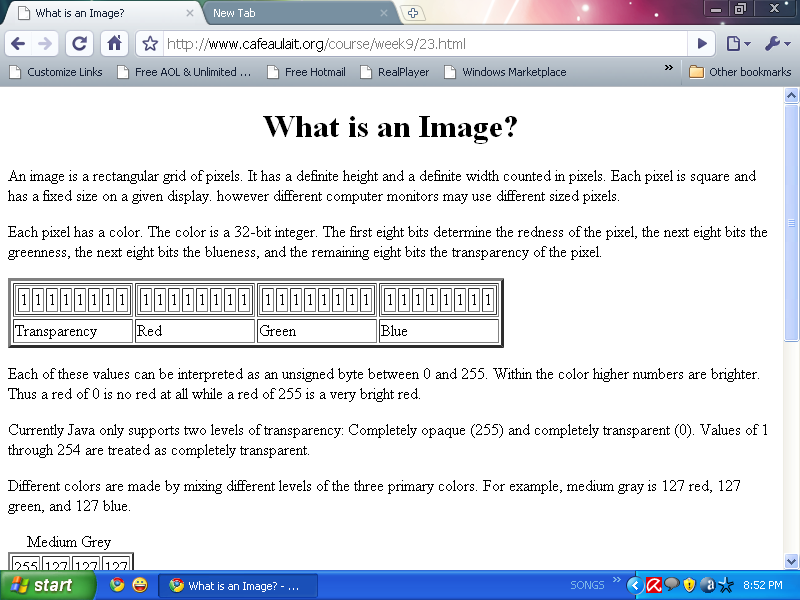
**Fig: Colour image to Gray scale Conversion Process**

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.



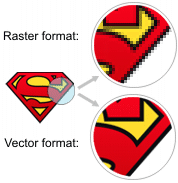
**Fig: BIT Transferred for Red, Green and Blue plane (24bit=8bit red;8-bit green;8bit blue)**

**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.



**Fig: Horizontal and Vertical Process**

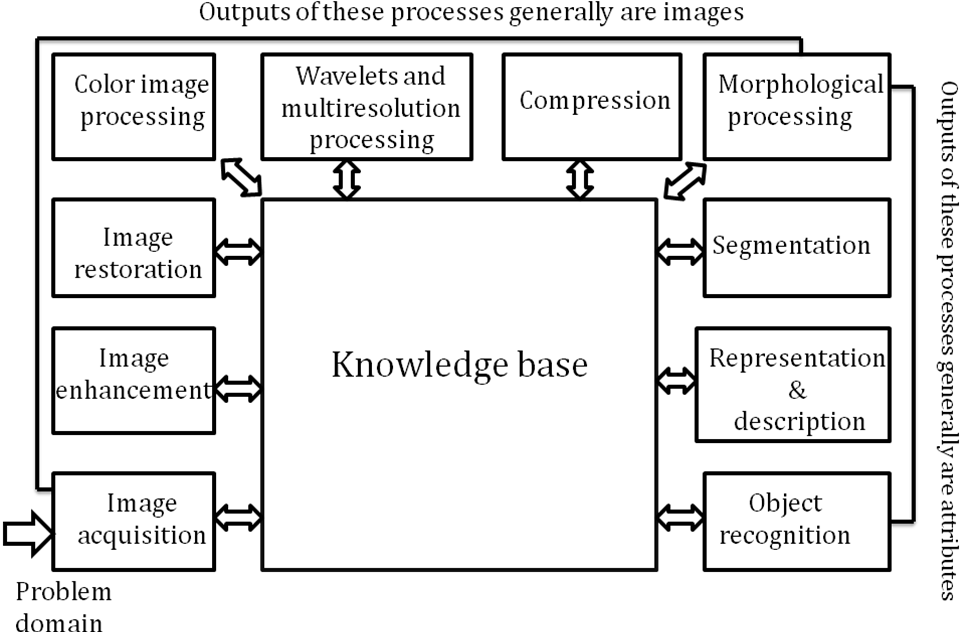
In addition to straight image formats, Metafile formats are portable formats which can include both raster and vector information. The metafile format is an intermediate format. Most Windows applications open metafiles and then save them in their own native format.

**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.

**FUNDAMENTAL STEPS IN DIGITAL IMAGE PROCESSING:**

**Fig: Basics steps of image Processing**

**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:**

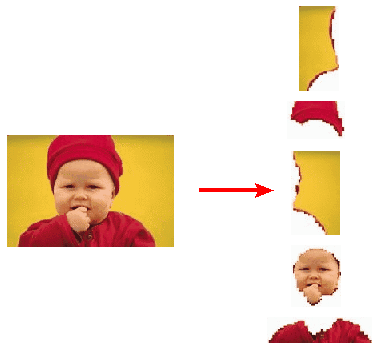
The use of color in image processing is motivated by two principal factors. First, color is a powerful descriptor that often simplifies object identification and extraction from a scene. Second, humans can discern thousands of color shades and intensities, compared to about only two dozen shades of gray.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

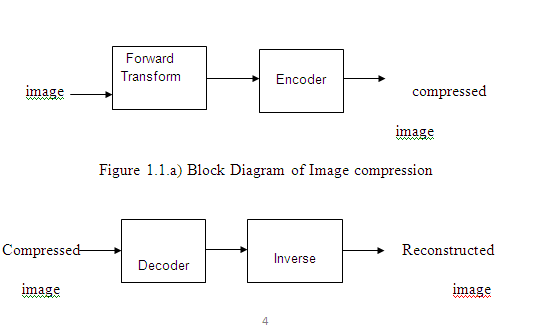


Fig:1.1b) Decompression Process for Image

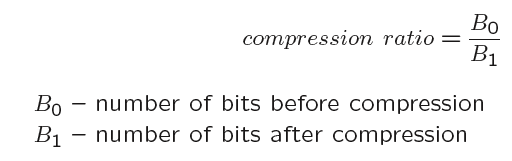
**Image Compression Types**

There are two types’ image compression techniques.

1. Lossy Image compression

2. Lossless Image compression

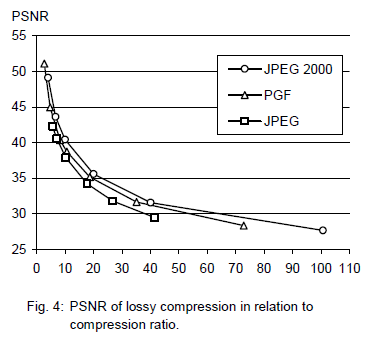
Compression ratio:



**1. Lossy Image compression :**

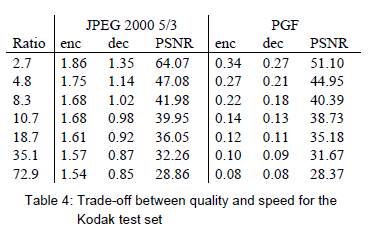
Lossy compression provides higher levels of data reduction but result in a less than perfect reproduction of the original image. It provides high compression ratio. lossy image compression is useful in applications such as broadcast television, videoconferencing, and facsimile transmission, in which 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.

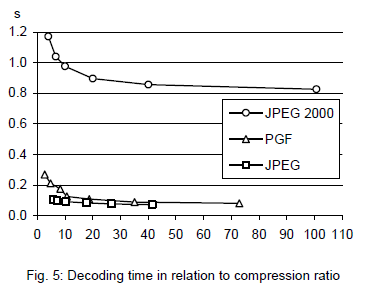


It is very fast and has a reasonably good compression efficiency for a wide range of compression ratios. The drawbacks of JPEG are the missing lossless compression and the often missing progressive decoding. Fig. 4 depicts the average rate-distortion behavior for the images in the Kodak test set when 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.



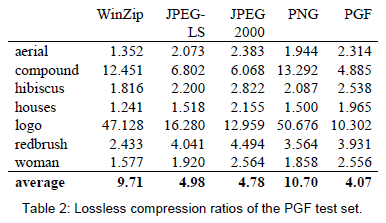
In Fig. 5 we see that the price we pay in PGF for the 3% more PSNR than JPEG is low: for small compression ratios (< 9) decoding in PGF takes two times longer than JPEG and for higher compression ratios (> 30) it takes only ten percent longer than JPEG. These test results are characteristic for both 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.

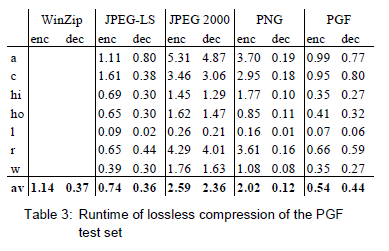


Table 3 shows the encoding (enc) and decoding (dec) times (measured in seconds) for the same algorithms and images as in Table 2. JPEG 2000 and PGF are both symmetric algorithms, while WinZip, JPEG-LS and in particular PNG are asymmetric with a clearly shorter decoding than encoding time. JPEG 2000, 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.

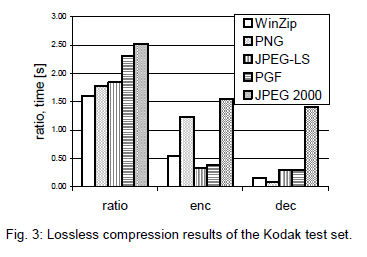


Fig. 3 shows the averages of the compression ratios (ratio), encoding (enc), and decoding (dec) times over all eight images. JPEG 2000 shows in this test set the best compression efficiency followed by PGF, JPEG-LS, PNG, and WinZip. In average PGF is eight percent worse than JPEG 2000. The fact that JPEG 2000 has a better lossless compression ratio than PGF does not surprise,

because JPEG 2000 is more quality driven than PGF.

However, it is remarkable that PGF is clearly better than JPEG-LS (+21%) and PNG (+23%) for natural images. JPEG-LS shows in the Kodak test set also a symmetric encoding and decoding time behaviour. It is encoding and decoding times are almost equal to PGF. Only PNG and WinZip can faster decode than PGF, but they also take longer than PGF to encode.

If both compression efficiency and runtime is important, then PGF is clearly the best of the tested algorithms for lossless compression of natural images and aerial ortho photos. In the third test we perform our lossless coder on the ‘Lena’ image.

