CHAPTER 1
INTRODUCTION

1.1 Problem statement :

The method of image processing is used to do some processes on a picture like an image enhancement or to remove some functional data from the image. are in lanes that have less frequency of traffic.
Cartooning of an image is an interesting project under image processing where it takes an input image, processes it and produces an output as a cartoon.

1.2 Motivation:

Cartoons are humorous, satirical, and at times opinionated. Drawing cartoons is however, not easy. Only those well-trained artists who possess this great skill can do it well. Recently, many technologies have been developed to make it possible to create cartoons entirely on the computer. This can be recreating and helps one to have a cartoonic view of everything..

1.3 Objective:

To develop an application to cartoonize humans, other objects and images Make it possible to blend as many cartoons as possible Save the cartoons in the application if needed.

1.3.1 Proposed System:

Downscale the image and then apply bilateral filter to get a cartoon flavour. Then again we upscale the image.
Bilateral Filter: This filter is the key element in the colour image processing chain, as it homogenizes colour regions while preserving edges, even over multiple iterations.
Getting a blurred version of the original image. For this, we first convert the image to gray – scale and then we apply the media blur filter.
Next step is to identify the edges in the image using Edge Detection.
Edge Detection- Process of identifying edges in an image to be used as a fundamental asset in image analysis and locating areas with strong intensity contrasts.

1.3.2 Advantages of proposed system :

Android Studio 3.0 and above

Open CV

Python

Camera to capture an image

CHAPTER 2
TECHNOLOGIES LEARNT

What is Python :-

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python :-

Let's see how Python dominates over other languages.

***1.* Extensive Libraries**

Python downloads with an extensive library and it *contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more.* So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print **'Hello World'**. But in Python, just a print statement will do. It is also quite **easy to learn**, **understand**, and **code**. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory**. This further aids the readability of the code.

8. Object-Oriented

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

9. Free and Open-Source

Like we said earlier, Python is **freely available**. But not only can you **download Python** for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code**

only once, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.
Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.
The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don't need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open DataBase Connectivity)**, Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python : -

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &

Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it." Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

What is Machine Learning : -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories Of Machine Leaning :-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machines Learning :-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

Applications of Machines Learning :-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention

- Recommendation of products to customer in online shopping

How to Start Learning Machine Learning?

Arthur Samuel coined the term **“Machine Learning”** in 1959 and defined it as a **“Field of study that gives computers the capability to learn without being explicitly programmed”.**
And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a *344%* growth and an average base salary of **$146,085** per year.
But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let’s get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don’t know these, never fear! You don’t need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!!
Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.
So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as **Fork Python** available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

- **Model –** A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature –** A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.

- **Target (Label)** – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training** – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction** – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

- **Supervised Learning** – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning** – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning** – This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning** – This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning :-

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized.

ML is also good at recognizing spam.

3. Continuous Improvement

As **ML algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning :-

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Python Development Steps : -

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it." Some changes in Python 7.3:

- Print is now a function
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e. int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead Of Unicode Vs. 8-bit

Purpose :-

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules Used in Project :-

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

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

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. **Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac :

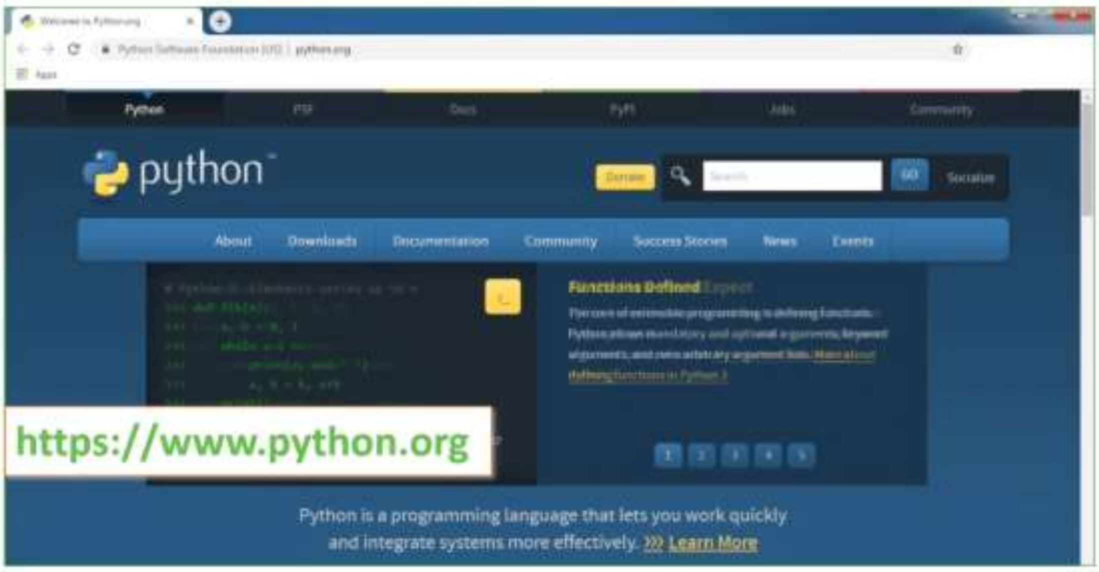
There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. **Download the Python Cheatsheet here.**The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

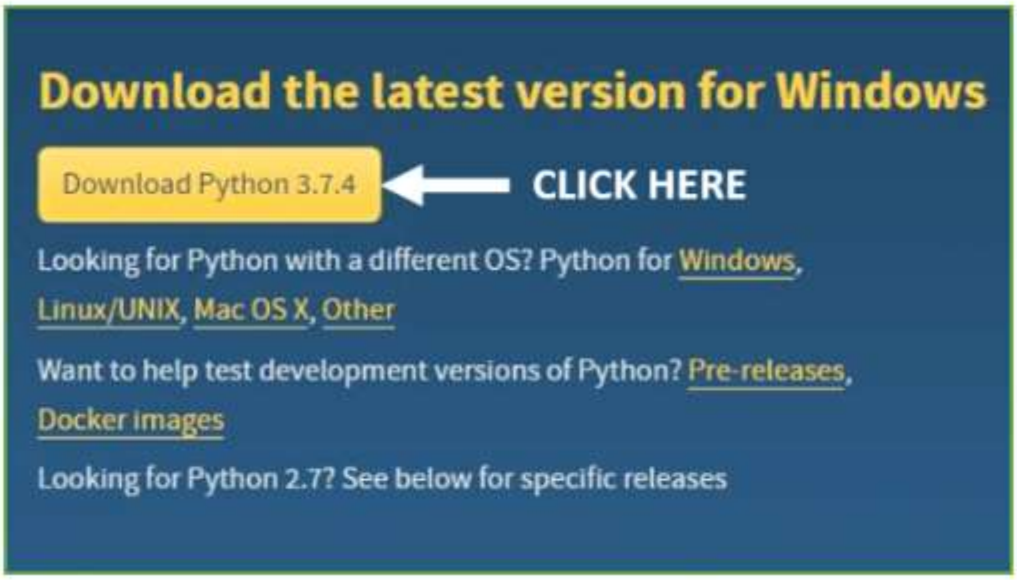
Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: **https://www.python.org**



Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.



Download the latest version for Windows

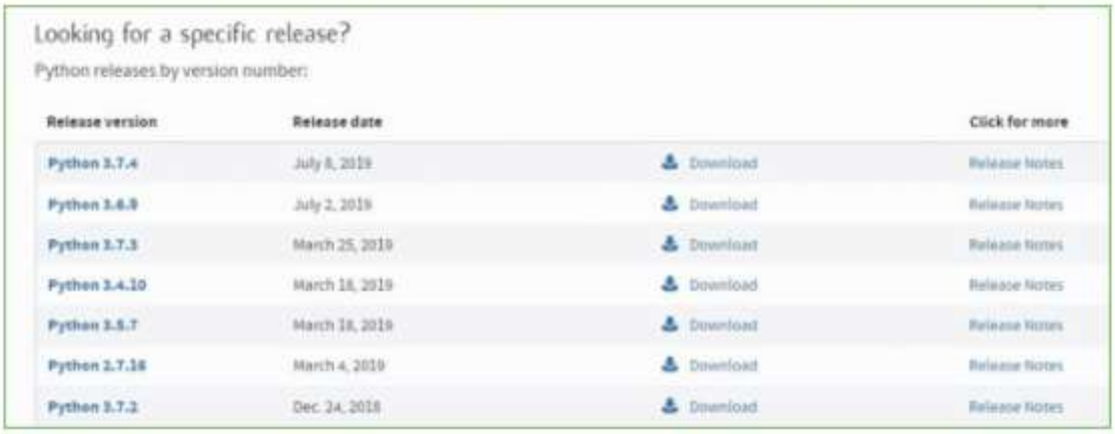
Download Python 3.7.4 ← CLICK HERE

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases, Docker images

Looking for Python 2.7? See below for specific releases

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



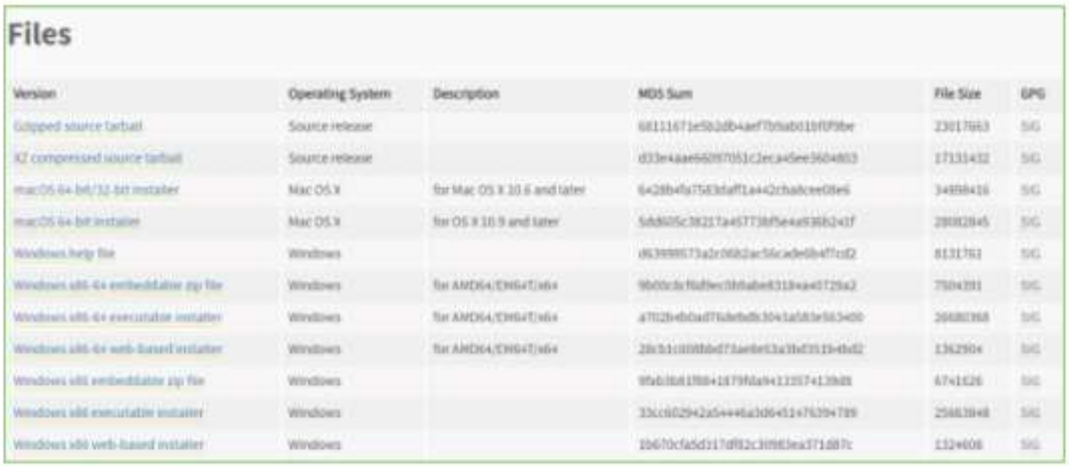
Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab01bf0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cba8cee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5e4a936b241f	28082845	SIG
Windows help file	Windows		d63999573a2c06b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c8cf6d9ec0b9abe83184a40729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76debdb3043a583e563400	26680368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c608bbd73ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3b81f8841879fda94133574139d8	6741626	SIG
Windows x86 executable installer	Windows		33cc602942a54446a3d6451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371d87c	1324608	SIG

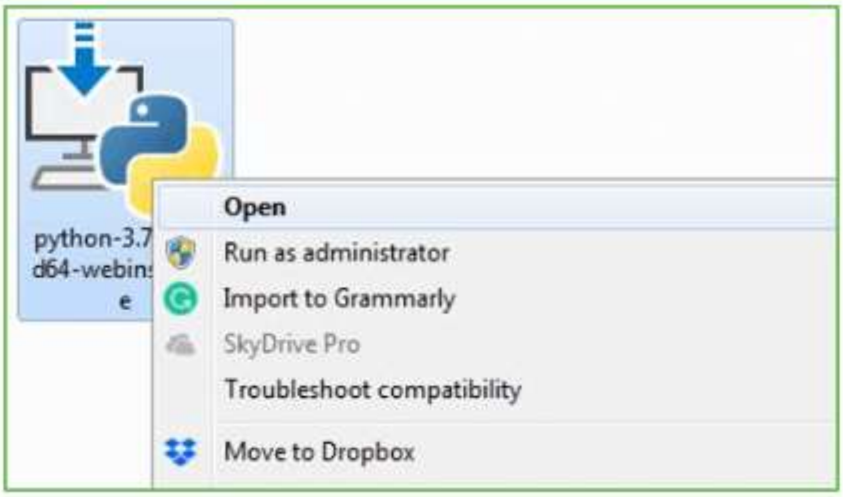
- To download **Windows 32-bit python**, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download **Windows 64-bit python**, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

