

FAKE LOGO DETECTION SYSTEM

A Project Report submitted in partial fulfillment of the degree of the

MASTERS OF COMPUTER SCIENCE

By

MADDELA AKHIL

23000I1042

GORRE ARUNDHATHI

23000I1035

NAGANI SAI KUMAR

23000I1026

Under the Guidance of

Mrs. DR B. MANJULA
(ASSISTANT PROFESSOR)



Department of Computer Science

UNIVERSITY COLLEGE KAKATIYA UNIVERSITY

VIDYARANYAPURI, HANAMKONDA, (T.S), 506009.

(Accredited with 'A' Grade by NAAC)

(2022-2024)

UNIVERSITY COLLEGE KAKATIYA UNIVERSITY

VIDYARANYAPURI, HANAMKONDA, (T.S), 506009.

(Accredited with 'A' Grade by NAAC)

CERTIFICATE



This is to certify that the Project Report entitled “**FAKE LOGO DETECTION SYSTEM**” is a bonafide work of the students’ **MADDELA AKHIL, GORRE ARUNDHATHI, NAGANI SAI KUMAR** bearing Roll No’s **23000I1042, 23000I1035, 23000I1026** submitted in partial fulfillment of the requirements for the award of the degree of **MASTERS OF COMPUTER SCIENCE** during the academic year **2023-2024**.

Guide

Head of the Department

Principal

ACKNOWLEDGEMENT

At the outset, I would like to express my sincere thanks to my Project Guide **Mrs DR B. MANJULA** for guiding my project work with meticulous care and patience.

I express my special thanks to **Mrs. DR. B. RAMA** Head, Department of Computer Science, **UNIVERSITY COLLEGE KAKATIYA UNIVERSITY** Hanamkonda for his inspiring guidance and cooperation.

Finally, I would like to thank all those who helped me directly or indirectly in preparation of my project work.

MADDELA AKHIL	23000I1042
GORRE ARUNDHATHI	23000I1035
NAGANI SAI KUMAR	23000I1026

ABSTRACT

Every year, top brands lose a significant percentage of their sales to unauthorized brands which are using their same logo but in a slightly different way. Moreover such products are usually of a low quality, they also end up damaging the credibility of the brand. Many times buyers get cheated out of their earned money as they end up buying the fake products. The old system provides the way to find only the brand logo so the system is not so useful to prevent people from buying forged projects. The existing system can find only the logo but can't find whether it's real or fake. Since the difference is more minute people can't identify it easily. This Logo Detection System aims to help consumers distinguish fake from the original product. Using this system, a consumer can verify whether a product real or forged. This System can also be helpful for brands struggling to fight against forged products. This system allows the users to complain the original brand about the fraudulent activities, so that the brand owners can take sufficient action to reduce the fraudulent activities and prevent the people from buying the fake products and also they can prevent themselves from damaging their credibility.

CONTENTS

1. INTRODUCTION	Pg no.
1.1 Logo Detection.....	1-2
1.2 Scope of the Project.....	3
1.3 Organization of Project Report.....	3
1.4 Conclusion.....	3
2. LITERATURE REVIEW	
2.1 Introduction.....	4
2.2 Related Works.....	4-7
2.3 Conclusion.....	7
3. SYSTEM DESIGN	
3.1 Existing System.....	8
3.2 Proposed System.....	8
3.3 System Architecture.....	9-10
3.4 Use-case Diagram.....	11
3.5 Conclusion.....	12
4. SYSTEM IMPLEMENTATION	
4.1 Introduction.....	13
4.2 Hardware and Software Specification.....	13
4.3 Technologies Used.....	14-19
5. TESTING	
5.1 Introduction.....	20
5.2 List of Modules.....	20-28
6. CONCLUSION	
6.1 Conclusion.....	29
6.2 Future Enhancement.....	29
7. APPENDIX	
7.1 Appendix1.....	30-31
7.2 Appendix2.....	32-33
7.3 Appendix3.....	34
8. BIBLIOGRAPHY	35-36