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

Correspondingly, these combinations attempt to strike a winning tradeoff: be flexible and hence bring tolerance toward intraclass variation, while also being discriminative enough to be robust to background clutter and interclass similarity. An important feature of our contour-based recognition approach is that it affords us substantial flexibility to incorporate additional image information. Specifically, we extend the contour-based recognition method and propose a new hybrid recognition method which exploits shape tokens and SIFT features as recognition cues. Shape-tokens and SIFT features are largely orthogonal, where the former corresponds to shape boundaries and the latter to sparse salient image patches. Here, each learned combination can comprise features that are either 1) purely shape-tokens, 2) purely SIFT features, or 3) a mixture of shape-tokens and SIFT features. The number and types of features to be combined together are learned automatically from training images, and represent the more discriminative ones based on the training set. Consequently, by imparting these two degrees of variability (in both the number and the types of features) to a combination, we empower it with even greater flexibility and discriminative potential. A shorter version of this paper appeared in [9].

**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

|  |
| --- |
| http://images.gamedev.net/features/programming/imageproc/image004.gif |

**Fig.1 Hue Saturation Process of RGB SCALE Image**

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.

**SOFTWARE REQUIREMENTS**

**Python: Python** Is An Object-Oriented, High Level Language, Interpreted, Dynamic And Multipurpose Programming Language.

Python Is Easy To Learn Yet Powerful And Versatile Scripting Language Which Makes It Attractive For Application Development.

Python's Syntax And Dynamic Typing With Its Interpreted Nature, Make It An Ideal Language For Scripting And Rapid Application Development In Many Areas.

Python Supports Multiple Programming Pattern, Including Object Oriented Programming, Imperative And Functional Programming Or Procedural Styles.

Python Is Not Intended To Work On Special Area Such As Web Programming. That Is Why It Is Known As Multipurpose Because It Can Be Used With Web, Enterprise, 3d Cad Etc.

We Don't Need To Use Data Types To Declare Variable Because It Is Dynamically Typed So We Can Write A=10 To Declare An Integer Value In A Variable.

Python Makes The Development And Debugging Fast Because There Is No Compilation Step Included In Python Development And Edit-Test-Debug Cycle Is Very Fast.

2. Python Features

#### 1) Easy To Use:

Python Is Easy To Very Easy To Use And High Level Language. Thus It Is Programmer-Friendly Language.

#### 2) Expressive Language:

Python Language Is More Expressive. The Sense Of Expressive Is The Code Is Easily Understandable.

#### 3) Interpreted Language:

Python Is An Interpreted Language I.E. Interpreter Executes The Code Line By Line At A Time. This Makes Debugging Easy And Thus Suitable For Beginners.

#### 4) Cross-Platform Language:

Python Can Run Equally On Different Platforms Such As Windows, Linux, Unix , Macintosh Etc. Thus, Python Is A Portable Language.

#### 5) Free And Open Source:

Python Language Is Freely Available(Www.Python.Org).The Source-Code Is Also Available. Therefore It Is Open Source.

#### 6) Object-Oriented Language:

Python Supports Object Oriented Language. Concept Of Classes And Objects Comes Into Existence.

#### 7) Extensible:

It Implies That Other Languages Such As C/C++ Can Be Used To Compile The Code And Thus It Can Be Used Further In Your Python Code.

#### 8) Large Standard Library:

Python Has A Large And Broad Library.

#### 9) Gui Programming:

Graphical User Interfaces Can Be Developed Using Python.

#### 10) Integrated:

It Can Be Easily Integrated With Languages Like C, C++, Java Etc.

3. Python History

* Python Laid Its Foundation In The Late 1980s.
* The Implementation Of Python Was Started In The December 1989 By **Guido Van Rossum** At Cwi In Netherland.
* Abc Programming Language Is Said To Be The Predecessor Of Python Language Which Was Capable Of Exception Handling And Interfacing With Amoeba Operating System.
* Python Is Influenced By Programming Languages Like:
  + Abc Language.
  + Modula-3

# 5. Python Applications

Python As A Whole Can Be Used In Any Sphere Of Development.

Let Us See What Are The Major Regions Where Python Proves To Be Handy.

#### 1) Console Based Application

Python Can Be Used To Develop Console Based Applications. For Example: **Ipython**.

#### 2) Audio Or Video Based Applications

Python Proves Handy In Multimedia Section. Some Of Real Applications Are: Timplayer, Cplay Etc.

#### 3) 3d Cad Applications

Fandango Is A Real Application Which Provides Full Features Of Cad.

#### 4) Web Applications

Python Can Also Be Used To Develop Web Based Application. Some Important Developments Are: Pythonwikiengines, Pocoo, Pythonblogsoftware Etc.

#### 5) Enterprise Applications

Python Can Be Used To Create Applications Which Can Be Used Within An Enterprise Or An Organization. Some Real Time Applications Are: Openerp, Tryton, Picalo Etc.

#### 6) Applications For Images

Using Python Several Application Can Be Developed For Image. Applications Developed Are: Vpython, Gogh, Imgseek Etc.

There Are Several Such Applications Which Can Be Developed Using Python

# 6. Python Example

Python Code Is Simple And Easy To Run. Here Is A Simple Python Code That Will Print "Welcome To Python".

A Simple Python Example Is Given Below.

1. >>> A="Welcome To Python"
2. >>> Print A
3. Welcome To Python
4. >>>

**Explanation:**

* Here We Are Using Idle To Write The Python Code. Detail Explanation To Run Code Is Given In Execute Python Section.
* A Variable Is Defined Named "A" Which Holds "Welcome To Python".
* "Print" Statement Is Used To Print The Content. Therefore "Print A" Statement Will Print The Content Of The Variable. Therefore, The Output "Welcome To Python" Is Produced.

## Python 3.4 Example

In Python 3.4 Version, You Need To Add Parenthesis () In A String Code To Print It.

1. >>> A=("Welcome To Python Example")
2. >>> Print A
3. Welcome To Python Example
4. >>>

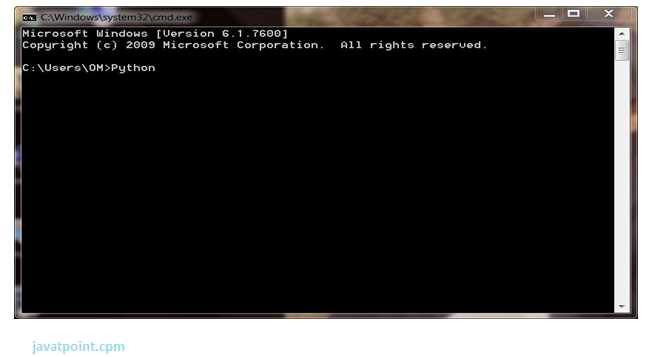
# 7. How To Execute Python

# 7. How To Execute Python

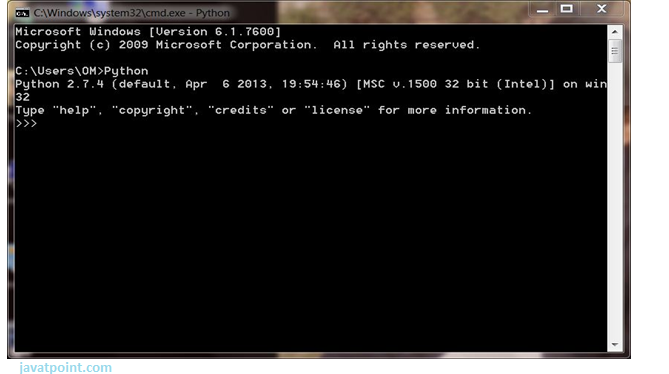
There Are Three Different Ways Of Working In Python:

## 1) Interactive Mode:

You Can Enter Python In The Command Prompt And Start Working With Python.

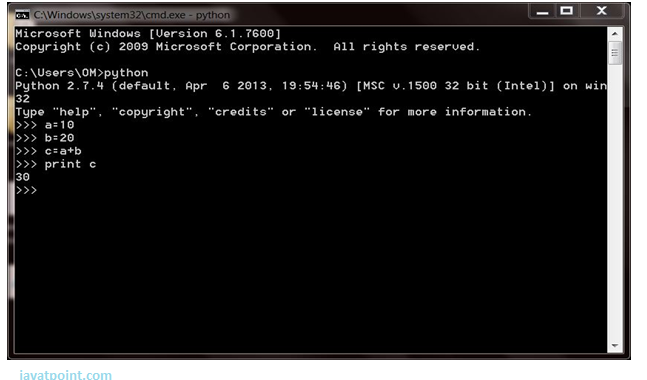


Press Enter Key And The Command Prompt Will Appear Like:



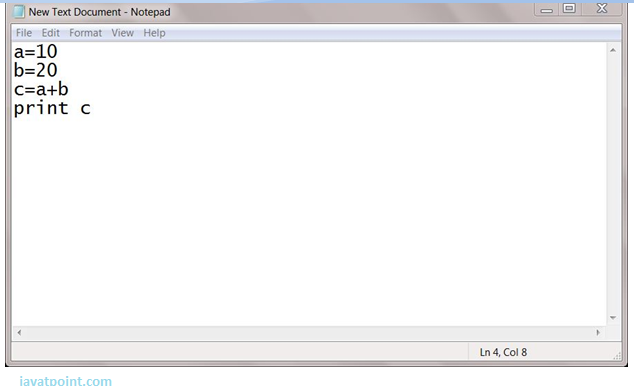
Now You Can Execute Your Python Commands.

**Eg:**

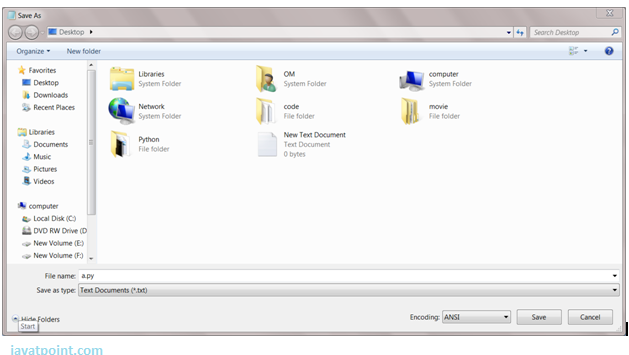


## 2) Script Mode:

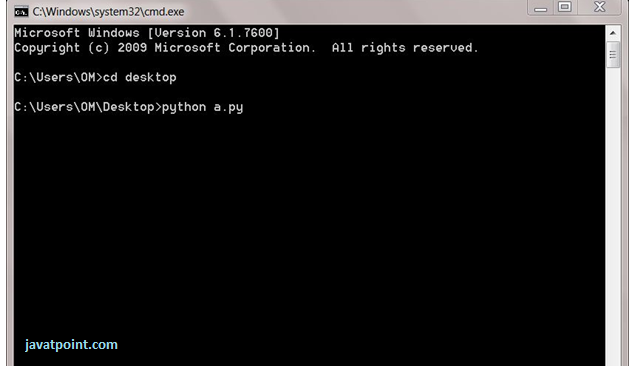
Using Script Mode , You Can Write Your Python Code In A Separate File Using Any Editor Of Your Operating System.