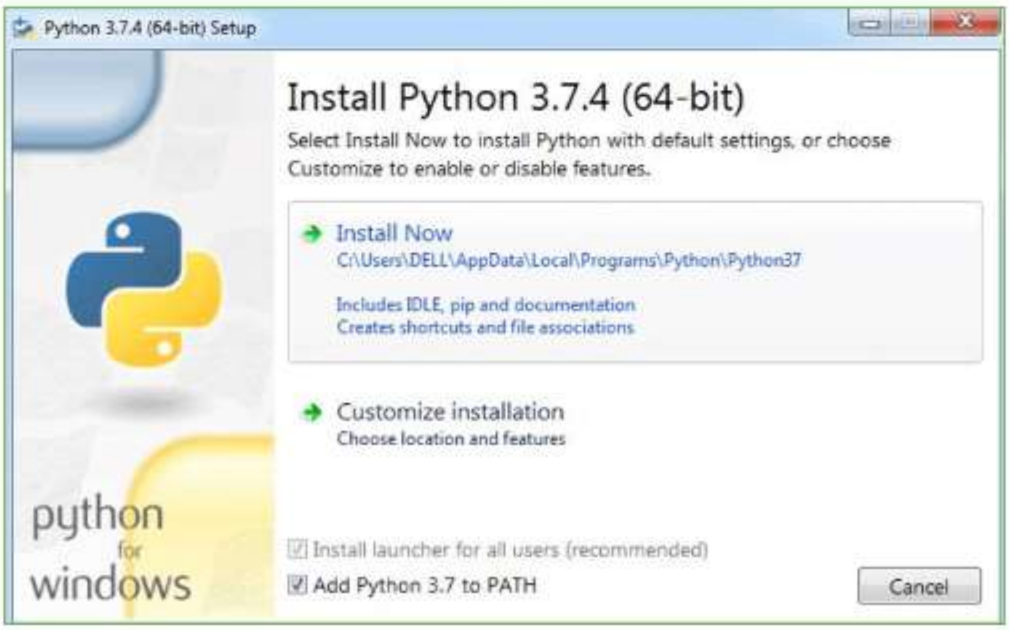
Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.


Python 3.7.4 (64-bit) Setup

Install Python 3.7.4 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

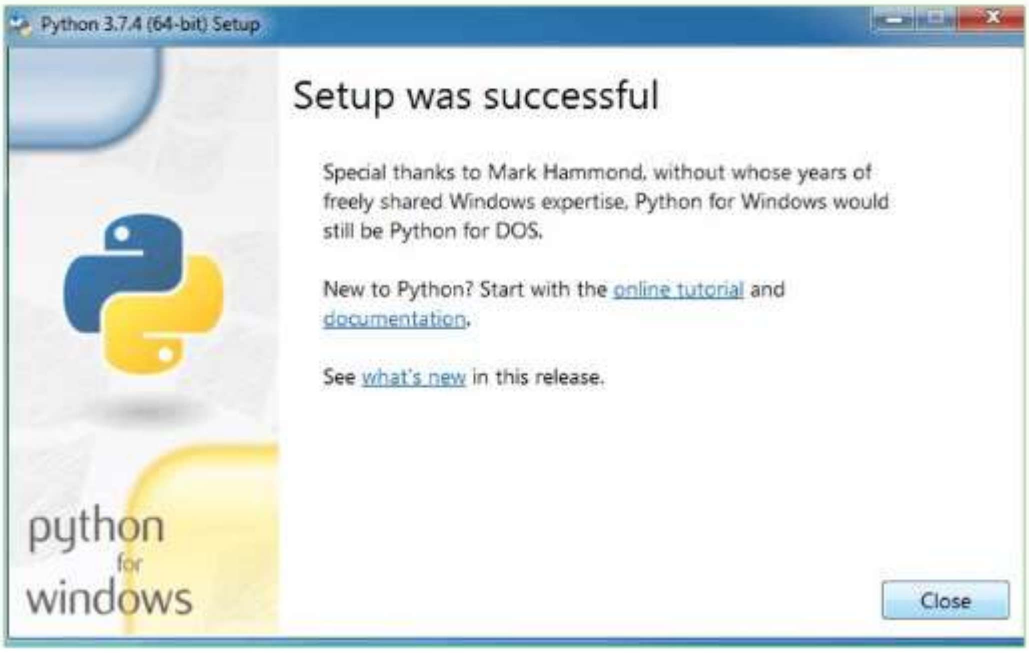
Install Now
C:\Users\DELL\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.7 to PATH

Step 3: Click on Install NOW After the installation is successful. Click on Close.


Python 3.7.4 (64-bit) Setup

Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

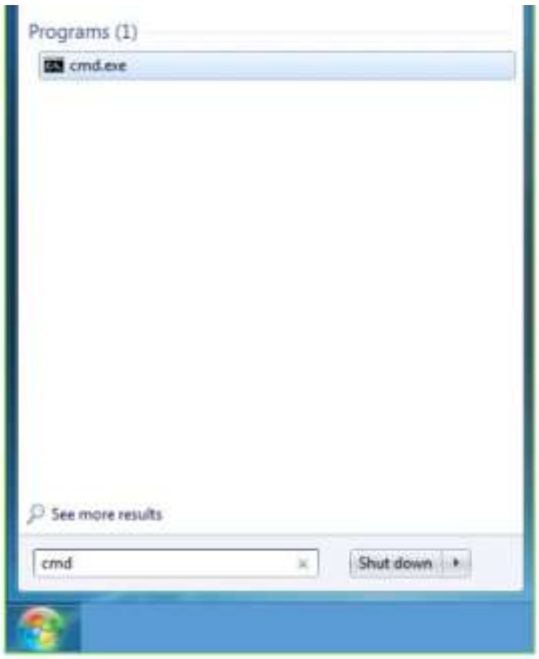
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

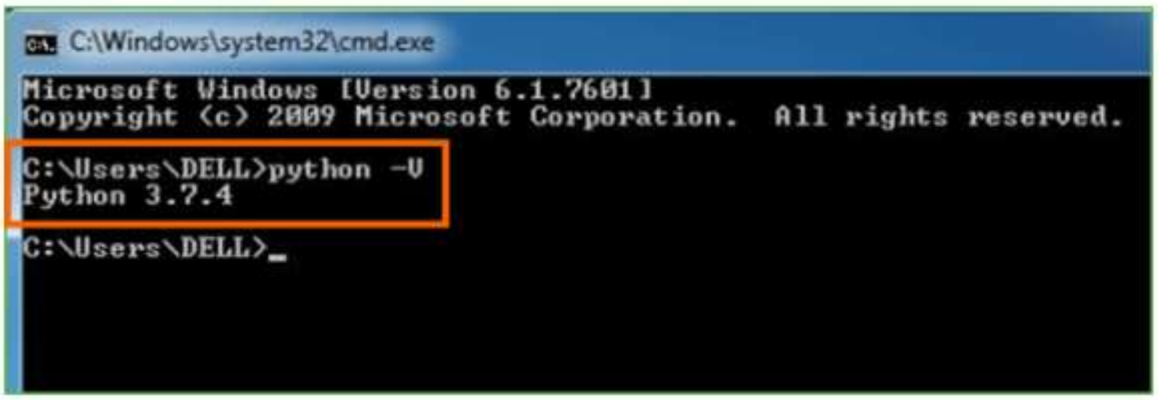
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python –V** and press Enter.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

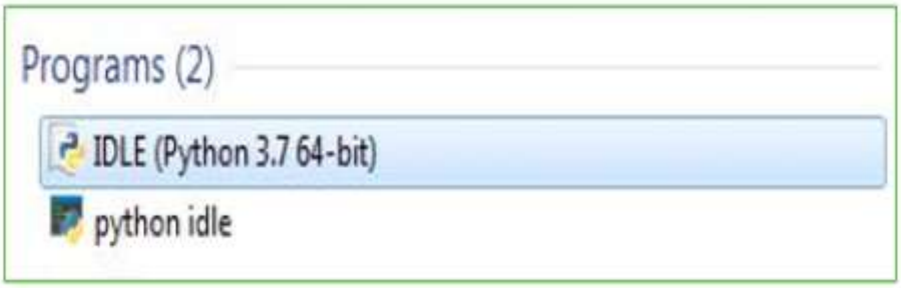
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

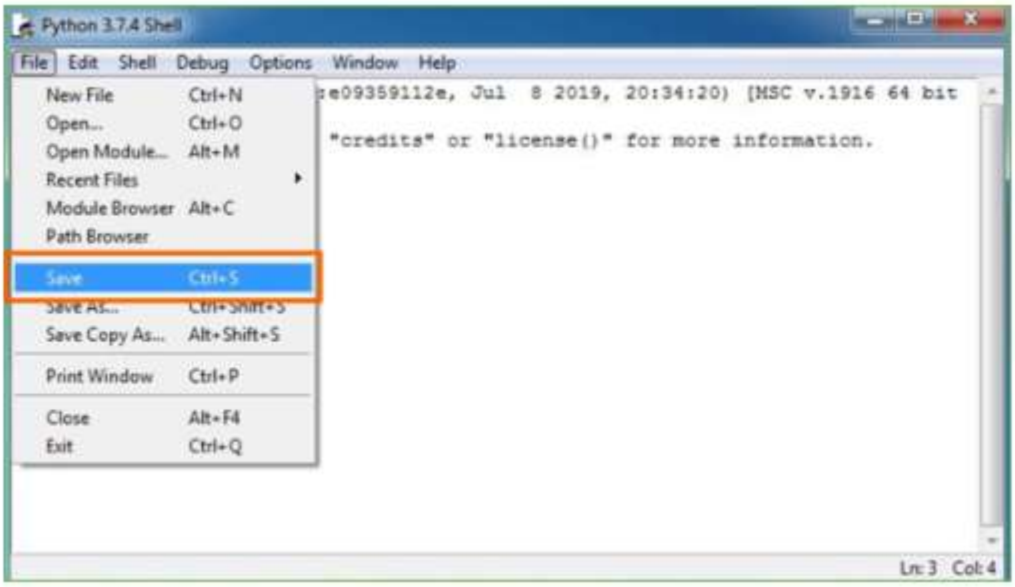
Step 1: Click on Start

Step 2: In the Windows Run command, type "python idle"



Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

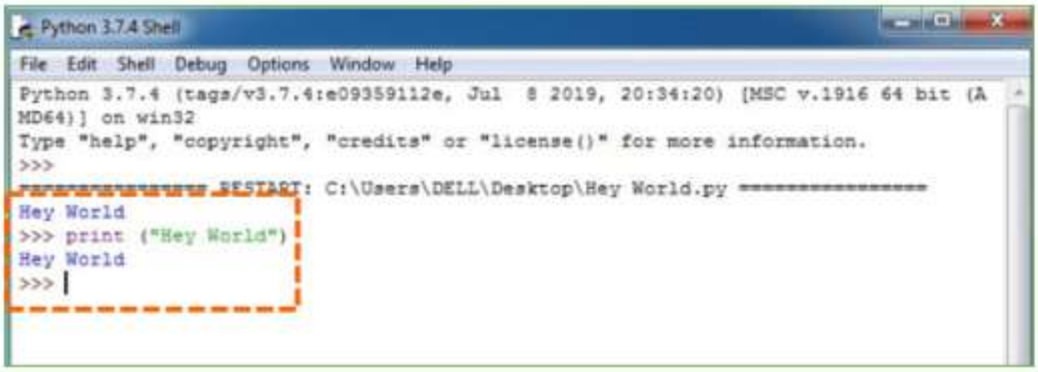
Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



```
:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
"credits" or "license()" for more information.
```

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. **enter print ("Hey World")** and Press Enter.



```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit (A
MD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: C:\Users\DELL\Desktop\Hey World.py ================
Hey World
>>> print ("Hey World")
Hey World
>>> |
```

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.
Note: Unlike Java, Python doesn't need semicolons at the end of the statements otherwise it won't work.
This stack that includes:

- world.