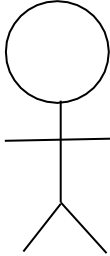
LIST OF FIGURES

Figure Number	Figure Name	PgNo.
3.3a	Architecture Diagram	9
3.4a	Use Case Diagram	11
4.3.3a	Yolo Architecture	18
5.2.1 a	User Interface	22
5.2.1 b	Training Model	22
5.2.1 c	Training Graph	23
5.2.2a	Detection Camera	24
5.2.3a	Detecting Original Balmain Logo	25
5.2.3b	Detecting Original Adidas Logo	27
5.2.3c	Detecting Fake Adidas Logo	27

LIST OF ABBREVIATIONS

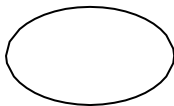
YOLO	- YOULOOKONLYONCE
OPENCV	- OPEN SOURCE COMPUTER VISION LIBRARY
SCL	- SYNTHETIC CONTEXT LOGO
UML	- UNIFIED MODELLING LANGUAGE

SYSTEM AND NOTATIONS



Actor

Standard UML icon for an actor is “Stickman” icon with the name of the actor above or below the icon



Usecase

Every usecase must have a name. The usecase is shown as an ellipse containing the name of the usecase



An association between an actor and use case indicates that the actor and the usecase communicate with each other. An actor could be associated with one or several use cases

1. INTRODUCTION

1.1 LOGO DETECTION

Logo detecting is one of the highly challenging tasks in current computer vision. Logo detection in the unclear and unconstrained images is very crucial for type of the real world vision applications. The challenging task in detection is due to the different sizes of the logos present. The existing Logo detection systems traditionally need a small logo class with large size data with bounding boxes using the real time object detection. In this System the logo is detected using the You Look Only Once (YOLO) algorithm.

In this work the logo is identified and it is displayed in a grid using the real time detection systems. If the logo is original the user can move further and buy the product. When the logo is fake the user can report it to the original brands through this system. The main motto of this idea is to restrain people from buying the fake products and help the brands to maintain their credibility.

With the increase in digital marketing and e-commerce, logos have become an essential aspect of a brand's identity. Logos are used to establish brand recognition and build consumer trust. However, the proliferation of counterfeit products has made it increasingly difficult to distinguish between genuine and fake logos. This has led to the need for effective fake logo detection systems. Fake logo detection involves identifying and distinguishing between genuine and fake logos. It is a challenging task due to the high degree of similarity between genuine and fake logos. Therefore, several techniques have been developed to detect fake logos, including image analysis, machine learning, and deep learning.

Python is a popular programming language for developing machine learning and deep learning models. It offers several libraries and frameworks that simplify the process of developing and implementing these models. This has led to the development of several fake logo detection systems using Python.

In this review, we provide a comprehensive overview of the different techniques that have been used for detecting fake logos using Python. We discuss the strengths and weaknesses of each technique and highlight the challenges that still need to be addressed in this field. This review will serve as a useful resource for researchers and practitioners who are interested in developing and implementing fake logo detection systems.

Fake logo detection is a challenging problem that has received significant attention in recent years. With the rise of e-commerce and digital marketing, logos have become an important aspect of branding and consumer trust. However, the proliferation of counterfeit products has made it increasingly difficult to distinguish between genuine and fake logos, leading to the need for effective fake logo detection systems.

Previous research on fake logo detection has focused on different techniques, including image analysis, machine learning, and deep learning. These techniques have been used to develop various approaches for detecting fake logos, such as texture analysis, feature extraction, and classification.

Python has become a popular programming language for developing machine learning and deep learning models due to its simplicity, flexibility, and extensive libraries and frameworks. Several Python libraries and frameworks, such as OpenCV, TensorFlow, and PyTorch, have been used for developing fake logo detection systems.

Different datasets have been used for training and testing fake logo detection systems, such as the Logos in the Wild dataset and the Fake Logos dataset. These datasets contain a large number of images of logos and are used to train and test the accuracy of the detection systems.

Fake logo detection has several applications, including e-commerce, anti-counterfeiting, and brand protection. Effective fake logo detection systems can help prevent the sale of counterfeit products and protect the reputation of brands.

In this review, we aim to provide a comprehensive overview of the different techniques that have been used for detecting fake logos using Python. We will review the strengths and weaknesses of each technique, describe the different datasets used for training and testing, and highlight the challenges that need to be addressed in this field.

1.2 SCOPE OF THE PROJECT

To help the people by making them buy only the original products and also help the brands to maintain their standard. To help the brands to know where the fake products are being sold and help to find them out and take sufficient action on that case. Finally it helps every people to buy quality and original products rather than the fake one.