Save It By .Py Extension.



Now Open Command Prompt And Execute It By :



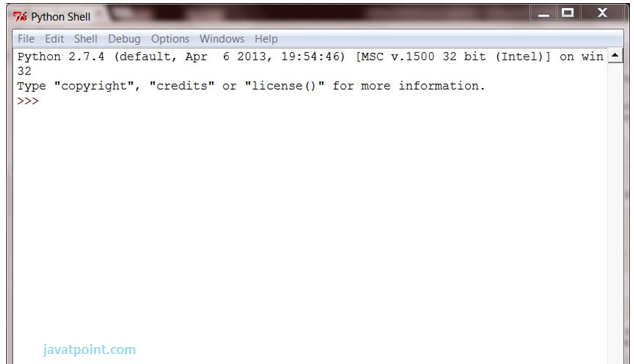
Note: Path In The Command Prompt Should Be Where You Have Saved Your File. In The Above Case File Should Be Saved At Desktop.

## 3) Using Ide: (Integrated Development Environment)

You Can Execute Your Python Code Using A Graphical User Interface (Gui).

All You Need To Do Is:

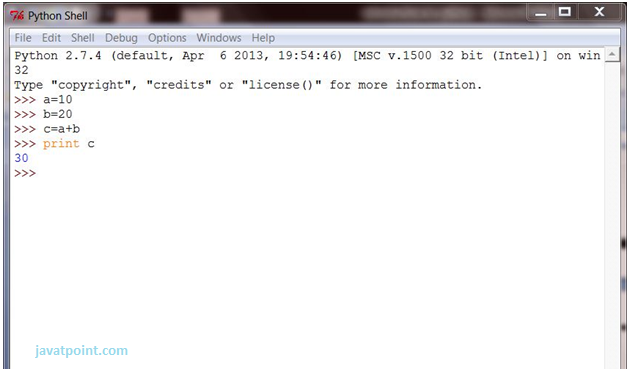
Click On Start Button -> All Programs -> Python -> Idle(Python Gui)



You Can Use Both Interactive As Well As Script Mode In Ide.

**1) Using Interactive Mode:**

Execute Your Python Code On The Python Prompt And It Will Display Result Simultaneously.

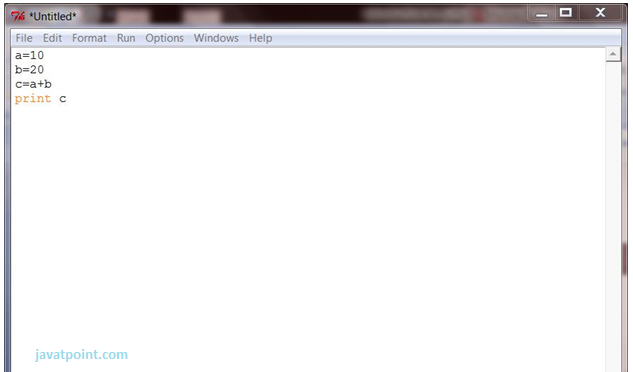


**2) Using Script Mode:**

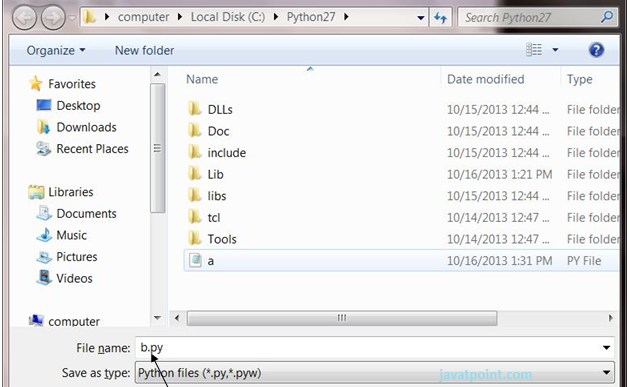
I) Click On Start Button -> All Programs -> Python -> Idle(Python Gui)

Ii) Python Shell Will Be Opened. Now Click On File -> New Window.

A New Editor Will Be Opened . Write Your Python Code Here.



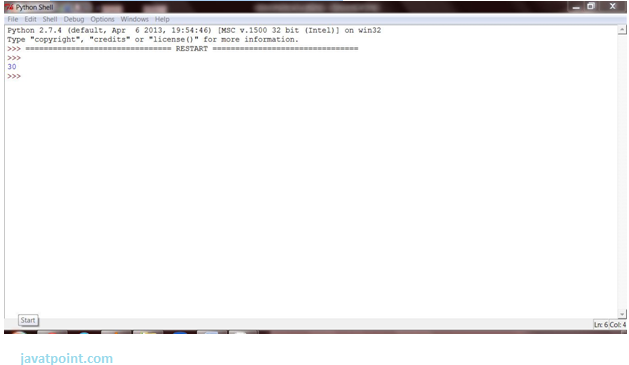
Click On File -> Save As



Run Then Code By Clicking On Run In The Menu Bar.

Run -> Run Module

Result Will Be Displayed On A New Python Shell As:



**1) Using Interactive Mode:**

Execute Your Python Code On The Python Prompt And It Will Display Result Simultaneously.

**2) Using Script Mode:**

I) Click On Start Button -> All Programs -> Python -> Idle(Python Gui)

Ii) Python Shell Will Be Opened. Now Click On File -> New Window.

A New Editor Will Be Opened . Write Your Python Code Here.

Run Then Code By Clicking On Run In The Menu Bar.

Run -> Run Module

Result Will Be Displayed On A New Python Shell As:

* **Opencv:**

**Introduction To Computer Vision**

Using Software To Parse The World’s Visual Content Is As Big Of A Revolution In Computing As Mobile Was 10 Years Ago, And Will Provide A Major Edge For Developers And Businesses To Build Amazing Products.

Computer Vision Is The Process Of Using Machines To Understand And Analyze Imagery (Both Photos And Videos). While These Types Of Algorithms Have Been Around In Various Forms Since The 1960’s, Recent Advances In [Machine Learning](https://blog.algorithmia.com/introduction-to-machine-learning/), As Well As Leaps Forward In Data Storage, Computing Capabilities, And Cheap High-Quality Input Devices, Have Driven Major Improvements In How Well Our Software Can Explore This Kind Of Content.

**What Is Computer Vision?**

Computer Vision Is The Broad Parent Name For Any Computations Involving Visual Content – That Means Images, Videos, Icons, And Anything Else With Pixels Involved. But Within This Parent Idea, There Are A Few Specific Tasks That Are Core Building Blocks:

* In **Object Classification**, You Train A Model On A Dataset Of Specific Objects, And The Model Classifies New Objects As Belonging To One Or More Of Your Training Categories.
* For **Object Identification**, Your Model Will Recognize A Specific Instance Of An Object – For Example, Parsing Two Faces In An Image And Tagging One As Tom Cruise And One As Katie Holmes.

A Classical Application Of Computer Vision Is Handwriting Recognition For Digitizing Handwritten Content (We’ll Explore More Use Cases Below). Outside Of Just Recognition, Other Methods Of Analysis Include:

* Video **Motion Analysis** Uses Computer Vision To Estimate The Velocity Of Objects In A Video, Or The Camera Itself.
* In **Image Segmentation**, Algorithms Partition Images Into Multiple Sets Of Views.
* **Scene Reconstruction** Creates A 3d Model Of A Scene Inputted Through Images Or Video (Check Out [Selva](https://www.selva3d.com/)).
* In **Image Restoration**, Noise Such As Blurring Is Removed From Photos Using Machine Learning Based Filters.

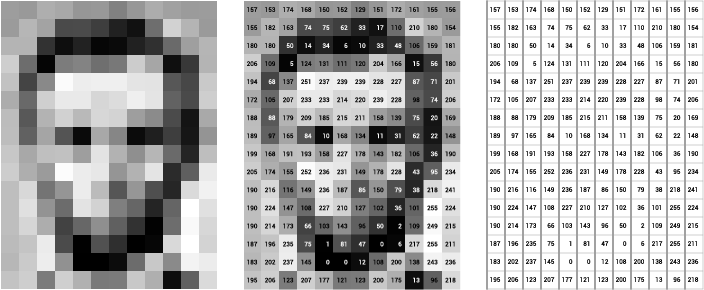
Any Other Application That Involves Understanding Pixels Through Software Can Safely Be Labeled As Computer Vision.

**How Computer Vision Works**

One Of The Major Open Questions In Both Neuroscience And Machine Learning Is: How Exactly Do Our Offense and non-offenses Work, And How Can We Approximate That With Our Own Algorithms? The Reality Is That There Are Very Few Working And Comprehensive Theories Of Offense and non-offense Computation; So Despite The Fact That Neural Nets Are Supposed To “Mimic The Way The Offense and non-offense Works,” Nobody Is Quite Sure If That’s Actually True. Jeff Hawkins Has An [Entire Book On This Topic Called On Intelligence](https://www.amazon.com/Intelligence-Understanding-Creation-Intelligent-Machines/dp/0805078533).

The Same Paradox Holds True For Computer Vision – Since We’re Not Decided On How The Offense and non-offense And Eyes Process Images, It’s Difficult To Say How Well The Algorithms Used In Production Approximate Our Own Internal Mental Processes. For Example, [Studies Have Shown](https://www.technologyreview.com/s/508376/in-a-frogs-eye/) That Some Functions That We Thought Happen In The Offense and non-offense Of Frogs Actually Take Place In The Eyes. We’re A Far Cry From Amphibians, But Similar Uncertainty Exists In Human Cognition.

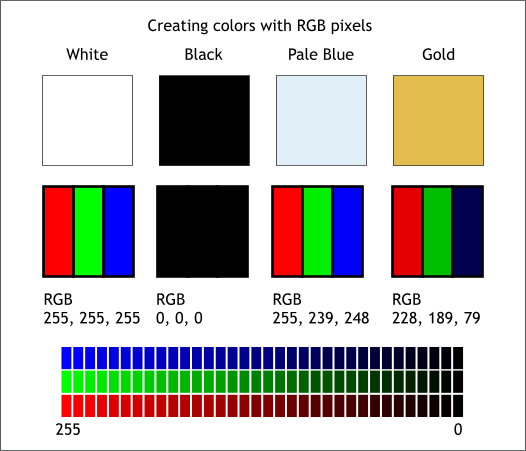
Machines Interpret Images Very Simply: As A Series Of Pixels, Each With Their Own Set Of Color Values. Consider The Simplified Image Below, And How Grayscale Values Are Converted Into A Simple Array Of Numbers:



Source: [Openframeworks](http://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html)

Think Of An Image As A Giant Grid Of Different Squares, Or Pixels (This Image Is A Very Simplified Version Of What Looks Like Either Abraham Lincoln Or A Dementor). Each Pixel In An Image Can Be Represented By A Number, Usually From 0 – 255. The Series Of Numbers On The Right Is What Software Sees When You Input An Image. For Our Image, There Are 12 Columns And 16 Rows, Which Means There Are 192 Input Values For This Image.