Django – Design Philosophies

Django comes with the following design philosophies −

- **Loosely Coupled** − Django aims to make each element of its stack independent of the others.
- **Less Coding** − Less code so in turn a quick development.
- **Don't Repeat Yourself (DRY)** − Everything should be developed only in exactly one place instead of repeating it again and again.
- **Fast Development** − Django's philosophy is to do all it can to facilitate hyper-fast development.
- **Clean Design** − Django strictly maintains a clean design throughout its own code and makes it easy to follow best web-development practices.

Advantages of Django

Here are few advantages of using Django which can be listed out here −

- **Object-Relational Mapping (ORM) Support** − Django provides a bridge between the data model and the database engine, and supports a large set of database systems including MySQL, Oracle, Postgres, etc. Django also supports NoSQL database through Django-nonrel fork. For now, the only NoSQL databases supported are MongoDB and google app engine.
- **Multilingual Support** − Django supports multilingual websites through its built-in internationalization system. So you can develop your website, which would support multiple languages.
- **Framework Support** − Django has built-in support for Ajax, RSS, Caching and various other frameworks.
- **Administration GUI** − Django provides a nice ready-to-use user interface for administrative activities.

- **Development Environment** − Django comes with a lightweight web server to facilitate end-to-end application development and testing.

As you already know, Django is a Python web framework. And like most modern framework, Django supports the MVC pattern. First let's see what is the Model-View-Controller (MVC) pattern, and then we will look at Django's specificity for the Model-View-Template (MVT) pattern.

MVC Pattern

When talking about applications that provides UI (web or desktop), we usually talk about MVC architecture. And as the name suggests, MVC pattern is based on three components: Model, View, and Controller. Check our MVC tutorial here to know more.

Django MVC – MVT Pattern

The Model-View-Template (MVT) is slightly different from MVC. In fact the main difference between the two patterns is that Django itself takes care of the Controller part (Software Code that controls the interactions between the Model and View), leaving us with the template. The template is a HTML file mixed with Django Template Language (DTL).

The following diagram illustrates how each of the components of the MVT pattern interacts with each other to serve a user request −

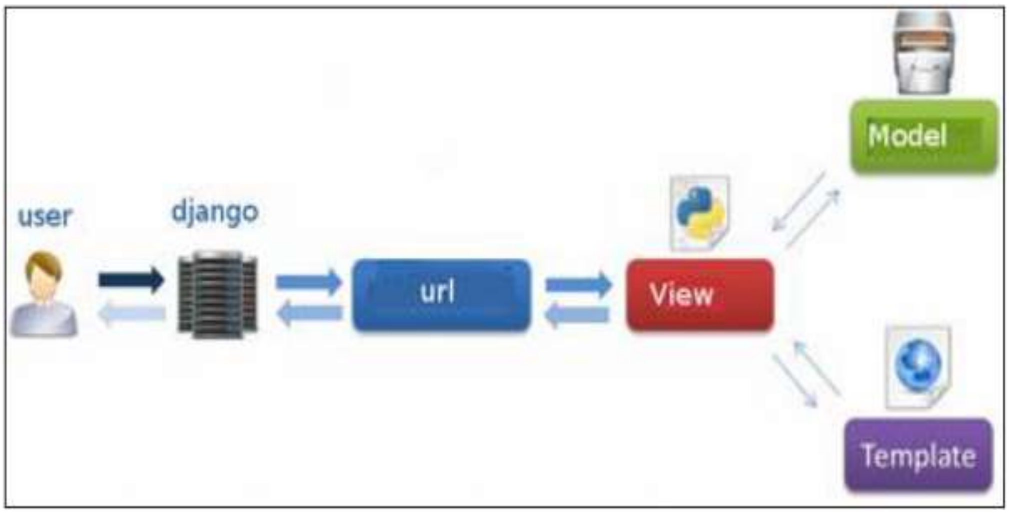


Fig 2.2: Django MVC – MVT Pattern

The developer provides the Model, the view and the template then just maps it to a URL and Django does the magic to serve it to the user.

Jupyter Notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

CHAPTER 3
SYSTEM DESIGN

3.1 Architecture Diagram :-

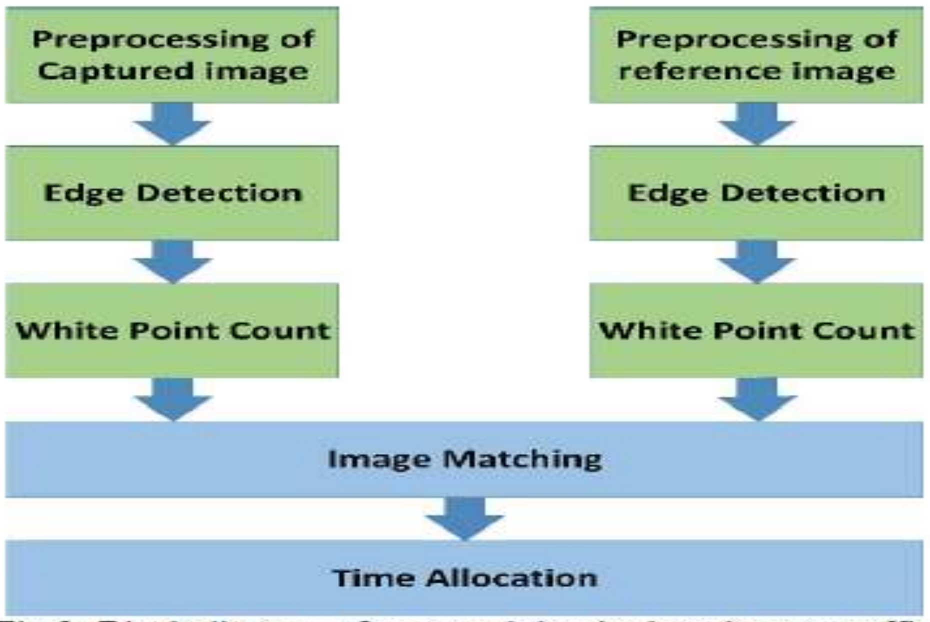


Fig 2: Block diagram of proposed density based smart traffic control system.

3.2 Module description

Admin Module: Administrator will login to application using username as 'admin' and password 'admin'. After login admin can view all registered users and all posts send by each users. Admin can send motivation messages to all depressed users. All positive and negative depression users can also be seen in the form of graph.

User Module: Users need to register with the application and then login to application to access various sub modules such as

Search Friends: Using this module user can see all peoples register with the application

Upload Posts: Using this module user can upload post in various formats such as text file, image or audio file. This application accepts only .WAV file format.

View Motivation Messages: Using this module users can view all motivation messages send by administrator.

Algorithms used in this project :-

- **K-means clustering algorithm :-**

KNN Algorithm:

- k nearest algorithm combines k nearest points based on their distances and joins them in a cluster and these clusters are then evaluated. ·

Artificial neural network :-

Artificial neural networks are one of the main tools used in machine learning. As the "neural" part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

3.3 System Specification

3.3.1 <u>Software Requirements</u>

Functional requirements for a secure cloud storage service are straightforward:

1. The service should be able to store the user's data;

2. The data should be accessible through any devices connected to the Internet;

3. The service should be capable to synchronize the user's data between multiple devices (notebooks, smart phones, etc.);

4. The service should preserve all historical changes (versioning);

5. Data should be shareable with other users;

6. The service should support SSO; and

7. The service should be interoperable with other cloud storage services, enabling data migration from one CSP to another.

• **Operating System:** Windows

• **Coding Language**: Python 3.7

• **Script:**

• **Database :**

3.3.2 Hardware Requirements:

• **Processor** - Pentium –III

• **Speed** – 2.4 GHz

• **RAM** - 512 MB (min)

• **Hard Disk** - 20 GB

• Floppy Drive - 1.44 MB

• **Key Board** - Standard Keyboard

• Monitor – 15 VGA Colour

Cloud computing has three fundamental models, these are:

3.4 Detailed Design

UML is an acronym that stands for **Unified Modeling Language**. Simply put, UML is a modern approach to modeling and documenting software. In fact, it's one of the most popular business process modeling techniques.

It is based on **diagrammatic representations** of software components. As the old proverb says: "a picture is worth a thousand words". By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

UML was created as a result of the chaos revolving around software development and documentation. In the 1990s, there were several different ways to represent and document software systems. The need arose for a more unified way to visually represent those systems and as a result, in 1994-1996, the UML was developed by three software engineers working at Rational Software. It was later adopted as the standard in 1997 and has remained the standard ever since, receiving only a few updates.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.

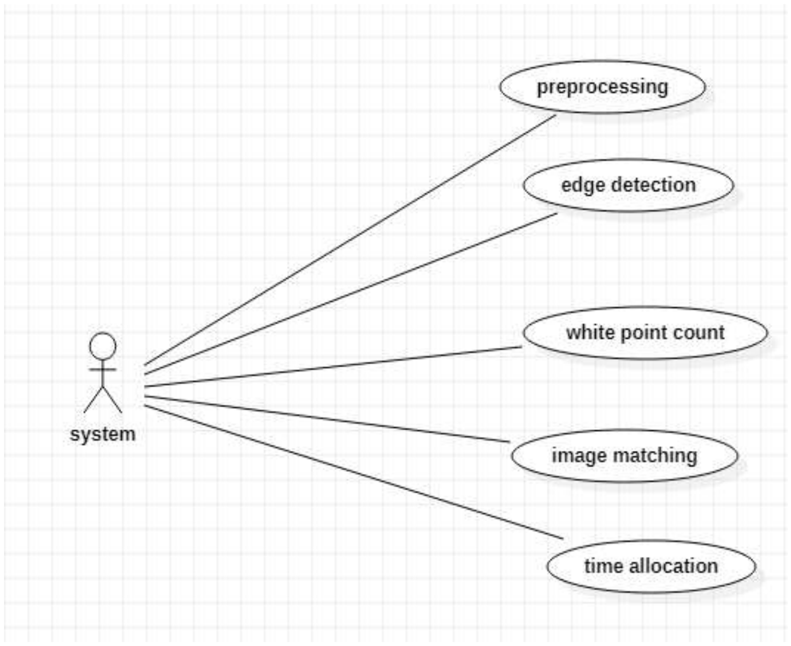
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.