1.3 ORGANIZATION OF THE PROJECT REPORT

The report begins by examining the problem space. It investigates current and upcoming studies on the subject. It starts with Logo detection and recognition and then moves on to dataset construction. The study then details the constraints imposed by these arrangements and how the proposed system will address them. It takes you on a tour of the System's architecture and functionality. Finally, it considers the suggested system's scope, prospective applications, and conclusion.

1.4 CONCLUSION

In order help the people to buy the quality products we use this System. As logos are the common mode of identification of the original product, it is very important to find the originality of the logo of the product while purchasing. In these ways this system will help the brand owners and the customers.

2. LITERATURE REVIEW

2.1 INTRODUCTION

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is a secondary source and discusses published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and maybe just a simple summary of sources.

2.2 RELATED WORKS

Deep Learning Logo Detection with Data Expansion by Synthesizing Context

AUTHOR: Hang Su, Xiatian Zhu ,Shaogang Gong

YEAR: 2017

METHODOLOGIES USED: YOLO

DESCRIPTION:

Logo detection in unconstrained image is challenging, particularly when only very sparse labelled training images are accessible due to high labelling costs. Synthetic Context Logo (SCL) training is the method to increase model robustness against unknown background clutters, resulting in superior logo detection performance.

Automatic Logo Detection and Extraction using Singular Value Decomposition

AUTHOR: Umesh D. Dixit and M. S. Shirdhonkar

YEAR: 2016

METHODOLOGIES USED: Singular Value Decomposition

DESCRIPTION:

Automatic logo detection and extraction from document image implementation with the use of Mathematical tool Singular Value Decomposition (SVD). The method proposed is tested publicly available realistic document image database Tobacco-800. We also compared the results of proposed method with current method. Average logo detection rate is achieved with proposed method.

Detecting TV logos from Web-Scale Videos

AUTHOR: Qiting Ye, Zhao Luo, Xiabing Xiao, Shiming Ge

YEAR: 2017

METHODOLOGIES USED: Deep Learning, Convolutional neural Networks

DESCRIPTION:

The key frame module first extracts several frames with the shot segment detection. In this manner, the data can be drastically reduced. Then, the spatial verification module takes the logo as the input and identifies them as real logos or not by performing the classification task within a ResNet network. Finally, the temporal verification module further identifies the detection results by checking the temporal consistency of logo locations.

Deep Learning for Logo Recognition

AUTHOR: Hang Su, Xiatian Zhu, Shaogang Gong

YEAR: 2017

METHODOLOGIES USED: Deep Learning

DESCRIPTION:

Our recognition pipeline is composed of a logo region proposal followed by CNN specifically trained for logo classification and identification, even if they are not right localized. Experiments are carried out on the FlickrLogos-32 dataset classes, and we evaluate the effect of the recognition performance of synthetic versus real data augmentation, and image pre-processing. Experimental result confirms the feasibility of proposed method.

The Halal Logo Classification by Using NVIDIA DIGITS

AUTHOR: Hendrick, Chih-Min, Wan, Aripriharta, Ciou-Guo, Jhe, Ping-Cheng, TsuGwo-Jia, Jong

YEAR: 2019

METHODOLOGIES USED: Deep Learning, Caffe, Halal Logo, GoogleLeNet

DESCRIPTION:

The Halal logo is not the same for every country. In both Halal logo Indonesia and Taiwan are different. In this research, the deep learning methodology has been applied to classify the halal logo. The classification of logo is based on the caffe framework with GoogleLeNet architecture. As the datasets, the halal logo and soft drink logo has been created. This purpose of the study is to produce a deep learning pre-trained model.

Signature and Logo Detection using Deep CNN for Document Image Retrieval.

AUTHOR: Nabin Sharma, Ranju Mandal, Rabi Sharma, UmapadaPal and Michael Blumenstein.

YEAR: 2019

METHODOLOGIES USED: R-CNN; Deep Learning; Document retrieval; Signature detection.

DESCRIPTION:

A large intra-category variance among signature and logo samples possess challenge to traditional hand-crafted feature extraction-based approaches. Hence, the potential of deep learning-based object detectors namely, Faster R-CNN and YOLO v2 were examined for automatic detection of signatures and logos from scanned administrative document. Four different network models namely ZF, VGG16, VGGM and YOLO v2.