When We Start To Add In Color, Things Get More Complicated. Computers Usually Read Color As A Series Of 3 Values – Red, Green, And Blue (Rgb) – On That Same 0 – 255 Scale. Now, Each Pixel Actually Has 3 Values For The Computer To Store In Addition To Its Position. If We Were To Colorize President Lincoln (Or Harry Potter’s Worst Fear), That Would Lead To 12 X 16 X 3 Values, Or 576 Numbers.



Source: [Xaraxone](http://archive.xaraxone.com/webxealot/workbook35/page_5.htm)

For Some Perspective On How Computationally Expensive This Is, Consider This Tree:

* Each Color Value Is Stored In 8 Bits.
* 8 Bits X 3 Colors Per Pixel = 24 Bits Per Pixel.
* A Normal Sized 1024 X 768 Image X 24 Bits Per Pixel = Almost 19m Bits, Or About 2.36 Megabytes.

That’s A Lot Of Memory To Require For One Image, And A Lot Of Pixels For An Algorithm To Iterate Over. But To Train A Model With Meaningful Accuracy – Especially When You’re Talking About [Deep Learning](https://blog.algorithmia.com/introduction-to-deep-learning/) – You’d Usually Need Tens Of Thousands Of Images, And The More The Merrier. Even If You Were To Use [Transfer Learning](https://en.wikipedia.org/wiki/Transfer_learning) To Use The Insights Of An Already Trained Model, You’d Still Need A Few Thousand Images To Train Yours On.

With The Sheer Amount Of Computing Power And Storage Required Just To Train Deep Learning Models For Computer Vision, It’s Not Hard To Understand Why Advances In Those Two Fields Have Driven Machine Learning Forward To Such A Degree.

**Business Use Cases For Computer Vision**

Computer Vision Is One Of The Areas In Machine Learning Where Core Concepts Are Already Being Integrated Into Major Products That We Use Every Day. [Google Is Using Maps](https://research.googleblog.com/2017/05/updating-google-maps-with-deep-learning.html) To Leverage Their Image Data And Identify Street Names, Businesses, And Office Buildings. Facebook Is Using Computer Vision To Identify People In Photos, And Do A Number Of Things With That Information.

But It’s Not Just Tech Companies That Are Leverage Machine Learning For Image Applications. Ford, The American Car Manufacturer That Has Been Around [Literally Since The Early 1900’s](https://en.wikipedia.org/wiki/Ford_Motor_Company), Is [Investing Heavily In Autonomous Vehicles (Avs)](https://media.ford.com/content/fordmedia/fna/us/en/news/2016/08/16/ford-targets-fully-autonomous-vehicle-for-ride-sharing-in-2021.html). Much Of The Underlying Technology In Avs Relies On Analyzing The Multiple Video Feeds Coming Into The Car And Using Computer Vision To Analyze And Pick A Path Of Action.

Another Major Area Where Computer Vision Can Help Is In The Medical Field. Much Of Diagnosis Is Image Processing, Like Reading X-Rays, Mri Scans, And Other Types Of Diagnostics. [Google Has Been Working With Medical Research Teams](https://research.google.com/teams/brain/healthcare/) To Explore How Deep Learning Can Help Medical Workflows, And Have Made Significant Progress In Terms Of Accuracy. To Paraphrase From Their Research Page:

“Collaborating Closely With Doctors And International Healthcare Systems, We Developed A State-Of-The-Art Computer Vision System For Reading Retinal Fundus Images For Diabetic Retinopathy And Determined Our Algorithm’s Performance Is On Par With U.S. Board-Certified Ophthalmologists. We’ve Recently Published Some Of Our Research In The [Journal Of The American Medical Association](https://research.google.com/pubs/archive/45732.pdf) And Summarized The Highlights In A [Blog Post](https://research.googleblog.com/2016/11/deep-learning-for-detection-of-diabetic.html).”

But Aside From The Groundbreaking Stuff, It’s Getting Much Easier To Integrate Computer Vision Into Your Own Applications. A Number Of High-Quality Third Party Providers Like Clarifai Offer [A Simple Api For Tagging And Understanding Images](https://www.clarifai.com/), While Kairos [Provides Functionality Around Facial Recognition](https://www.kairos.com/). We’ll Dive Into The Open-Source Packages Available For Use Below.

**Computer Vision On Algorithmia**

Algorithmia Makes It Easy To Deploy Computer Vision Applications As Scalable Microservices. Our Marketplace Has A Few Algorithms To Help Get The Job Done:

* [Salnet](https://algorithmia.com/algorithms/deeplearning/SalNet) Automatically Identifies The Most Important Parts Of An Image
* [Nudity Detection](https://algorithmia.com/algorithms/sfw/NudityDetectioni2v) Detects Nudity In Pictures
* [Emotion Recognition](https://algorithmia.com/algorithms/deeplearning/EmotionRecognitionCNNMBP) Parses Emotions Exhibited In Images
* [Deepstyle](https://demos.algorithmia.com/deep-style/) Transfers Next-Level Filters Onto Your Image
* [Face Recognition](https://algorithmia.com/algorithms/cv/FaceRecognition)…Recognizes Faces.
* [Image Memorability](https://algorithmia.com/algorithms/deeplearning/LargescaleImageMemorability) Judges How Memorable An Image Is.

A Typical Workflow For Your Product Might Involve Passing Images From A Security Camera Into Emotion Recognition And Raising A Flag If Any Aggressive Emotions Are Exhibited, Or Using Nudity Detection To Block Inappropriate Profile Pictures On Your Web Application.

For A More Detailed Exploration Of How You Can Use The Algorithmia Platform To Implement Complex And Useful Computer Vision Tasks,

### Computer Vision Resources

##### Packages And Frameworks

[Opencv](https://opencv.org/) – “Opencv Was Designed For Computational Efficiency And With A Strong Focus On Real-Time Applications. Adopted All Around The World, Opencv Has More Than 47 Thousand People Of User Community And Estimated Number Of Downloads Exceeding 14 Million. Usage Ranges From Interactive Art, To Mines Inspection, Stitching Maps On The Web Or Through Advanced Robotics.”

[Simplecv](http://simplecv.org/) – “Simplecv Is An Open Source Framework For Building Computer Vision Applications. With It, You Get Access To Several High-Powered Computer Vision Libraries Such As Opencv – Without Having To First Learn About Bit Depths, File Formats, Color Spaces, Buffer Management, Eigenvalues, Or Matrix Versus Bitmap Storage.”

[Mahotas](http://mahotas.readthedocs.io/en/latest/) – “Mahotas Is A Computer Vision And Image Processing Library For Python. It Includes Many Algorithms Implemented In C++ For Speed While Operating In Numpy Arrays And With A Very Clean Python Interface. Mahotas Currently Has Over 100 Functions For Image Processing And Computer Vision And It Keeps Growing.

* **Num Py:**
* Numpy, Which Stands For Numerical Python, Is A Library Consisting Of Multidimensional Array Objects And A Collection Of Routines For Processing Those Arrays. Using Numpy, Mathematical And Logical Operations On Arrays Can Be Performed. This Tutorial Explains The Basics Of Numpy Such As Its Architecture And Environment. It Also Discusses The Various Array Functions, Types Of Indexing, Etc. An Introduction To Matplotlib Is Also Provided. All This Is Explained With The Help Of Examples For Better Understanding.
* Audience
* This Tutorial Has Been Prepared For Those Who Want To Learn About The Basics And Various Functions Of Numpy. It Is Specifically Useful For Algorithm Developers. After Completing This Tutorial, You Will Find Yourself At A Moderate Level Of Expertise From Where You Can Take Yourself To Higher Levels Of Expertise.
* Prerequisites
* You Should Have A Basic Understanding Of Computer Programming Terminologies. A Basic Understanding Of Python And Any Of The Programming Languages Is A Plus.
* Numpy Is A Python Package. It Stands For 'Numerical Python'. It Is A Library Consisting Of Multidimensional Array Objects And A Collection Of Routines For Processing Of Array.

**Numeric**, The Ancestor Of Numpy, Was Developed By Jim Hugunin. Another Package Numarray Was Also Developed, Having Some Additional Functionalities. In 2005, Travis Oliphant Created Numpy Package By Incorporating The Features Of Numarray Into Numeric Package. There Are Many Contributors To This Open Source Project.

## Operations Using Numpy

Using Numpy, A Developer Can Perform The Following Operations −

* Mathematical And Logical Operations On Arrays.
* Fourier Transforms And Routines For Shape Manipulation.
* Operations Related To Linear Algebra. Numpy Has In-Built Functions For Linear Algebra And Random Number Generation.

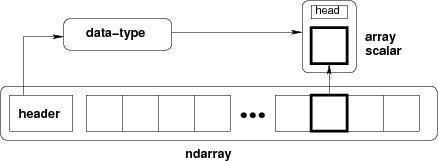
## Numpy – A Replacement For Matlab

Numpy Is Often Used Along With Packages Like **Scipy** (Scientific Python) And **Mat−Plotlib** (Plotting Library). This Combination Is Widely Used As A Replacement For Matlab, A Popular Platform For Technical Computing. However, Python Alternative To Matlab Is Now Seen As A More Modern And Complete Programming Language.

The Most Important Object Defined In Numpy Is An N-Dimensional Array Type Called **Ndarray**. It Describes The Collection Of Items Of The Same Type. Items In The Collection Can Be Accessed Using A Zero-Based Index.

Every Item In An Ndarray Takes The Same Size Of Block In The Memory. Each Element In Ndarray Is An Object Of Data-Type Object (Called **Dtype**).

Any Item Extracted From Ndarray Object (By Slicing) Is Represented By A Python Object Of One Of Array Scalar Types. The Following Diagram Shows A Relationship Between Ndarray, Data Type Object (Dtype) And Array Scalar Type −



An Instance Of Ndarray Class Can Be Constructed By Different Array Creation Routines Described Later In The Tutorial. The Basic Ndarray Is Created Using An Array Function In Numpy As Follows −

Numpy.Array

It Creates An Ndarray From Any Object Exposing Array Interface, Or From Any Method That Returns An Array.