6 Support higher level development concepts such as collaborations, frameworks, patterns and components.

7 Integrate best practices.

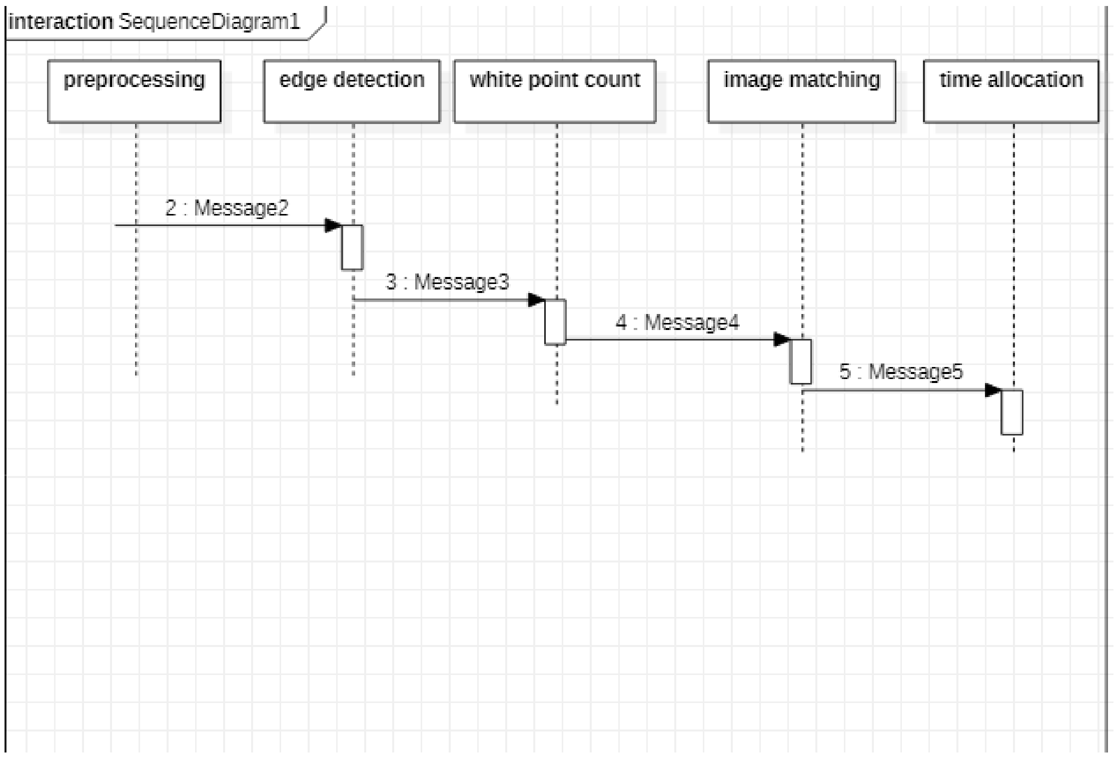
i. USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



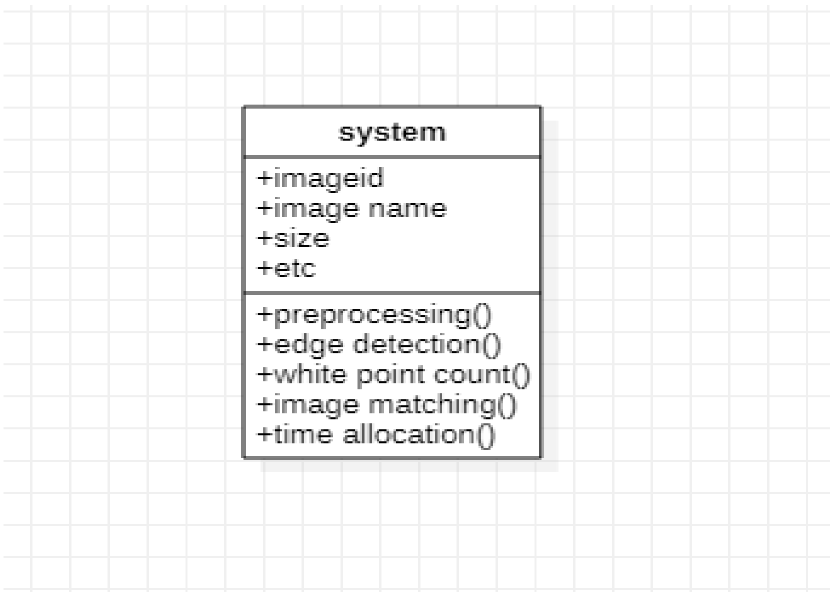
ii. SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



iii. CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

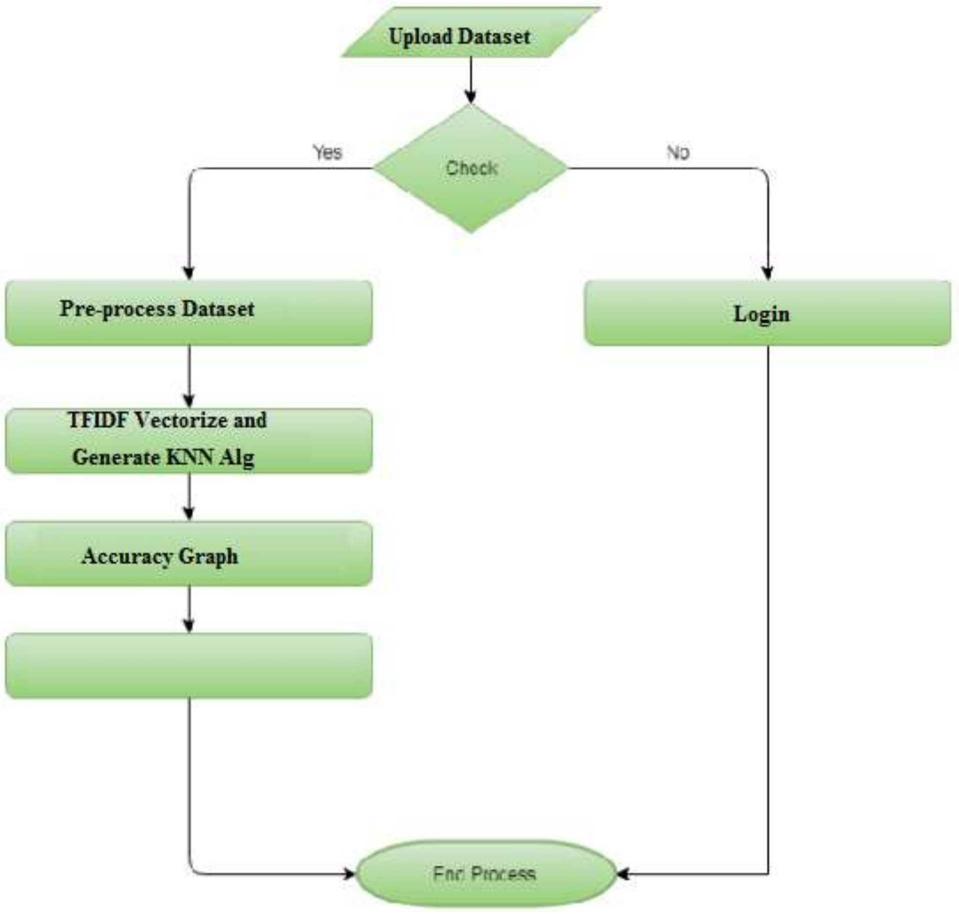


9. Data Flow diagram :-

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

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

DFD graphically representing the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system. The visual representation makes it a good communication tool between User and System designer. Structure of DFD allows starting from a broad overview and expand it to a hierarchy of detailed diagrams. DFD has often been used due to the following reasons:



Component Diagram :-

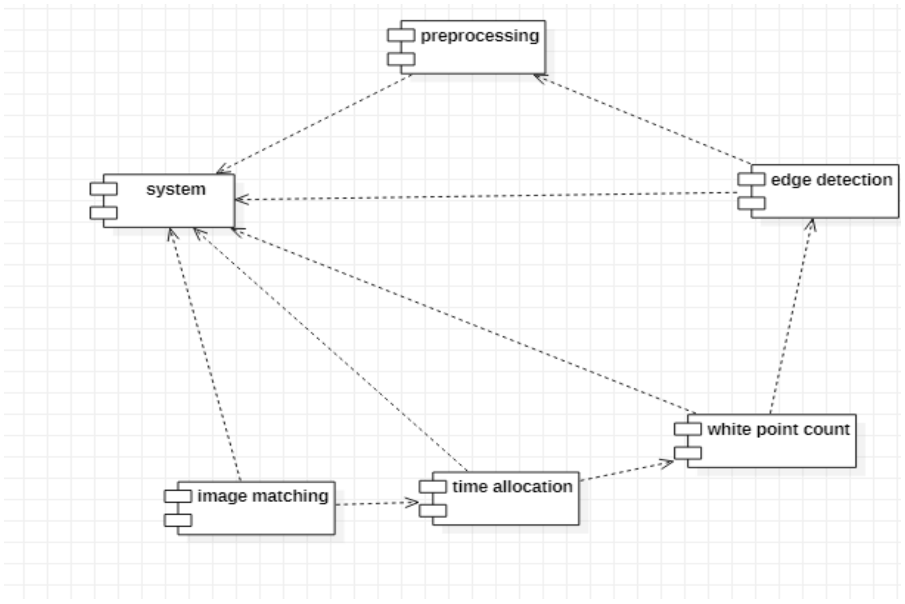
Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

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

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

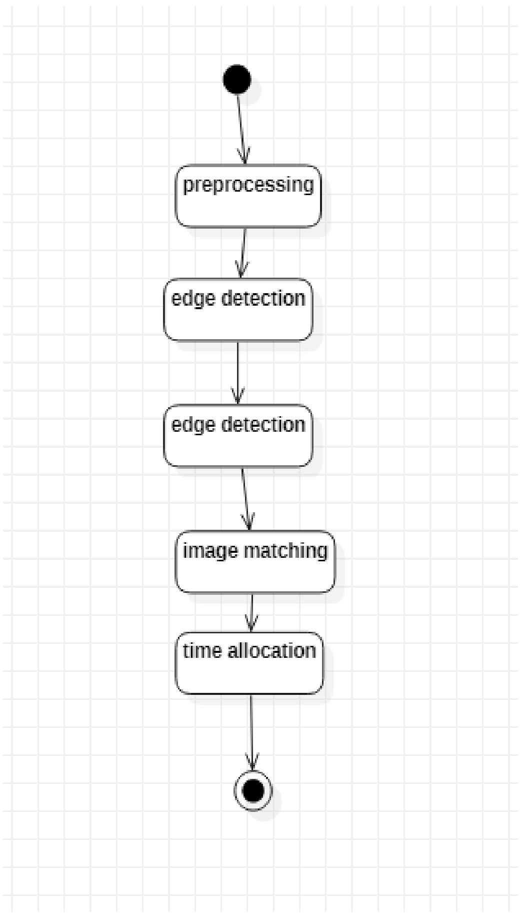
A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

UML Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.



iv. ACTIVITY DIAGRAM:

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



CHAPTER 4
IMPLEMENTATION

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import argparse
import collections
from datetime import datetime
import hashlib
import os.path
import random
import re
import sys
import tarfile

import numpy as np
from six.moves import urllib
import tensorflow as tf

from tensorflow.python.framework import graph_util
from tensorflow.python.framework import tensor_shape
from tensorflow.python.platform import gfile
from tensorflow.python.util import compat