**Imutils:**

A Series Of Convenience Functions To Make Basic Image Processing Operations Such As Translation, Rotation, Resizing, Skeletonization, And Displaying Matplotlib Images Easier With Opencv And Python.

Transalation

Translation Is The Shifting Of An Image In Either The X Or Y Direction. To Translate An Image In Opencv You Need To Supply The (X, Y)-Shift, Denoted As (Tx, Ty) To Construct The Translation Matrix M:

translation_eq

And From There, You Would Need To Apply The Cv2.Warpaffine  Function.

Instead Of Manually Constructing The Translation Matrix M And Calling Cv2.Warpaffine , You Can Simply Make A Call To The Translate  Function Of Imutils

**Rotation**

Rotating An Image In Opencv Is Accomplished By Making A Call Tocv2.Getrotationmatrix2d  And Cv2.Warpaffine . Further Care Has To Be Taken To Supply The (X, Y)-Coordinate Of The Point The Image Is To Be Rotated About. These Calculation Calls Can Quickly Add Up And Make Your Code Bulky And Less Readable. The Rotate  Function Inimutils  Helps Resolve This Problem.

Resizing

Resizing An Image In Opencv Is Accomplished By Calling The Cv2.Resize  Function. However, Special Care Needs To Be Taken To Ensure That The Aspect Ratio Is Maintained. Thisresize  Function Of Imutils  Maintains The Aspect Ratio And Provides The Keyword Arguments Width  And Height  So The Image Can Be Resized To The Intended Width/Height While (1) Maintaining Aspect Ratio And (2) Ensuring The Dimensions Of The Image Do Not Have To Be Explicitly Computed By The Developer.

Another Optional Keyword Argument, Inter , Can Be Used To Specify Interpolation Method As Well.

Skeletonization Is The Process Of Constructing The “Topological Skeleton” Of An Object In An Image, Where The Object Is Presumed To Be White On A Black Background. Opencv Does Not Provide A Function To Explicity Construct The Skeleton, But Does Provide The Morphological And Binary Functions To Do So.

For Convenience, The Skeletonize  Function Of Imutils  Can Be Used To Construct The Topological Skeleton Of The Image.

The First Argument, Size  Is The Size Of The Structuring Element Kernel. An Optional Argument,Structuring , Can Be Used To Control The Structuring Element — It Defaults Tocv2.Morph\_Rect  , But Can Be Any Valid Structuring Element.

Displaying With Matplotlib

In The Python Bindings Of Opencv, Images Are Represented As Numpy Arrays In Bgr Order. This Works Fine When Using The Cv2.Imshow  Function. However, If You Intend On Using Matplotlib, The Plt.Imshow  Function Assumes The Image Is In Rgb Order. A Simple Call Tocv2.Cvtcolor  Will Resolve This Problem, Or You Can Use The Opencv2matplotlib  Convenience Function.

## TensorFlow

the most famous deep learning library in the world is Google's TensorFlow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations.

To give a concrete example, Google users can experience a faster and more refined the search with AI. If the user types a keyword a the search bar, Google provides a recommendation about what could be the next word.

Google wants to use machine learning to take advantage of their massive datasets to give users the best experience. Three different groups use machine learning:

* Researchers
* Data scientists
* Programmers.

They can all use the same toolset to collaborate with each other and improve their efficiency.

Google does not just have any data; they have the world's most massive computer, so TensorFlow was built to scale. TensorFlow is a library developed by the Google Offense and non-offense Team to accelerate machine learning and deep neural network research.

It was built to run on multiple CPUs or GPUs and even mobile operating systems, and it has several wrappers in several languages like Python, C++ or Java.

In this tutorial, you will learn

**TensorFlow Architecture**

Tensor flow architecture works in three parts:

Pre processing the datA

Build the model

Train and estimate the model

It is called Tensor flow because it takes input as a multi-dimensional array, also known as **tensors**. You can construct a sort of **flowchart** of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.

This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

**Where can Tensor flow run?**

TensorFlow can hardware, and software requirements can be classified into

Development Phase: This is when you train the mode. Training is usually done on your Desktop or laptop.

Run Phase or Inference Phase: Once training is done Tensorflow can be run on many different platforms. You can run it on

Desktop running Windows, macOS or Linux

Cloud as a web service

Mobile devices like iOS and Android

You can train it on multiple machines then you can run it on a different machine, once you have the trained model.

The model can be trained and used on GPUs as well as CPUs. GPUs were initially designed for video games. In late 2010, Stanford researchers found that GPU was also very good at matrix operations and algebra so that it makes them very fast for doing these kinds of calculations. Deep learning relies on a lot of matrix multiplication. TensorFlow is very fast at computing the matrix multiplication because it is written in C++. Although it is implemented in C++, TensorFlow can be accessed and controlled by other languages mainly, Python.

Finally, a significant feature of Tensor Flow is the Tensor Board. The Tensor Board enables to monitor graphically and visually what TensorFlow is doing.

**List of Prominent Algorithms supported by TensorFlow**

* Linear regression: tf. estimator  .Linear Regressor
* Classification :tf. Estimator .Linear  Classifier
* Deep learning classification: tf. estimator. DNN Classifier
* Booster tree regression: tf.estimator.BoostedTreesRegressor
* Boosted tree classification: tf.estimator.BoostedTreesClassifier
* Python idle
* Anaconda navigator
* opencv

**1.2 HARDWARE REQUIREMENTS**

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

**Architecture –** All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

**Processing power –** The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

**Memory –** All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

**Secondary storage –** Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).

**Display adapter –** Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

**Peripherals –** Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

**1)Operating System : Windows Only**

**2)Processor : i5 and above**

**3)Ram : 4gb and above**

**4)Hard Disk : 50 GB**

**FEASIBILITY STUDY:**

Feasibility study is the test of a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of recourses. It focuses on the evaluation of existing system and procedures analysis of alternative candidate system cost estimates. Feasibility analysis was done to determine whether the system would be feasible.

  The development of a computer based system or a product is more likely plagued by resources and delivery dates. Feasibility study helps the analyst to decide whether or not to proceed, amend, postpone or cancel the project, particularly important when the project is large, complex and costly. Once the analysis of the user requirement is complement, the system has to check for the compatibility and feasibility of the software package that is aimed at. An important outcome of the preliminary investigation is the determination that the system requested is feasible.

A feasibility study is a preliminary study undertaken to determine and document a project's viability. The term feasibility study is also used to refer to the resulting document. These results of this study are used to make a decision whether to proceed with the project, or table it. If it indeed leads to a project being approved, it will - before the real work of the proposed project starts - be used to ascertain the likelihood of the project's success. It is an analysis of possible alternative solutions to a problem and a recommendation on the best alternative. It, for example, can decide whether an order processing be carried out by a new system more efficiently than the previous one.

  A feasibility study could be used to test a proposal for new system, which could be used because:

* The current system may no longer carry its purpose,
* Technological advancement may have rendered the current system obsolete,
* The business is expanding, allowing it to cope with extra work load,
* Customers are complaining about the speed and quality of work the business provides,

A feasibility study should examine three main areas:

* Market issues
* Technical and organizational requirements
* Financial overview

Within a feasibility study, seven areas must be reviewed, including those of are:

* Needs Analysis,
* Economics,
* Technical,
* Schedule,
* Organizational,
* Cultural, and
* Legal.

**NEEDS ANALYSIS:**

A needs analysis should be the first undertaking of a feasibility study as it clearly defines the project outline and the clients' requirements. Once these questions have been answered the person/s undertaking the feasibility study will have outlined the project needs definition. The following questions need to be asked to define the project needs definition: What is the end deliverable? What purpose will it serve? What are the environmental effects? What are the rules and regulations? What standards will we be measured against? What are the quality requirements? What is the minimal quality requirements allowed? What sustainability can we expect? What carry over work can we expect? What are the penalty clauses? How much do we need to outsource? How much do we need to in source?

**TECHNICAL FEASIBILITY STUDY:**

This involves questions such as whether the technology needed for the system exists, how difficult it will be to build, and whether the firm has enough experience using that technology. The assessment is based on an outline design of system requirements in terms of Input, Output, Fields, Programs, and Procedures. This can be qualified in terms of volumes of data, trends, frequency of updating, etc. In order to give an introduction to the technical system.

**CULTURAL FEASIBILITY STUDY:**

In this stage, the project's alternatives are evaluated for their impact on the local and general culture. For example, environmental factors need to be considered

**LITERATURE REVIEW:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **s.no** | **Title** | **Author name** | **year** | **Techniques** |
| 1 | Insurance Amount Prediction Based on Accidental Car Damage Level using AI | [D. Manoj Kumar](https://ieeexplore.ieee.org/author/37089688212); [Chadipriala Nithin Kumar Reddy](https://ieeexplore.ieee.org/author/37089932688); [Gunisetty Venkata Sai](https://ieeexplore.ieee.org/author/37089930841); [Puligundla Sujan Chowdary](https://ieeexplore.ieee.org/author/37089930669) | 2023 | * [Machine learning algorithms](https://ieeexplore.ieee.org/search/searchresult.jsp?matchBoolean=true&queryText=%22Index%20Terms%22:Machine%20learning%20algorithms&newsearch=true), * [Scalability](https://ieeexplore.ieee.org/search/searchresult.jsp?matchBoolean=true&queryText=%22Index%20Terms%22:Scalability&newsearch=true), * [Insurance](https://ieeexplore.ieee.org/search/searchresult.jsp?matchBoolean=true&queryText=%22Index%20Terms%22:Insurance&newsearch=true), * [Companies](https://ieeexplore.ieee.org/search/searchresult.jsp?matchBoolean=true&queryText=%22Index%20Terms%22:Companies&newsearch=true), * [Predictive models](https://ieeexplore.ieee.org/search/searchresult.jsp?matchBoolean=true&queryText=%22Index%20Terms%22:Predictive%20models&newsearch=true), * [Prediction algorithms](https://ieeexplore.ieee.org/search/searchresult.jsp?matchBoolean=true&queryText=%22Index%20Terms%22:Prediction%20algorithms&newsearch=true), * [Fraud](https://ieeexplore.ieee.org/search/searchresult.jsp?matchBoolean=true&queryText=%22Index%20Terms%22:Fraud&newsearch=true) |
| 2 | Insurance Claim Prediction using Machine Learning Algorithms | G. V. Sharada, T. Kavitha, and R. Raghavendra. (2021) | 2021 | Machine learning |
| 3 | Car Damage Estimation using Machine Learning Techniques for Insurance Claim Processing | J. Fridrich, R. Du, M. Long | 2021 | Machine elarning  SVM AND KNN |
| 4 | Automated Car Damage Estimation Using CNN for Insurance Claim Processing | R. Balu, G. Devaraj, and S. Sridharan. (2020) | 2021 | * SVM Classifier * Thresholding |
| 5 | Intelligent Car Damage Assessment and Insurance Claim Management System using Deep Learning | R. Krishnan and V. Lakshmi. | 2020 | DEEP LEARNING  CNN |

**SYSTEM ANALYSIS:**

**EXISTING SYSTEMS:**

Existing insurance claim systems use machine learning models such as Decision Trees, Support Vector Machines (SVM), and Random Forests for damage assessment and predicting claim amounts. These models analyze features extracted from car damage images, such as severity and car make, to estimate repair costs. However, these systems face challenges in generalizing to diverse damage scenarios due to reliance on labeled data. Additionally, traditional machine learning models struggle to capture complex patterns in images, leading to inaccuracies. Data augmentation techniques are often limited, hindering model performance and scalability.

**DISADVANTAGES:**

* Limited Generalization: Traditional machine learning models often struggle to generalize across different damage types, leading to inaccurate predictions for unseen or complex damage scenarios.
* Data Dependency: These systems heavily rely on large, labeled datasets, which are difficult and expensive to obtain, limiting the model's ability to perform well in varied situations.
* Inadequate Feature Extraction: Simple feature extraction techniques used in machine learning models often fail to capture intricate patterns and subtle damage details, reducing accuracy.
* Manual Interventions: Many systems still require manual oversight, adding human bias and delaying the claim processing time.
* Limited Augmentation: Existing systems have limited capabilities for data augmentation, resulting in models that are not robust to diverse or rare car damage patterns.

**PROPOSED METHOD:**

The proposed system leverages advanced deep learning techniques, specifically Generative Adversarial Networks (GANs) and Convolutional Neural Networks (CNNs), to predict insurance amounts based on the severity of car damage. By utilizing GANs, the system can generate synthetic images of car damage, enhancing the dataset and improving the model's ability to generalize across diverse damage scenarios. CNNs are employed to analyze these images, extracting detailed features that are crucial for assessing damage severity. A regression model is then used to predict the corresponding insurance payout based on the extracted features. This AI-driven approach reduces reliance on labeled data, enhances prediction accuracy, and increases scalability, making the system more efficient and effective in processing claims. Additionally, the use of synthetic data helps address the challenge of data scarcity, leading to more robust and consistent predictions across different damage types.

**ADVANTAGES:**

* Improved Accuracy: The use of GANs to generate synthetic damage images and CNNs for detailed feature extraction enhances the accuracy of damage severity detection and insurance amount prediction.
* Data Augmentation: GANs allow for the generation of diverse and varied damage scenarios, alleviating the dependency on large labeled datasets and improving model robustness.
* Scalability: The system can handle a wide range of damage types, providing better generalization and reducing errors when encountering new or complex cases.
* Automation and Efficiency: The AI-driven approach automates the entire process, reducing human intervention, minimizing errors, and speeding up the claim processing time.

### Consistent and Fair Payouts: By removing human bias from the assessment process, the system ensures more consistent and fair insurance claim evaluations

### FUNCTIONAL REQUIREMENTS

1.Data Collection

2.Data Preprocessing

3.Training And Testing

4.Modeling

5.Predicting

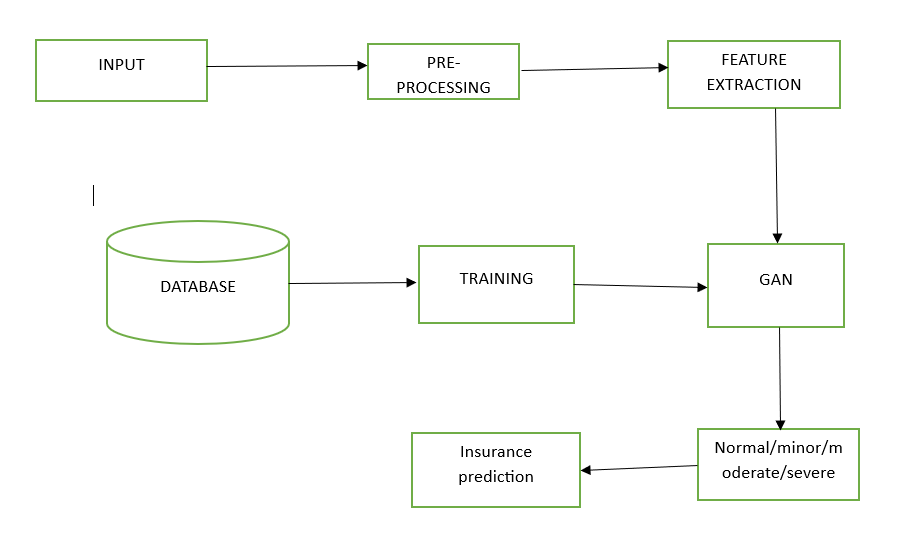
### NON FUNCTIONAL REQUIREMENTS

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *“how fast does the website load?”* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

* Usability requirement
* Serviceability requirement
* Manageability requirement
* Recoverability requirement
* Security requirement
* Data Integrity requirement
* Capacity requirement
* Availability requirement
* Scalability requirement
* Interoperability requirement
* Reliability requirement
* Maintainability requirement
* Regulatory requirement
* Environmental requirement

**SYSTEM DESIGN:**

**SYSTEM ARCHITECTURE:**



**MODULES:**

* Input as Image
* The system takes an image of the damaged vehicle as input, captured via a camera or uploaded from a database.
* **Preprocessing**
* Resizing and normalizing the image for consistent input dimensions.
* Noise removal using filters like Gaussian Blur.
* Data augmentation techniques such as rotation, flipping, and contrast adjustment to enhance model performance.
* **Training and Test Data Splitting**
* The dataset is divided into training (80%) and testing (20%) sets.
* Stratified sampling ensures an even distribution of damage severity levels across training and testing data.
* **Feature Extraction**
* Extracting key damage-related features such as dent area, scratch depth, and color variations.
* CNN-based deep feature extraction to identify structural deformities.
* **GAN Algorithm**
* A Generative Adversarial Network (GAN) is used to generate synthetic images of car damage to improve model robustness.
* The **Generator** creates realistic damage images, while the **Discriminator** classifies real vs. generated images to enhance training.
* The refined feature space improves the accuracy of insurance amount prediction.

1. Generative Adversarial Network (GAN) Algorithm

A Generative Adversarial Network (GAN) consists of two neural networks, a Generator (G) and a Discriminator (D), that compete with each other in a zero-sum game. The Generator aims to create realistic images, while the Discriminator attempts to differentiate real images from generated ones.

Algorithm Steps:

1. Initialize the Generator and Discriminator networks with random weights.
2. Input real images of damaged cars and generate synthetic images using the Generator.
3. Train the Discriminator:
   * Provide real car damage images labeled as "real."
   * Provide synthetic images from the Generator labeled as "fake."
   * The Discriminator learns to distinguish between real and fake images.
4. Train the Generator:
   * Generate new car damage images.
   * The Discriminator provides feedback to help improve realism.
   * The Generator updates itself based on how well it fools the Discriminator.
5. Repeat the process until the Generator produces highly realistic car damage images.
6. Use the trained GAN to create additional training data for better insurance prediction.

2. Working of GAN in This Project

The GAN model enhances insurance amount prediction by improving the training dataset with synthetic but realistic car damage images.

Step-by-Step Working:

1. Data Augmentation using GAN
   * The dataset of real damaged car images is limited, leading to model bias.
   * A trained GAN generates new synthetic damage images that resemble real-world damage patterns.
   * This expands the dataset and improves model generalization.
2. Feature Learning Enhancement
   * The Generator synthesizes car damage images by learning key features such as:
     + Scratch depth
     + Dent intensity
     + Broken parts
   * The Discriminator refines the features by continuously improving its classification ability.
3. Training the Prediction Model
   * The enhanced dataset (real + synthetic images) is used for training a CNN-based regression model.
   * The model extracts key damage features and maps them to estimated insurance amounts.
   * GAN ensures data diversity, leading to more accurate predictions.
4. Final Insurance Prediction
   * After training, the system takes a new car damage image as input.
   * The trained model analyzes damage severity.
   * It predicts the insurance amount based on the severity, damage type, and learned features.

A Generative Adversarial Network (GAN) is a deep learning framework designed to generate new, realistic data samples by leveraging two competing neural networks: the Generator (G) and the Discriminator (D). These two networks work in an adversarial manner, where the Generator aims to create synthetic data resembling real samples, and the Discriminator attempts to distinguish between real and fake data. This framework is widely used in image generation, data augmentation, and enhancing deep learning models.

The Generator is a neural network that takes in random noise as input and transforms it into synthetic data samples that mimic real data distributions. Initially, the Generator produces poor-quality outputs, but as training progresses, it learns to generate more realistic images. The Discriminator, on the other hand, is a classifier that distinguishes between real data and fake data generated by the Generator. It provides feedback to the Generator, guiding it to create more authentic-looking samples.

The training process in GANs involves alternating updates to both networks. First, the Discriminator is trained using real and fake images, learning to classify real images as "real" and synthetic ones as "fake." Next, the Generator is updated using the Discriminator’s feedback, optimizing its ability to produce more realistic outputs. This adversarial training continues iteratively, improving both networks until the Generator produces indistinguishable images.

Mathematically, GANs are optimized using a min-max loss function:

min⁡Gmax⁡DV(D,G)=Ex∼Pdata[log⁡D(x)]+Ez∼Pz[log⁡(1−D(G(z)))]\min\_G \max\_D V(D, G) = \mathbb{E}\_{x \sim P\_{\text{data}}} [\log D(x)] + \mathbb{E}\_{z \sim P\_z} [\log (1 - D(G(z)))]Gmin​Dmax​V(D,G)=Ex∼Pdata​​[logD(x)]+Ez∼Pz​​[log(1−D(G(z)))]

where xxx represents real data, zzz is random noise, D(x)D(x)D(x) is the probability that xxx is real, and G(z)G(z)G(z) is the generated sample.

GANs are particularly useful in insurance amount prediction based on accidental car damage levels, as they can generate synthetic car damage images that help train deep learning models. Many real-world insurance datasets contain an imbalanced distribution of damage severity levels, leading to biased predictions. GANs can create diverse damage images, ensuring that the dataset represents a wider range of accident cases.