FLAGS = None
MAX_NUM_IMAGES_PER_CLASS = 2 ** 27 - 1  # ~134M


def create_image_lists(image_dir, testing_percentage, validation_percentage):
  if not gfile.Exists(image_dir):
    tf.logging.error("Image directory '" + image_dir + "' not found.")
    return None
  result = collections.OrderedDict()
  sub_dirs = [
  os.path.join(image_dir,item)
  for item in gfile.ListDirectory(image_dir)]
sub_dirs = sorted(item for item in sub_dirs
                  if gfile.IsDirectory(item))
for sub_dir in sub_dirs:
  extensions = ['jpg', 'jpeg', 'JPG', 'JPEG']
  file_list = []
  dir_name = os.path.basename(sub_dir)
  if dir_name == image_dir:
    continue
  tf.logging.info("Looking for images in '" + dir_name + "'")
  for extension in extensions:
    file_glob = os.path.join(image_dir, dir_name, '*.' + extension)
    file_list.extend(gfile.Glob(file_glob))
  if not file_list:
    tf.logging.warning('No files found')
    continue
  if len(file_list) < 20:
    tf.logging.warning(
        'WARNING: Folder has less than 20 images, which may cause issues.')
  elif len(file_list) > MAX_NUM_IMAGES_PER_CLASS:
    tf.logging.warning(
        'WARNING: Folder {} has more than {} images. Some images will '
        'never be selected.'.format(dir_name, MAX_NUM_IMAGES_PER_CLASS))
  label_name = re.sub(r'[^a-z0-9]+', ' ', dir_name.lower())
  training_images = []
  testing_images = []
  validation_images = []
  for file_name in file_list:
    base_name = os.path.basename(file_name)
    hash_name = re.sub(r'_nohash_.*$', '', file_name)
    hash_name_hashed = hashlib.sha1(compat.as_bytes(hash_name)).hexdigest()
    percentage_hash = ((int(hash_name_hashed, 16) %
                        (MAX_NUM_IMAGES_PER_CLASS + 1)) *
                       (100.0 / MAX_NUM_IMAGES_PER_CLASS))
    if percentage_hash < validation_percentage:
      validation_images.append(base_name)
      elif percentage_hash < (testing_percentage + validation_percentage):
        testing_images.append(base_name)
      else:
        training_images.append(base_name)
    result[label_name] = {
        'dir': dir_name,
        'training': training_images,
        'testing': testing_images,
        'validation': validation_images,
    }
  return result


def get_image_path(image_lists, label_name, index, image_dir, category):
  if label_name not in image_lists:
    tf.logging.fatal('Label does not exist %s.', label_name)
  label_lists = image_lists[label_name]
  if category not in label_lists:
    tf.logging.fatal('Category does not exist %s.', category)
  category_list = label_lists[category]
  if not category_list:
    tf.logging.fatal('Label %s has no images in the category %s.',
                     label_name, category)
  mod_index = index % len(category_list)
  base_name = category_list[mod_index]
  sub_dir = label_lists['dir']
  full_path = os.path.join(image_dir, sub_dir, base_name)
  return full_path


def get_bottleneck_path(image_lists, label_name, index, bottleneck_dir,
                        category, architecture):
  return get_image_path(image_lists, label_name, index, bottleneck_dir,
                        category) + '_' + architecture + '.txt'


def create_model_graph(model_info):
  with tf.Graph().as_default() as graph:
    model_path = os.path.join(FLAGS.model_dir, model_info['model_file_name'])
    with gfile.FastGFile(model_path, 'rb') as f:
      graph_def = tf.GraphDef()
      graph_def.ParseFromString(f.read())
      bottleneck_tensor, resized_input_tensor = (tf.import_graph_def(
          graph_def,
          name='',
          return_elements=[
              model_info['bottleneck_tensor_name'],
              model_info['resized_input_tensor_name'],
          ]))
  return graph, bottleneck_tensor, resized_input_tensor


def run_bottleneck_on_image(sess, image_data, image_data_tensor,
                            decoded_image_tensor, resized_input_tensor,
                            bottleneck_tensor):
  resized_input_values = sess.run(decoded_image_tensor,
                                  {image_data_tensor: image_data})
  bottleneck_values = sess.run(bottleneck_tensor,
                               {resized_input_tensor: resized_input_values})
  bottleneck_values = np.squeeze(bottleneck_values)
  return bottleneck_values


def maybe_download_and_extract(data_url):
  dest_directory = FLAGS.model_dir
  if not os.path.exists(dest_directory):
    os.makedirs(dest_directory)
  filename = data_url.split('/')[-1]
  filepath = os.path.join(dest_directory, filename)
  if not os.path.exists(filepath):

    def _progress(count, block_size, total_size):
      sys.stdout.write('\r>> Downloading %s %.1f%%' %
                       (filename,
                    float(count * block_size) / float(total_size) * 100.0))
      sys.stdout.flush()

    filepath, _ = urllib.request.urlretrieve(data_url, filepath, _progress)
    print()
    statinfo = os.stat(filepath)
    tf.logging.info('Successfully downloaded', filename, statinfo.st_size,
                    'bytes.')
  tarfile.open(filepath, 'r:gz').extractall(dest_directory)


def ensure_dir_exists(dir_name):
  if not os.path.exists(dir_name):
    os.makedirs(dir_name)


bottleneck_path_2_bottleneck_values = {}


def create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
                           image_dir, category, sess, jpeg_data_tensor,
                           decoded_image_tensor, resized_input_tensor,
                           bottleneck_tensor):
  tf.logging.info('Creating bottleneck at ' + bottleneck_path)
  image_path = get_image_path(image_lists, label_name, index,
                              image_dir, category)
  if not gfile.Exists(image_path):
    tf.logging.fatal('File does not exist %s', image_path)
  image_data = gfile.FastGFile(image_path, 'rb').read()
  try:
    bottleneck_values = run_bottleneck_on_image(
        sess, image_data, jpeg_data_tensor, decoded_image_tensor,
        resized_input_tensor, bottleneck_tensor)
  except Exception as e:
    raise RuntimeError('Error during processing file %s (%s)' % (image_path,
                                                                 str(e)))
  bottleneck_string = ','.join(str(x) for x in bottleneck_values)
  with open(bottleneck_path, 'w') as bottleneck_file:
    bottleneck_file.write(bottleneck_string)



def get_or_create_bottleneck(sess, image_lists, label_name, index, image_dir,
                             category, bottleneck_dir, jpeg_data_tensor,
                             decoded_image_tensor, resized_input_tensor,
                             bottleneck_tensor, architecture):
  label_lists = image_lists[label_name]
  sub_dir = label_lists['dir']
  sub_dir_path = os.path.join(bottleneck_dir, sub_dir)
  ensure_dir_exists(sub_dir_path)
  bottleneck_path = get_bottleneck_path(image_lists, label_name, index,
                                        bottleneck_dir, category, architecture)
  if not os.path.exists(bottleneck_path):
    create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
                           image_dir, category, sess, jpeg_data_tensor,
                           decoded_image_tensor, resized_input_tensor,
                           bottleneck_tensor)
  with open(bottleneck_path, 'r') as bottleneck_file:
    bottleneck_string = bottleneck_file.read()
  did_hit_error = False
  try:
    bottleneck_values = [float(x) for x in bottleneck_string.split(',')]
  except ValueError:
    tf.logging.warning('Invalid float found, recreating bottleneck')
    did_hit_error = True
  if did_hit_error:
    create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
                           image_dir, category, sess, jpeg_data_tensor,
                           decoded_image_tensor, resized_input_tensor,
                           bottleneck_tensor)
    with open(bottleneck_path, 'r') as bottleneck_file:
      bottleneck_string = bottleneck_file.read()
    bottleneck_values = [float(x) for x in bottleneck_string.split(',')]
  return bottleneck_values
```

```
def cache_bottlenecks(sess, image_lists, image_dir, bottleneck_dir,
                      jpeg_data_tensor, decoded_image_tensor,
                      resized_input_tensor, bottleneck_tensor, architecture):
  how_many_bottlenecks = 0
  ensure_dir_exists(bottleneck_dir)
  for label_name, label_lists in image_lists.items():
    for category in ['training', 'testing', 'validation']:
      category_list = label_lists[category]
      for index, unused_base_name in enumerate(category_list):
        get_or_create_bottleneck(
            sess, image_lists, label_name, index, image_dir, category,
            bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,
            resized_input_tensor, bottleneck_tensor, architecture)

        how_many_bottlenecks += 1
        if how_many_bottlenecks % 100 == 0:
          tf.logging.info(
              str(how_many_bottlenecks) + ' bottleneck files created.')



def get_random_cached_bottlenecks(sess, image_lists, how_many, category,
                                  bottleneck_dir, image_dir, jpeg_data_tensor,
                                  decoded_image_tensor, resized_input_tensor,
                                  bottleneck_tensor, architecture):
  class_count = len(image_lists.keys())
  bottlenecks = []
  ground_truths = []
  filenames = []
  if how_many >= 0:
    for unused_i in range(how_many):
      label_index = random.randrange(class_count)
      label_name = list(image_lists.keys())[label_index]
      image_index = random.randrange(MAX_NUM_IMAGES_PER_CLASS + 1)
      image_name = get_image_path(image_lists, label_name, image_index,
                                  image_dir, category)
      bottleneck = get_or_create_bottleneck(
          sess, image_lists, label_name, image_index, image_dir, category,
          bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,
          resized_input_tensor, bottleneck_tensor, architecture)
      ground_truth = np.zeros(class_count, dtype=np.float32)
      ground_truth[label_index] = 1.0
      bottlenecks.append(bottleneck)
      ground_truths.append(ground_truth)
      filenames.append(image_name)
  else:
    for label_index, label_name in enumerate(image_lists.keys()):
      for image_index, image_name in enumerate(
          image_lists[label_name][category]):
        image_name = get_image_path(image_lists, label_name, image_index,
                                    image_dir, category)
        bottleneck = get_or_create_bottleneck(
            sess, image_lists, label_name, image_index, image_dir, category,
            bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,
            resized_input_tensor, bottleneck_tensor, architecture)
        ground_truth = np.zeros(class_count, dtype=np.float32)
        ground_truth[label_index] = 1.0
        bottlenecks.append(bottleneck)
        ground_truths.append(ground_truth)
        filenames.append(image_name)
  return bottlenecks, ground_truths, filenames


def get_random_distorted_bottlenecks(
    sess, image_lists, how_many, category, image_dir, input_jpeg_tensor,
    distorted_image, resized_input_tensor, bottleneck_tensor):
  class_count = len(image_lists.keys())
  bottlenecks = []
  ground_truths = []
  for unused_i in range(how_many):
    label_index = random.randrange(class_count)
    label_name = list(image_lists.keys())[label_index]
    image_index = random.randrange(MAX_NUM_IMAGES_PER_CLASS + 1)
    image_path = get_image_path(image_lists, label_name, image_index, image_dir,
                              category)
    if not gfile.Exists(image_path):
      tf.logging.fatal('File does not exist %s', image_path)
    jpeg_data = gfile.FastGFile(image_path, 'rb').read()
    distorted_image_data = sess.run(distorted_image,
                                {input_jpeg_tensor: jpeg_data})
    bottleneck_values = sess.run(bottleneck_tensor,
                              {resized_input_tensor: distorted_image_data})
    bottleneck_values = np.squeeze(bottleneck_values)
    ground_truth = np.zeros(class_count, dtype=np.float32)
    ground_truth[label_index] = 1.0
    bottlenecks.append(bottleneck_values)
    ground_truths.append(ground_truth)
  return bottlenecks, ground_truths


def should_distort_images(flip_left_right, random_crop, random_scale,
                          random_brightness):
  return (flip_left_right or (random_crop != 0) or (random_scale != 0) or
          (random_brightness != 0))


def add_input_distortions(flip_left_right, random_crop, random_scale,
                          random_brightness, input_width, input_height,
                          input_depth, input_mean, input_std):
  jpeg_data = tf.placeholder(tf.string, name='DistortJPGInput')
  decoded_image = tf.image.decode_jpeg(jpeg_data, channels=input_depth)
  decoded_image_as_float = tf.cast(decoded_image, dtype=tf.float32)
  decoded_image_4d = tf.expand_dims(decoded_image_as_float, 0)
  margin_scale = 1.0 + (random_crop / 100.0)
  resize_scale = 1.0 + (random_scale / 100.0)
  margin_scale_value = tf.constant(margin_scale)
  resize_scale_value = tf.random_uniform(tensor_shape.scalar(),
                                         minval=1.0,
                                         maxval=resize_scale)
  scale_value = tf.multiply(margin_scale_value, resize_scale_value)
  precrop_width = tf.multiply(scale_value, input_width)
  precrop_height = tf.multiply(scale_value, input_height)
  precrop_shape = tf.stack([precrop_height, precrop_width])
  precrop_shape_as_int = tf.cast(precrop_shape, dtype=tf.int32)
  precropped_image = tf.image.resize_bilinear(decoded_image_4d,
                                              precrop_shape_as_int)
  precropped_image_3d = tf.squeeze(precropped_image, squeeze_dims=[0])
  cropped_image = tf.random_crop(precropped_image_3d,
                                 [input_height, input_width, input_depth])
  if flip_left_right:
    flipped_image = tf.image.random_flip_left_right(cropped_image)
  else:
    flipped_image = cropped_image
  brightness_min = 1.0 - (random_brightness / 100.0)
  brightness_max = 1.0 + (random_brightness / 100.0)
  brightness_value = tf.random_uniform(tensor_shape.scalar(),
                                       minval=brightness_min,
                                       maxval=brightness_max)
  brightened_image = tf.multiply(flipped_image, brightness_value)
  offset_image = tf.subtract(brightened_image, input_mean)
  mul_image = tf.multiply(offset_image, 1.0 / input_std)
  distort_result = tf.expand_dims(mul_image, 0, name='DistortResult')
  return jpeg_data, distort_result