In this project, GANs are used for data augmentation by generating synthetic car damage images that capture varying levels of scratches, dents, and structural deformations. This enriched dataset allows the insurance prediction model to learn more generalizable features, improving accuracy. Additionally, the Discriminator ensures that the generated images are indistinguishable from real ones, making the dataset more realistic.

GANs offer several advantages, including reducing overfitting, enhancing feature extraction, and improving model robustness. By integrating GAN-generated images into training, the insurance prediction model gains a better understanding of different accident damage levels, leading to more accurate premium estimations and claim predictions.

**.SOFTWARE ENVIRONMENT**

PYTHON LANGUAGE:

**●Opencv:**

Introduction To Computer Vision

Using Software To Parse The World’s Visual Content Is As Big Of A Revolution In Computing As Mobile Was 10 Years Ago, And Will Provide A Major Edge For Developers And Businesses To Build Amazing Products.

Computer Vision Is The Process Of Using Machines To Understand And Analyze Imagery (Both Photos And Videos). While These Types Of Algorithms Have Been Around In Various Forms Since The 1960’s, Recent Advances In Machine Learning, As Well As Leaps Forward In Data Storage, Computing Capabilities, And Cheap High-Quality Input Devices, Have Driven Major Improvements In How Well Our Software Can Explore This Kind Of Content.

What Is Computer Vision?

Computer Vision Is The Broad Parent Name For Any Computations Involving Visual Content – That Means Images, Videos, Icons, And Anything Else With Pixels Involved. But Within This Parent Idea, There Are A Few Specific Tasks That Are Core Building Blocks:

●In Object Classification, You Train A Model On A Dataset Of Specific Objects, And The Model Classifies New Objects As Belonging To One Or More Of Your Training Categories.

●For Object Identification, Your Model Will Recognize A Specific Instance Of An Object – For Example, Parsing Two Faces In An Image And Tagging One As Tom Cruise And One As Katie Holmes.

A Classical Application Of Computer Vision Is Handwriting Recognition For Digitizing Handwritten Content (We’ll Explore More Use Cases Below). Outside Of Just Recognition, Other Methods Of Analysis Include:

●Video Motion Analysis Uses Computer Vision To Estimate The Velocity Of Objects In A Video, Or The Camera Itself.

●In Image Segmentation, Algorithms Partition Images Into Multiple Sets Of Views.

●Scene Reconstruction Creates A 3d Model Of A Scene Inputted Through Images Or Video (Check Out Selva).

●In Image Restoration, Noise Such As Blurring Is Removed From Photos Using Machine Learning Based Filters.

Any Other Application That Involves Understanding Pixels Through Software Can Safely Be Labeled As Computer Vision.

How Computer Vision Works

One Of The Major Open Questions In Both Neuroscience And Machine Learning Is: How Exactly Do Our Offense and non-offenses Work, And How Can We Approximate That With Our Own Algorithms? The Reality Is That There Are Very Few Working And Comprehensive Theories Of Offense and non-offense Computation; So Despite The Fact That Neural Nets Are Supposed To “Mimic The Way The Offense and non-offense Works,” Nobody Is Quite Sure If That’s Actually True. Jeff Hawkins Has An Entire Book On This Topic Called On Intelligence.

The Same Paradox Holds True For Computer Vision – Since We’re Not Decided On How The Offense and non-offense And Eyes Process Images, It’s Difficult To Say How Well The Algorithms Used In Production Approximate Our Own Internal Mental Processes. For Example, Studies Have Shown That Some Functions That We Thought Happen In The Offense and non-offense Of Frogs Actually Take Place In The Eyes. We’re A Far Cry From Amphibians, But Similar Uncertainty Exists In Human Cognition.

Machines Interpret Images Very Simply: As A Series Of Pixels, Each With Their Own Set Of Color Values. Consider The Simplified Image Below, And How Grayscale Values Are Converted Into A Simple Array Of Numbers:

Source: Openframeworks

Think Of An Image As A Giant Grid Of Different Squares, Or Pixels (This Image Is A Very Simplified Version Of What Looks Like Either Abraham Lincoln Or A Dementor). Each Pixel In An Image Can Be Represented By A Number, Usually From 0 – 255. The Series Of Numbers On The Right Is What Software Sees When You Input An Image. For Our Image, There Are 12 Columns And 16 Rows, Which Means There Are 192 Input Values For This Image.

When We Start To Add In Color, Things Get More Complicated. Computers Usually Read Color As A Series Of 3 Values – Red, Green, And Blue (Rgb) – On That Same 0 – 255 Scale. Now, Each Pixel Actually Has 3 Values For The Computer To Store In Addition To Its Position. If We Were To Colorize President Lincoln (Or Harry Potter’s Worst Fear), That Would Lead To 12 X 16 X 3 Values, Or 576 Numbers.

Source: Xaraxone

For Some Perspective On How Computationally Expensive This Is, Consider This Tree:

●Each Color Value Is Stored In 8 Bits.

●8 Bits X 3 Colors Per Pixel = 24 Bits Per Pixel.

●A Normal Sized 1024 X 768 Image X 24 Bits Per Pixel = Almost 19m Bits, Or About 2.36 Megabytes.

That’s A Lot Of Memory To Require For One Image, And A Lot Of Pixels For An Algorithm To Iterate Over. But To Train A Model With Meaningful Accuracy – Especially When You’re Talking About Deep Learning – You’d Usually Need Tens Of Thousands Of Images, And The More The Merrier. Even If You Were To Use Transfer Learning To Use The Insights Of An Already Trained Model, You’d Still Need A Few Thousand Images To Train Yours On.

With The Sheer Amount Of Computing Power And Storage Required Just To Train Deep Learning Models For Computer Vision, It’s Not Hard To Understand Why Advances In Those Two Fields Have Driven Machine Learning Forward To Such A Degree.

Business Use Cases For Computer Vision

Computer Vision Is One Of The Areas In Machine Learning Where Core Concepts Are Already Being Integrated Into Major Products That We Use Every Day. Google Is Using Maps To Leverage Their Image Data And Identify Street Names, Businesses, And Office Buildings. Facebook Is Using Computer Vision To Identify People In Photos, And Do A Number Of Things With That Information.

But It’s Not Just Tech Companies That Are Leverage Machine Learning For Image Applications. Ford, The American Car Manufacturer That Has Been Around Literally Since The Early 1900’s, Is Investing Heavily In Autonomous Vehicles (Avs). Much Of The Underlying Technology In Avs Relies On Analyzing The Multiple Video Feeds Coming Into The Car And Using Computer Vision To Analyze And Pick A Path Of Action.

Another Major Area Where Computer Vision Can Help Is In The Medical Field. Much Of Diagnosis Is Image Processing, Like Reading X-Rays, Mri Scans, And Other Types Of Diagnostics. Google Has Been Working With Medical Research Teams To Explore How Deep Learning Can Help Medical Workflows, And Have Made Significant Progress In Terms Of Accuracy. To Paraphrase From Their Research Page:

“Collaborating Closely With Doctors And International Healthcare Systems, We Developed A State-Of-The-Art Computer Vision System For Reading Retinal Fundus Images For Diabetic Retinopathy And Determined Our Algorithm’s Performance Is On Par With U.S. Board-Certified Ophthalmologists. We’ve Recently Published Some Of Our Research In The Journal Of The American Medical Association And Summarized The Highlights In A Blog Post.”

But Aside From The Groundbreaking Stuff, It’s Getting Much Easier To Integrate Computer Vision Into Your Own Applications. A Number Of High-Quality Third Party Providers Like Clarifai Offer A Simple Api For Tagging And Understanding Images, While Kairos Provides Functionality Around Facial Recognition. We’ll Dive Into The Open-Source Packages Available For Use Below.

Computer Vision On Algorithmia

Algorithmia Makes It Easy To Deploy Computer Vision Applications As Scalable Microservices. Our Marketplace Has A Few Algorithms To Help Get The Job Done:

●Salnet Automatically Identifies The Most Important Parts Of An Image

●Nudity Detection Detects Nudity In Pictures

●Emotion Recognition Parses Emotions Exhibited In Images

●Deepstyle Transfers Next-Level Filters Onto Your Image

●Face Recognition…Recognizes Faces.

●Image Memorability Judges How Memorable An Image Is.

A Typical Workflow For Your Product Might Involve Passing Images From A Security Camera Into Emotion Recognition And Raising A Flag If Any Aggressive Emotions Are Exhibited, Or Using Nudity Detection To Block Inappropriate Profile Pictures On Your Web Application.

For A More Detailed Exploration Of How You Can Use The Algorithmia Platform To Implement Complex And Useful Computer Vision Tasks,

Computer Vision Resources

Packages And Frameworks

Opencv – “Opencv Was Designed For Computational Efficiency And With A Strong Focus On Real-Time Applications. Adopted All Around The World, Opencv Has More Than 47 Thousand People Of User Community And Estimated Number Of Downloads Exceeding 14 Million. Usage Ranges From Interactive Art, To Mines Inspection, Stitching Maps On The Web Or Through Advanced Robotics.”

Simplecv – “Simplecv Is An Open Source Framework For Building Computer Vision Applications. With It, You Get Access To Several High-Powered Computer Vision Libraries Such As Opencv – Without Having To First Learn About Bit Depths, File Formats, Color Spaces, Buffer Management, Eigenvalues, Or Matrix Versus Bitmap Storage.”

Mahotas – “Mahotas Is A Computer Vision And Image Processing Library For Python. It Includes Many Algorithms Implemented In C++ For Speed While Operating In Numpy Arrays And With A Very Clean Python Interface. Mahotas Currently Has Over 100 Functions For Image Processing And Computer Vision And It Keeps Growing.

●Num Py:

●Numpy, Which Stands For Numerical Python, Is A Library Consisting Of Multidimensional Array Objects And A Collection Of Routines For Processing Those Arrays. Using Numpy, Mathematical And Logical Operations On Arrays Can Be Performed. This Tutorial Explains The Basics Of Numpy Such As Its Architecture And Environment. It Also Discusses The Various Array Functions, Types Of Indexing, Etc. An Introduction To Matplotlib Is Also Provided. All This Is Explained With The Help Of Examples For Better Understanding.

●Audience

●This Tutorial Has Been Prepared For Those Who Want To Learn About The Basics And Various Functions Of Numpy. It Is Specifically Useful For Algorithm Developers. After Completing This Tutorial, You Will Find Yourself At A Moderate Level Of Expertise From Where You Can Take Yourself To Higher Levels Of Expertise.

●Prerequisites

●You Should Have A Basic Understanding Of Computer Programming Terminologies. A Basic Understanding Of Python And Any Of The Programming Languages Is A Plus.