def variable_summaries(var):
  with tf.name_scope('summaries'):
    mean = tf.reduce_mean(var)
    tf.summary.scalar('mean', mean)
    with tf.name_scope('stddev'):
      stddev = tf.sqrt(tf.reduce_mean(tf.square(var - mean)))
    tf.summary.scalar('stddev', stddev)
    tf.summary.scalar('max', tf.reduce_max(var))
    tf.summary.scalar('min', tf.reduce_min(var))
    tf.summary.histogram('histogram', var)


def add_final_training_ops(class_count, final_tensor_name, bottleneck_tensor,
                           bottleneck_tensor_size):
  with tf.name_scope('input'):
    bottleneck_input = tf.placeholder_with_default(
        bottleneck_tensor,
        shape=[None, bottleneck_tensor_size],
        name='BottleneckInputPlaceholder')

    ground_truth_input = tf.placeholder(tf.float32,
                                        [None, class_count],
                                        name='GroundTruthInput')
  layer_name = 'final_training_ops'
  with tf.name_scope(layer_name):
    with tf.name_scope('weights'):
      initial_value = tf.truncated_normal(
          [bottleneck_tensor_size, class_count], stddev=0.001)

      layer_weights = tf.Variable(initial_value, name='final_weights')

      variable_summaries(layer_weights)
    with tf.name_scope('biases'):
      layer_biases = tf.Variable(tf.zeros([class_count]), name='final_biases')
      variable_summaries(layer_biases)
    with tf.name_scope('Wx_plus_b'):
      logits = tf.matmul(bottleneck_input, layer_weights) + layer_biases
      tf.summary.histogram('pre_activations', logits)

  final_tensor = tf.nn.softmax(logits, name=final_tensor_name)
  tf.summary.histogram('activations', final_tensor)

  with tf.name_scope('cross_entropy'):
    cross_entropy = tf.nn.softmax_cross_entropy_with_logits(
        labels=ground_truth_input, logits=logits)
    with tf.name_scope('total'):
      cross_entropy_mean = tf.reduce_mean(cross_entropy)
  tf.summary.scalar('cross_entropy', cross_entropy_mean)

  with tf.name_scope('train'):
      optimizer = tf.train.GradientDescentOptimizer(FLAGS.learning_rate)
      train_step = optimizer.minimize(cross_entropy_mean)

    return (train_step, cross_entropy_mean, bottleneck_input, ground_truth_input,
            final_tensor)


  def add_evaluation_step(result_tensor, ground_truth_tensor):
    with tf.name_scope('accuracy'):
      with tf.name_scope('correct_prediction'):
        prediction = tf.argmax(result_tensor, 1)
        correct_prediction = tf.equal(
            prediction, tf.argmax(ground_truth_tensor, 1))
      with tf.name_scope('accuracy'):
        evaluation_step = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
    tf.summary.scalar('accuracy', evaluation_step)
    return evaluation_step, prediction


  def save_graph_to_file(sess, graph, graph_file_name):
    output_graph_def = graph_util.convert_variables_to_constants(
        sess, graph.as_graph_def(), [FLAGS.final_tensor_name])
    with gfile.FastGFile(graph_file_name, 'wb') as f:
      f.write(output_graph_def.SerializeToString())
    return


  def prepare_file_system():
    if tf.gfile.Exists(FLAGS.summaries_dir):
      tf.gfile.DeleteRecursively(FLAGS.summaries_dir)
    tf.gfile.MakeDirs(FLAGS.summaries_dir)
    if FLAGS.intermediate_store_frequency > 0:
      ensure_dir_exists(FLAGS.intermediate_output_graphs_dir)
    return


  def create_model_info(architecture):
   architecture = architecture.lower()
   if architecture == 'inception_v3':
     data_url = 'http://download.tensorflow.org/models/image/imagenet/inception-2015-12-05.tgz'
     bottleneck_tensor_name = 'pool_3/_reshape:0'
     bottleneck_tensor_size = 2048
     input_width = 299
     input_height = 299
     input_depth = 3
     resized_input_tensor_name = 'Mul:0'
     model_file_name = 'classify_image_graph_def.pb'
     input_mean = 128
     input_std = 128
   elif architecture.startswith('mobilenet_'):
     parts = architecture.split('_')
     if len(parts) != 3 and len(parts) != 4:
       tf.logging.error("Couldn't understand architecture name '%s'",
                        architecture)
       return None
     version_string = parts[1]
     if (version_string != '1.0' and version_string != '0.75' and
         version_string != '0.50' and version_string != '0.25'):
       tf.logging.error(
           """"The Mobilenet version should be '1.0', '0.75', '0.50', or '0.25',
   but found '%s' for architecture '%s'"""",
           version_string, architecture)
       return None
     size_string = parts[2]
     if (size_string != '224' and size_string != '192' and
         size_string != '160' and size_string != '128'):
       tf.logging.error(
           """The Mobilenet input size should be '224', '192', '160', or '128',
  but found '%s' for architecture '%s'""",
           size_string, architecture)
       return None
     if len(parts) == 3:
       is_quantized = False
     else:
       if parts[3] != 'quantized':
         tf.logging.error(
             "Couldn't understand architecture suffix '%s' for '%s'", parts[3],
             architecture)
         return None
       is_quantized = True
     data_url = 'http://download.tensorflow.org/models/mobilenet_v1_'
     data_url += version_string + '_' + size_string + '_frozen.tgz'
     bottleneck_tensor_name = 'MobilenetV1/Predictions/Reshape:0'
     bottleneck_tensor_size = 1001
     input_width = int(size_string)
     input_height = int(size_string)
     input_depth = 3
     resized_input_tensor_name = 'input:0'
     if is_quantized:
       model_base_name = 'quantized_graph.pb'
     else:
       model_base_name = 'frozen_graph.pb'
     model_dir_name = 'mobilenet_v1_' + version_string + '_' + size_string
     model_file_name = os.path.join(model_dir_name, model_base_name)
     input_mean = 127.5
     input_std = 127.5
   else:
     tf.logging.error("Couldn't understand architecture name '%s'", architecture)
     raise ValueError('Unknown architecture', architecture)

   return {
       'data_url': data_url,
       'bottleneck_tensor_name': bottleneck_tensor_name,
       'bottleneck_tensor_size': bottleneck_tensor_size,
       'input_width': input_width,
       'input_height': input_height,
       'input_depth': input_depth,
       'resized_input_tensor_name': resized_input_tensor_name,
       'model_file_name': model_file_name,
       'input_mean': input_mean,
       'input_std': input_std,
     }


   def add_jpeg_decoding(input_width, input_height, input_depth, input_mean,
                         input_std):
     jpeg_data = tf.placeholder(tf.string, name='DecodeJPGInput')
     decoded_image = tf.image.decode_jpeg(jpeg_data, channels=input_depth)
     decoded_image_as_float = tf.cast(decoded_image, dtype=tf.float32)
     decoded_image_4d = tf.expand_dims(decoded_image_as_float, 0)
     resize_shape = tf.stack([input_height, input_width])
     resize_shape_as_int = tf.cast(resize_shape, dtype=tf.int32)
     resized_image = tf.image.resize_bilinear(decoded_image_4d,
                                              resize_shape_as_int)
     offset_image = tf.subtract(resized_image, input_mean)
     mul_image = tf.multiply(offset_image, 1.0 / input_std)
     return jpeg_data, mul_image


   def main(_):
     # Needed to make sure the logging output is visible.
     # See https://github.com/tensorflow/tensorflow/issues/3047
     tf.logging.set_verbosity(tf.logging.INFO)

     # Prepare necessary directories  that can be used during training
     prepare_file_system()

     # Gather information about the model architecture we'll be using.
     model_info = create_model_info(FLAGS.architecture)
     if not model_info:
       tf.logging.error('Did not recognize architecture flag')
       return -1

     # Set up the pre-trained graph.
     maybe_download_and_extract(model_info['data_url'])
     graph, bottleneck_tensor, resized_image_tensor = (
         create_model_graph(model_info))
```

```
# Look at the folder structure, and create lists of all the images.
image_lists = create_image_lists(FLAGS.image_dir, FLAGS.testing_percentage,
                                 FLAGS.validation_percentage)
class_count = len(image_lists.keys())
if class_count == 0:
  tf.logging.error('No valid folders of images found at ' + FLAGS.image_dir)
  return -1
if class_count == 1:
  tf.logging.error('Only one valid folder of images found at ' +
                   FLAGS.image_dir +
                   ' - multiple classes are needed for classification.')
  return -1

# See if the command-line flags mean we're applying any distortions.
do_distort_images = should_distort_images(
    FLAGS.flip_left_right, FLAGS.random_crop, FLAGS.random_scale,
    FLAGS.random_brightness)

with tf.Session(graph=graph) as sess:
  # Set up the image decoding sub-graph.
  jpeg_data_tensor, decoded_image_tensor = add_jpeg_decoding(
      model_info['input_width'], model_info['input_height'],
      model_info['input_depth'], model_info['input_mean'],
      model_info['input_std'])

  if do_distort_images:
    # We will be applying distortions, so setup the operations we'll need.
    (distorted_jpeg_data_tensor,
     distorted_image_tensor) = add_input_distortions(
         FLAGS.flip_left_right, FLAGS.random_crop, FLAGS.random_scale,
         FLAGS.random_brightness, model_info['input_width'],
         model_info['input_height'], model_info['input_depth'],
         model_info['input_mean'], model_info['input_std'])
  else:
    # We'll make sure we've calculated the 'bottleneck' image summaries and
    # cached them on disk.
    cache_bottlenecks(sess, image_lists, FLAGS.image_dir,
                    FLAGS.bottleneck_dir, jpeg_data_tensor,
                    decoded_image_tensor, resized_image_tensor,
                    bottleneck_tensor, FLAGS.architecture)

# Add the new layer that we'll be training.
(train_step, cross_entropy, bottleneck_input, ground_truth_input,
 final_tensor) = add_final_training_ops(
     len(image_lists.keys()), FLAGS.final_tensor_name, bottleneck_tensor,
     model_info['bottleneck_tensor_size'])

# Create the operations we need to evaluate the accuracy of our new layer.
evaluation_step, prediction = add_evaluation_step(
    final_tensor, ground_truth_input)

# Merge all the summaries and write them out to the summaries_dir
merged = tf.summary.merge_all()
train_writer = tf.summary.FileWriter(FLAGS.summaries_dir + '/train',
                                     sess.graph)

validation_writer = tf.summary.FileWriter(
    FLAGS.summaries_dir + '/validation')

# Set up all our weights to their initial default values.
init = tf.global_variables_initializer()
sess.run(init)

# Run the training for as many cycles as requested on the command line.
for i in range(FLAGS.how_many_training_steps):
  # Get a batch of input bottleneck values, either calculated fresh every
  # time with distortions applied, or from the cache stored on disk.
  if do_distort_images:
    (train_bottlenecks,
     train_ground_truth) = get_random_distorted_bottlenecks(
         sess, image_lists, FLAGS.train_batch_size, 'training',
         FLAGS.image_dir, distorted_jpeg_data_tensor,
         distorted_image_tensor, resized_image_tensor, bottleneck_tensor)
  else:
```

```
  (train_bottlenecks,
   train_ground_truth, _) = get_random_cached_bottlenecks(
       sess, image_lists, FLAGS.train_batch_size, 'training',
       FLAGS.bottleneck_dir, FLAGS.image_dir, jpeg_data_tensor,
       decoded_image_tensor, resized_image_tensor, bottleneck_tensor,
       FLAGS.architecture)
# Feed the bottlenecks and ground truth into the graph, and run a training
# step. Capture training summaries for TensorBoard with the `merged` op.
train_summary, _ = sess.run(
    [merged, train_step],
    feed_dict={bottleneck_input: train_bottlenecks,
               ground_truth_input: train_ground_truth})
train_writer.add_summary(train_summary, i)

# Every so often, print out how well the graph is training.
is_last_step = (i + 1 == FLAGS.how_many_training_steps)
if (i % FLAGS.eval_step_interval) == 0 or is_last_step:
  train_accuracy, cross_entropy_value = sess.run(
      [evaluation_step, cross_entropy],
      feed_dict={bottleneck_input: train_bottlenecks,
                 ground_truth_input: train_ground_truth})
  tf.logging.info('%s: Step %d: Train accuracy = %.1f%%' %
                  (datetime.now(), i, train_accuracy * 100))
  tf.logging.info('%s: Step %d: Cross entropy = %f' %
                  (datetime.now(), i, cross_entropy_value))
  validation_bottlenecks, validation_ground_truth, _ = (
      get_random_cached_bottlenecks(
          sess, image_lists, FLAGS.validation_batch_size, 'validation',
          FLAGS.bottleneck_dir, FLAGS.image_dir, jpeg_data_tensor,
          decoded_image_tensor, resized_image_tensor, bottleneck_tensor,
          FLAGS.architecture))
  # Run a validation step and capture training summaries for TensorBoard
  # with the `merged` op.
  validation_summary, validation_accuracy = sess.run(
      [merged, evaluation_step],
      feed_dict={bottleneck_input: validation_bottlenecks,
                 ground_truth_input: validation_ground_truth})
      validation_writer.add_summary(validation_summary, i)
      tf.logging.info('%s: Step %d: Validation accuracy = %.1f%% (N=%d)' %
                      (datetime.now(), i, validation_accuracy * 100,
                       len(validation_bottlenecks)))

    # Store intermediate results
    intermediate_frequency = FLAGS.intermediate_store_frequency

    if (intermediate_frequency > 0 and (i % intermediate_frequency == 0)
        and i > 0):
      intermediate_file_name = (FLAGS.intermediate_output_graphs_dir +
                                'intermediate_' + str(i) + '.pb')
      tf.logging.info('Save intermediate result to : ' +
                      intermediate_file_name)
      save_graph_to_file(sess, graph, intermediate_file_name)

  # We've completed all our training, so run a final test evaluation on
  # some new images we haven't used before.
  test_bottlenecks, test_ground_truth, test_filenames = (
      get_random_cached_bottlenecks(
          sess, image_lists, FLAGS.test_batch_size, 'testing',
          FLAGS.bottleneck_dir, FLAGS.image_dir, jpeg_data_tensor,
          decoded_image_tensor, resized_image_tensor, bottleneck_tensor,
          FLAGS.architecture))
  test_accuracy, predictions = sess.run(
      [evaluation_step, prediction],
      feed_dict={bottleneck_input: test_bottlenecks,
                 ground_truth_input: test_ground_truth})
  tf.logging.info('Final test accuracy = %.1f%% (N=%d)' %
                  (test_accuracy * 100, len(test_bottlenecks)))

  if FLAGS.print_misclassified_test_images:
    tf.logging.info('=== MISCLASSIFIED TEST IMAGES ===')
    for i, test_filename in enumerate(test_filenames):
      if predictions[i] != test_ground_truth[i].argmax():
        tf.logging.info('%70s  %s' %
                        (test_filename,
                        list(image_lists.keys())[predictions[i]]))

    # Write out the trained graph and labels with the weights stored as
    # constants.
    save_graph_to_file(sess, graph, FLAGS.output_graph)
    with gfile.FastGFile(FLAGS.output_labels, 'w') as f:
      f.write('\n'.join(image_lists.keys()) + '\n')


if __name__ == '__main__':
  parser = argparse.ArgumentParser()
  parser.add_argument(
      '--image_dir',
      type=str,
      default='',
      help='Path to folders of labeled images.'
  )
  parser.add_argument(
      '--output_graph',
      type=str,
      default='/tmp/output_graph.pb',
      help='Where to save the trained graph.'
  )
  parser.add_argument(
      '--intermediate_output_graphs_dir',
      type=str,
      default='/tmp/intermediate_graph/',
      help='Where to save the intermediate graphs.'
  )
  parser.add_argument(
      '--intermediate_store_frequency',
      type=int,
      default=0,
      help="""\
         How many steps to store intermediate graph. If "0" then will not
         store.\
      """
```

```
)
parser.add_argument(
    '--output_labels',
    type=str,
    default='/tmp/output_labels.txt',
    help='Where to save the trained graph\'s labels.'
)
parser.add_argument(
    '--summaries_dir',
    type=str,
    default='/tmp/retrain_logs',
    help='Where to save summary logs for TensorBoard.'
)
parser.add_argument(
    '--how_many_training_steps',
    type=int,
    default=4000,
    help='How many training steps to run before ending.'
)
parser.add_argument(
    '--learning_rate',
    type=float,
    default=0.01,
    help='How large a learning rate to use when training.'
)
parser.add_argument(
    '--testing_percentage',
    type=int,
    default=10,
    help='What percentage of images to use as a test set.'
)
parser.add_argument(
    '--validation_percentage',
    type=int,
    default=10,
    help='What percentage of images to use as a validation set.'
)
```

```
parser.add_argument(
    '--eval_step_interval',
    type=int,
    default=10,
    help='How often to evaluate the training results.'
)
parser.add_argument(
    '--train_batch_size',
    type=int,
    default=100,
    help='How many images to train on at a time.'
)
parser.add_argument(
    '--test_batch_size',
    type=int,
    default=-1,
    help="""\
    How many images to test on. This test set is only used once, to evaluate
    the final accuracy of the model after training completes.
    A value of -1 causes the entire test set to be used, which leads to more
    stable results across runs.\
    """
)
parser.add_argument(
    '--validation_batch_size',
    type=int,
    default=100,
    help="""\
    How many images to use in an evaluation batch. This validation set is
    used much more often than the test set, and is an early indicator of how
    accurate the model is during training.
    A value of -1 causes the entire validation set to be used, which leads to
    more stable results across training iterations, but may be slower on large
    training sets.\
    """
)
parser.add_argument(
    '--print_misclassified_test_images',
    default=False,
    help="""\
    Whether to print out a list of all misclassified test images.\
    """,
    action='store_true'
)
parser.add_argument(
    '--model_dir',
    type=str,
    default='/tmp/imagenet',
    help="""\
    Path to classify_image_graph_def.pb,
    imagenet_synset_to_human_label_map.txt, and
    imagenet_2012_challenge_label_map_proto.pbtxt.\
    """
)
parser.add_argument(
    '--bottleneck_dir',
    type=str,
    default='/tmp/bottleneck',
    help='Path to cache bottleneck layer values as files.'
)
parser.add_argument(
    '--final_tensor_name',
    type=str,
    default='final_result',
    help="""\
    The name of the output classification layer in the retrained graph.\
    """
)
parser.add_argument(
    '--flip_left_right',
    default=False,
    help="""\
    Whether to randomly flip half of the training images horizontally.\
    """,
    action='store_true'
)
parser.add_argument(
    '--random_crop',
    type=int,
    default=0,
    help="""\
    A percentage determining how much of a margin to randomly crop off the
    training images.\
    """
)
parser.add_argument(
    '--random_scale',
    type=int,
    default=0,
    help="""\
    A percentage determining how much to randomly scale up the size of the
    training images by.\
    """
)
parser.add_argument(
    '--random_brightness',
    type=int,
    default=0,
    help="""\
    A percentage determining how much to randomly multiply the training image
    input pixels up or down by.\
    """
)
parser.add_argument(
    '--architecture',
    type=str,
    default='inception_v3',
    help="""\
```

CHAPTER – 5

TEST RESULTS

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the
Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot ―see‖ into it. The test provides inputs and responds to outputs without considering how the software works.

5.1 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

5.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated

software components on a single platform to produce failures caused by interface defects.
The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

5.3 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

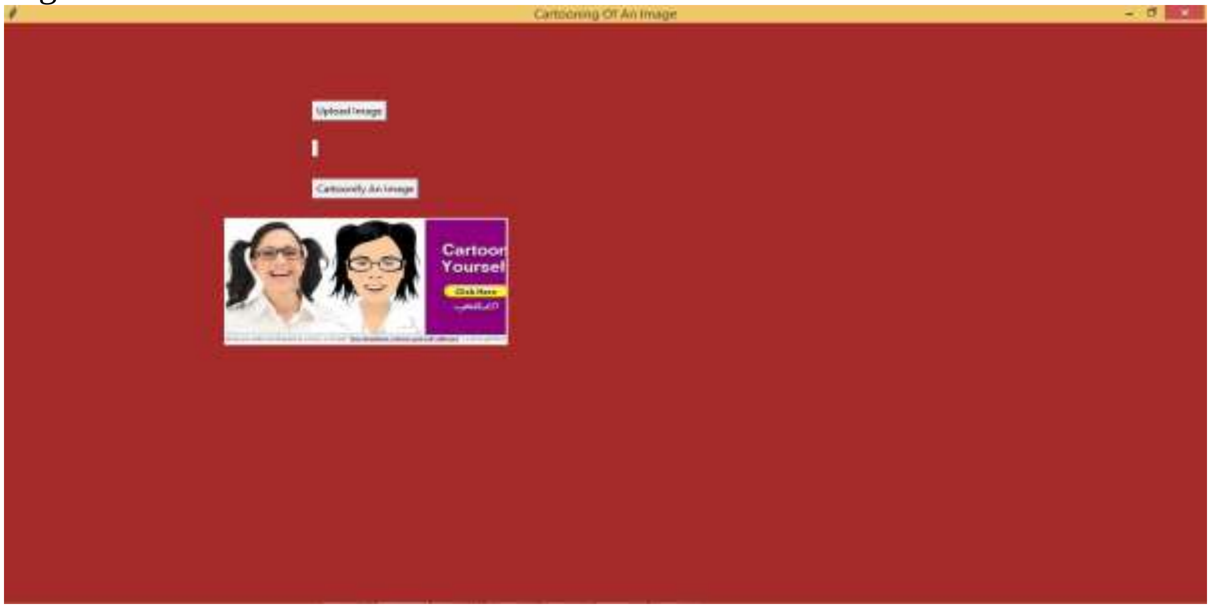
CHAPTER 6

RESULTS

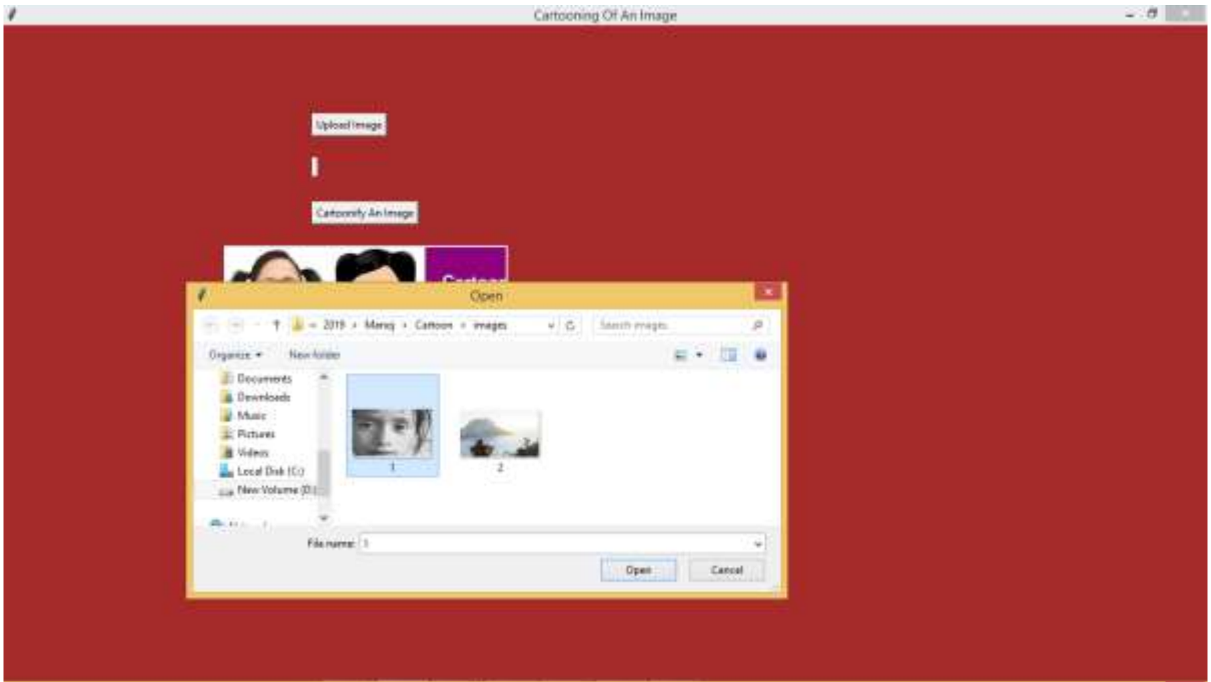
In this project we are accepting input image and then extracting edges, gray image, then applying median blur with Bilateral Filter technique to convert input image to cartoon image.
To run this project you need to have 64 bit operating system and then install python3.7 software which i will send you with code. Install this python software and then in first or second installation screen in bottom you will find checkbox saying add path to variable. You just select that checkbox and continue with installation. After installation open command prompt and execute below commands. Your system must connect to internet before executing below commands.