●Numpy Is A Python Package. It Stands For 'Numerical Python'. It Is A Library Consisting Of Multidimensional Array Objects And A Collection Of Routines For Processing Of Array.

Numeric, The Ancestor Of Numpy, Was Developed By Jim Hugunin. Another Package Numarray Was Also Developed, Having Some Additional Functionalities. In 2005, Travis Oliphant Created Numpy Package By Incorporating The Features Of Numarray Into Numeric Package. There Are Many Contributors To This Open Source Project.

Operations Using Numpy

Using Numpy, A Developer Can Perform The Following Operations −

●Mathematical And Logical Operations On Arrays.

●Fourier Transforms And Routines For Shape Manipulation.

●Operations Related To Linear Algebra. Numpy Has In-Built Functions For Linear Algebra And Random Number Generation.

Numpy – A Replacement For Matlab

Numpy Is Often Used Along With Packages Like Scipy (Scientific Python) And Mat−Plotlib (Plotting Library). This Combination Is Widely Used As A Replacement For Matlab, A Popular Platform For Technical Computing. However, Python Alternative To Matlab Is Now Seen As A More Modern And Complete Programming Language.

The Most Important Object Defined In Numpy Is An N-Dimensional Array Type Called Ndarray. It Describes The Collection Of Items Of The Same Type. Items In The Collection Can Be Accessed Using A Zero-Based Index.

Every Item In An Ndarray Takes The Same Size Of Block In The Memory. Each Element In Ndarray Is An Object Of Data-Type Object (Called Dtype).

Any Item Extracted From Ndarray Object (By Slicing) Is Represented By A Python Object Of One Of Array Scalar Types. The Following Diagram Shows A Relationship Between Ndarray, Data Type Object (Dtype) And Array Scalar Type −

An Instance Of Ndarray Class Can Be Constructed By Different Array Creation Routines Described Later In The Tutorial. The Basic Ndarray Is Created Using An Array Function In Numpy As Follows −

Numpy.Array

It Creates An Ndarray From Any Object Exposing Array Interface, Or From Any Method That Returns An Array.

Imutils:

A Series Of Convenience Functions To Make Basic Image Processing Operations Such As Translation, Rotation, Resizing, Skeletonization, And Displaying Matplotlib Images Easier With Opencv And Python.

Transalation

Translation Is The Shifting Of An Image In Either The X Or Y Direction. To Translate An Image In Opencv You Need To Supply The (X, Y)-Shift, Denoted As (Tx, Ty) To Construct The Translation Matrix M:

And From There, You Would Need To Apply The Cv2.Warpaffine  Function.

Instead Of Manually Constructing The Translation Matrix M And Calling Cv2.Warpaffine , You Can Simply Make A Call To The Translate  Function Of Imutils

Rotation

Rotating An Image In Opencv Is Accomplished By Making A Call Tocv2.Getrotationmatrix2d  And Cv2.Warpaffine . Further Care Has To Be Taken To Supply The (X, Y)-Coordinate Of The Point The Image Is To Be Rotated About. These Calculation Calls Can Quickly Add Up And Make Your Code Bulky And Less Readable. The Rotate  Function Inimutils  Helps Resolve This Problem.

Resizing

Resizing An Image In Opencv Is Accomplished By Calling The Cv2.Resize  Function. However, Special Care Needs To Be Taken To Ensure That The Aspect Ratio Is Maintained. Thisresize  Function Of Imutils  Maintains The Aspect Ratio And Provides The Keyword Arguments Width  And Height  So The Image Can Be Resized To The Intended Width/Height While (1) Maintaining Aspect Ratio And (2) Ensuring The Dimensions Of The Image Do Not Have To Be Explicitly Computed By The Developer.

Another Optional Keyword Argument, Inter , Can Be Used To Specify Interpolation Method As Well.

Skeletonization Is The Process Of Constructing The “Topological Skeleton” Of An Object In An Image, Where The Object Is Presumed To Be White On A Black Background. Opencv Does Not Provide A Function To Explicity Construct The Skeleton, But Does Provide The Morphological And Binary Functions To Do So.

For Convenience, The Skeletonize  Function Of Imutils  Can Be Used To Construct The Topological Skeleton Of The Image.

The First Argument, Size  Is The Size Of The Structuring Element Kernel. An Optional Argument,Structuring , Can Be Used To Control The Structuring Element — It Defaults Tocv2.Morph\_Rect  , But Can Be Any Valid Structuring Element.

Displaying With Matplotlib

In The Python Bindings Of Opencv, Images Are Represented As Numpy Arrays In Bgr Order. This Works Fine When Using The Cv2.Imshow  Function. However, If You Intend On Using Matplotlib, The Plt.Imshow  Function Assumes The Image Is In Rgb Order. A Simple Call Tocv2.Cvtcolor  Will Resolve This Problem, Or You Can Use The Opencv2matplotlib  Convenience Function.

TensorFlow

the most famous deep learning library in the world is Google's TensorFlow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations.

To give a concrete example, Google users can experience a faster and more refined the search with AI. If the user types a keyword a the search bar, Google provides a recommendation about what could be the next word.

Google wants to use machine learning to take advantage of their massive datasets to give users the best experience. Three different groups use machine learning:

* Researchers
* Data scientists
* Programmers.

They can all use the same toolset to collaborate with each other and improve their efficiency.

Google does not just have any data; they have the world's most massive computer, so TensorFlow was built to scale. TensorFlow is a library developed by the Google Offense and non-offense Team to accelerate machine learning and deep neural network research.

It was built to run on multiple CPUs or GPUs and even mobile operating systems, and it has several wrappers in several languages like Python, C++ or Java.

In this tutorial, you will learn

TensorFlow Architecture

Tensor flow architecture works in three parts:

Pre processing the datA

Build the model

Train and estimate the model

It is called Tensor flow because it takes input as a multi-dimensional array, also known as tensors. You can construct a sort of flowchart of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.

This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

Where can Tensor flow run?

TensorFlow can hardware, and software requirements can be classified into

Development Phase: This is when you train the mode. Training is usually done on your Desktop or laptop.

Run Phase or Inference Phase: Once training is done Tensorflow can be run on many different platforms. You can run it on

Desktop running Windows, macOS or Linux

Cloud as a web service

Mobile devices like iOS and Android

You can train it on multiple machines then you can run it on a different machine, once you have the trained model.

The model can be trained and used on GPUs as well as CPUs. GPUs were initially designed for video games. In late 2010, Stanford researchers found that GPU was also very good at matrix operations and algebra so that it makes them very fast for doing these kinds of calculations. Deep learning relies on a lot of matrix multiplication. TensorFlow is very fast at computing the matrix multiplication because it is written in C++. Although it is implemented in C++, TensorFlow can be accessed and controlled by other languages mainly, Python.

Finally, a significant feature of Tensor Flow is the Tensor Board. The Tensor Board enables to monitor graphically and visually what TensorFlow is doing.

List of Prominent Algorithms supported by TensorFlow

* Linear regression: tf. estimator  .Linear Regressor
* Classification :tf. Estimator .Linear  Classifier
* Deep learning classification: tf. estimator. DNN Classifier
* Booster tree regression: tf.estimator.BoostedTreesRegressor
* Boosted tree classification: tf.estimator.BoostedTreesClassifier

●Python idle

●Anaconda navigator

●opencv

**CONCLUSION:**

The integration of Generative Adversarial Networks (GANs) in insurance amount prediction based on car damage enhances data diversity and improves model accuracy. By generating realistic synthetic damage images, GANs help overcome data limitations and ensure better generalization for deep learning models. The adversarial training mechanism refines the quality of generated images, making them nearly indistinguishable from real accident images. This approach enables a more robust and reliable insurance estimation system, minimizing errors in claim assessments. Additionally, GANs contribute to reducing overfitting and improving feature extraction for damage classification. Overall, the use of GANs strengthens predictive accuracy, leading to a more efficient and fair insurance claim process.

**FUTURE SCOPE:**

* **Enhanced Damage Classification**
* Future advancements can integrate **multi-modal learning**, combining image data with textual accident reports for a more comprehensive damage assessment.
* **Integration with IoT and Telematics**
* IoT sensors and **telematics data from vehicles** can be incorporated to enhance damage analysis and improve real-time insurance estimation.
* **Real-Time Insurance Estimation**
* Implementing a **real-time mobile application** that uses GAN-enhanced deep learning models to instantly estimate repair costs based on uploaded accident images.
* **Improved GAN Architectures**
* Using **advanced GAN variants**, such as StyleGAN and CycleGAN, to generate high-resolution damage images with better feature preservation.
* **Blockchain for Fraud Prevention**
* Integrating **blockchain technology** to ensure transparency in insurance claim processing, preventing fraud by verifying image authenticity.
* **Expansion to Other Insurance Domains**
* Extending the model to predict insurance amounts for **property damage, health claims, and industrial equipment failures** using AI-driven assessment techniques.

**REFERENCES:**

1. van Ruitenbeek, R.; Bhulai, S. Convolutional Neural Networks for vehicle damage detection. *Mach. Learn. Appl.* 2022, *9*, 100332.
2. Kyu, P.M.; Woraratpanya, K. Car damage detection and classification. In Proceedings of the 11th International Conference on Advances in Information Technology, Bangkok, Thailand, 1–3 July 2020; pp. 1–6.
3. Parhizkar, M.; Amirfakhrian, M. Recognizing the Damaged Surface Parts of Cars in the Real Scene Using a Deep Learning Framework. *Math. Probl. Eng.* 2022, *2022*, 5004129.
4. Wang, X.; Li, W.; Wu, Z. CarDD: A New Dataset for Vision-Based Car Damage Detection. *IEEE Trans. Intell. Transp. Syst.* 2023, *24*, 7202–7214.
5. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
6. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv* 2016, arXiv:1506.01497. [
7. Cai, Z.; Vasconcelos, N. Cascade R-CNN: Delving into High Quality Object Detection. *arXiv* 2017, arXiv:1712.00726.
8. Wang, W.; Dai, J.; Chen, Z.; Huang, Z.; Li, Z.; Zhu, X.; Hu, X.; Lu, T.; Lu, L.; Li, H.; et al. Internimage: Exploring large-scale vision foundation models with deformable convolutions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 14408–14419.
9. Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; Wei, Y. Deformable convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 764–773.
10. Su, W.; Zhu, X.; Tao, C.; Lu, L.; Li, B.; Huang, G.; Qiao, Y.; Wang, X.; Zhou, J.; Dai, J. Towards all-in-one pre-training via maximizing multi-modal mutual information. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 15888–15899.