```
pip install numpy
pip install scipy
pip install opencv-python
pip install pillow
```

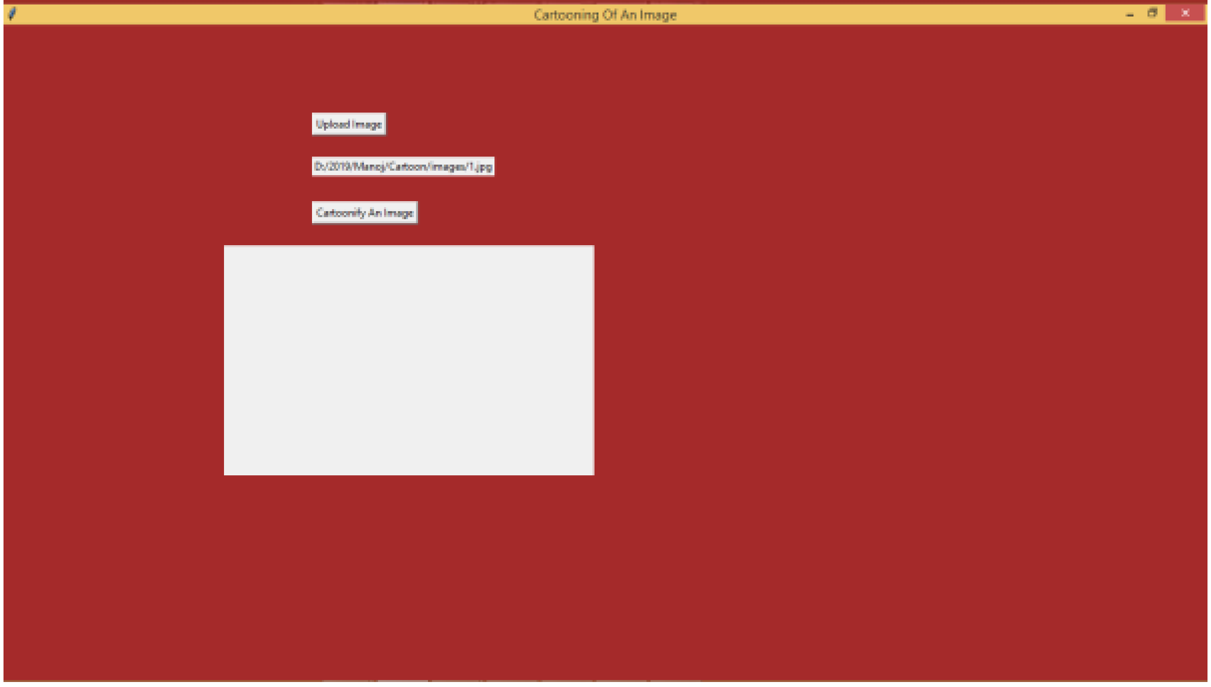
After executing above commands then double click on 'run.bat' file from Cartoon folder to get below screen



In above screen click on 'Upload Image' button and select input image

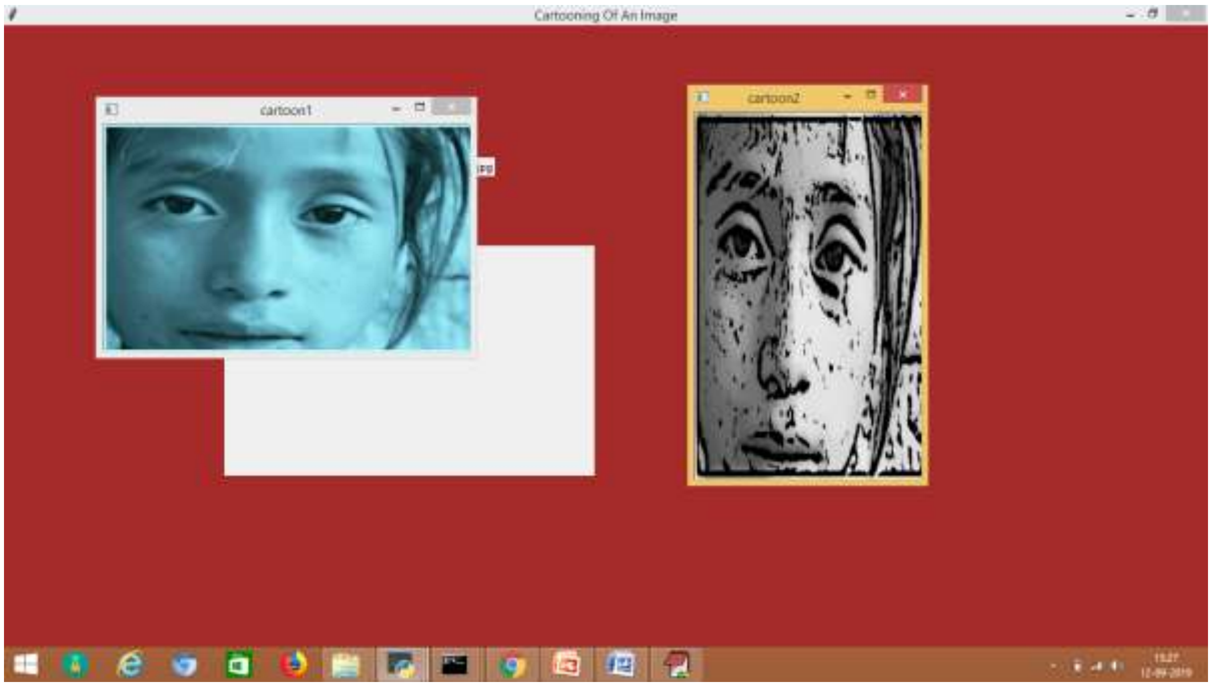


File name: 1

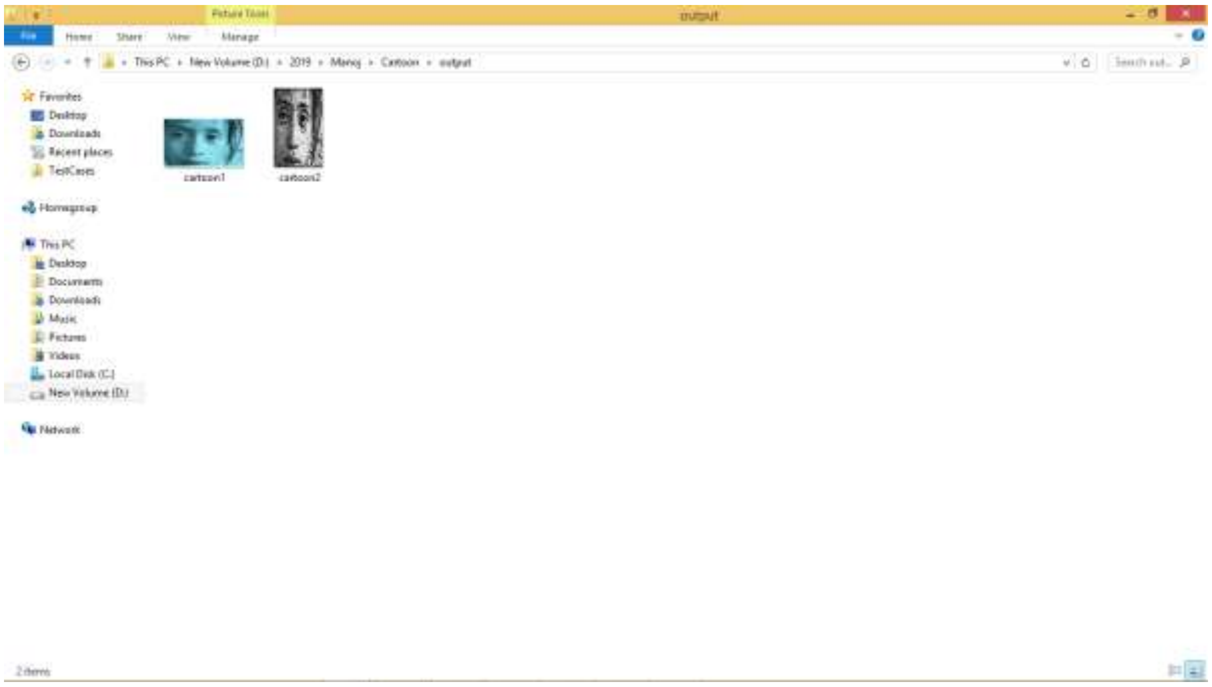


D:/2019/Manoj/Cartoon/images/1.jpg

After uploading image click on ‘Cartonify Image’ button to get below screens



We will get above carton converted images and same images will saved inside output folder



In above screen showing carton images saved at output folder.

CHAPTER 7

CONCLUSION & FUTURE WORK

CONCLUSION

This application is used to create personalised cartoons for an input image and blend them with other images as we require. We can also save the images and use them again later. Using the bilateral filter and edge detection we can create the

Future Work

cartoonized image with following characteristics: Really clear edges and Homogeneous colours. Starting from an original image taken with

a camera we’re going to give it a cartoon effect keeping in mind these characteristics.

CHAPTER-8
REFERENCES

M. Sweet, “Traffic Congestion’s Economic Impacts: Evidence from US Metropolitan Regions,” ***Urban Studies,*** **vol. 51, no. 10, pp. 2088–2110, Oct. 2013**

Traffic congestion alleviation has long been a common core transport policy objective, but it remains unclear under which conditions this universal byproduct of urban life also impedes the economy. Using panel data for 88 US metropolitan statistical areas, this study estimates congestion’s drag on employment growth (1993 to 2008) and productivity growth per worker (2001 to 2007).

Md. Munir Hasan, Gobinda Saha, Aminul Hoque and Md. Badruddoja Majumder, “Smart Traffic Control System with Application of Image Processing Techniques,”in 3rd International Conference on Informatic Electronics & Vision, Dhaka, May 2014.

In this paper we propose a method for determining traffic congestion on roads using image processing techniques and a model for controlling traffic signals based on information received from images of roads taken by video camera. We extract traffic density which corresponds to total area occupied by vehicles on the road in terms of total amount of pixels in a video frame instead of calculating number of vehicles. We set two parameters as output, variable traffic cycle and weighted time for each road based on traffic density and control traffic lights in a sequential manner.