Sport Teams Logo Detection Based On Deep Local Features

AUTHOR: Andrey Kuznetsov, Andrey Savchenko

YEAR: 2019

METHODOLOGIES USED: logo detection, sport logo, NHL dataset, local descriptors, SIFT, SURF, ORB, BRISK, FREAK, AKAZE, DELF

DESCRIPTION:

Logo detection in an unconstrained environment is an experimental task. The problem lies in the small size of logo that are detected and different representation of logos. We propose an approach based on local descriptor calculation (SIFT, SURF, ORB, BRISK, FREAK, AKAZE) and provide the results of conducted experiments. Second, we approach based on deep local features (DELF), which are pre-trained on Google Landmarks dataset, and compare them with classic local features.

Unconstrained Logo Dataset with Evaluation by Deep learning Methods

AUTHOR: Nhat-DuyNguyen, ThuaNguyen, TienDo, ThanhDucNgo and Duy-Dinh Le

YEAR: 2019

METHODOLOGIES USED: logo detection, sport logo, NHL dataset, local descriptors, SIFT, SURF, ORB, BRISK, FREAK, AKAZE, and DELF

DESCRIPTION:

Detecting the appearance of logo in image is applied to many applications, such as brand recognition for marketing analysis and intellectual property rights protection we provide an evaluation of the state of the art model based on Deep learning including YOLO, Faster

RCNN, Mask RCNN, RetinaNet in order to illustrate how well these models overcome unconstrained condition. Comparative evaluations demonstrate the best performance on U15-Logos.

Clothing Brand Prediction via Logo Recognition

AUTHOR: Kuan-HsienLiu1, FeiWang2, Tsung-JungLiu2

YEAR: 2020

METHODOLOGIES USED: YOLOV3

DESCRIPTION:

Several dense blocks are designed to improve prediction accuracy based on clothing brand logo. We construct a new clothing dataset with brand and logo information to facilitate this task. In the experiment, we show our method can achieve better performance than some state-of-the-art methods.

Detecting Motion Blurred Vehicle Logo in IoV Using Filter-DeblurGAN and VL-YOLO

AUTHOR : Linghua Zhou, Weidong Min, Member, IEEE, Deyu Lin, Member,IEEE, Qing Han, Ruikang Liu

YEAR : 2020

METHODOLOGIES USED: Filter-DeblurGAN and VL-YOLO

DESCRIPTION:

A new approach is proposed to detect vehicle logo under motion blur with the combination of Filter-DeblurGAN and Vehicle Logo-YOLO. Filter-DeblurGAN possesses a judgment mechanism, which can determine whether the image need to be deblurred based on the degree of blur in every image. It can also deblur images with random resolution. Filter-DeblurGAN solves the defect that DeblurGAN lacks the judgment mechanism and is too hard resolution.

2.3 CONCLUSION

This chapter presented the advantages and limitations of different logo detection applications. This helps in developing and identifying the ideal techniques that can be used to develop an efficient application with high accuracy and minimal limitations. There has been much research and discussion conducted on logo detection applications.

3. SYSTEM DESIGN

3.1 EXISTING SYSTEM

The existing system provides the way to find only the brand logo so the system is not so useful to prevent people from buying forged projects. The existing system can find only the logo but can't find whether its real or fake. Since the differences are more minute people can't identify it easily. In The Existing system just the logo name is displayed with the grid like structure around the logo and not much thing to find the fake one. The old systems that didn't use the Yolo very much struggled to find the logo in the real time environment. Since the old systems scanning speed is less and it doesn't have the ability to find the small logo images.

3.2 PROPOSED SYSTEM

The proposed fake Logo Detection System aims to help consumers distinguish fake from the original product. Using this system, a consumer can verify whether a product real or forged. This System can also be helpful for brands struggling to fight against forged products. This system allows the users to complain the original brand about the fraudulent activities, so that the brand owners can take sufficient action to reduce the fraudulent activities and prevent the people from buying the fake products and also they can prevent themselves from damaging their credibility.

3.3 SYSTEM ARCHITECTURE

Overall system architecture for the proposed system is shown in Figure 3.3a as below.

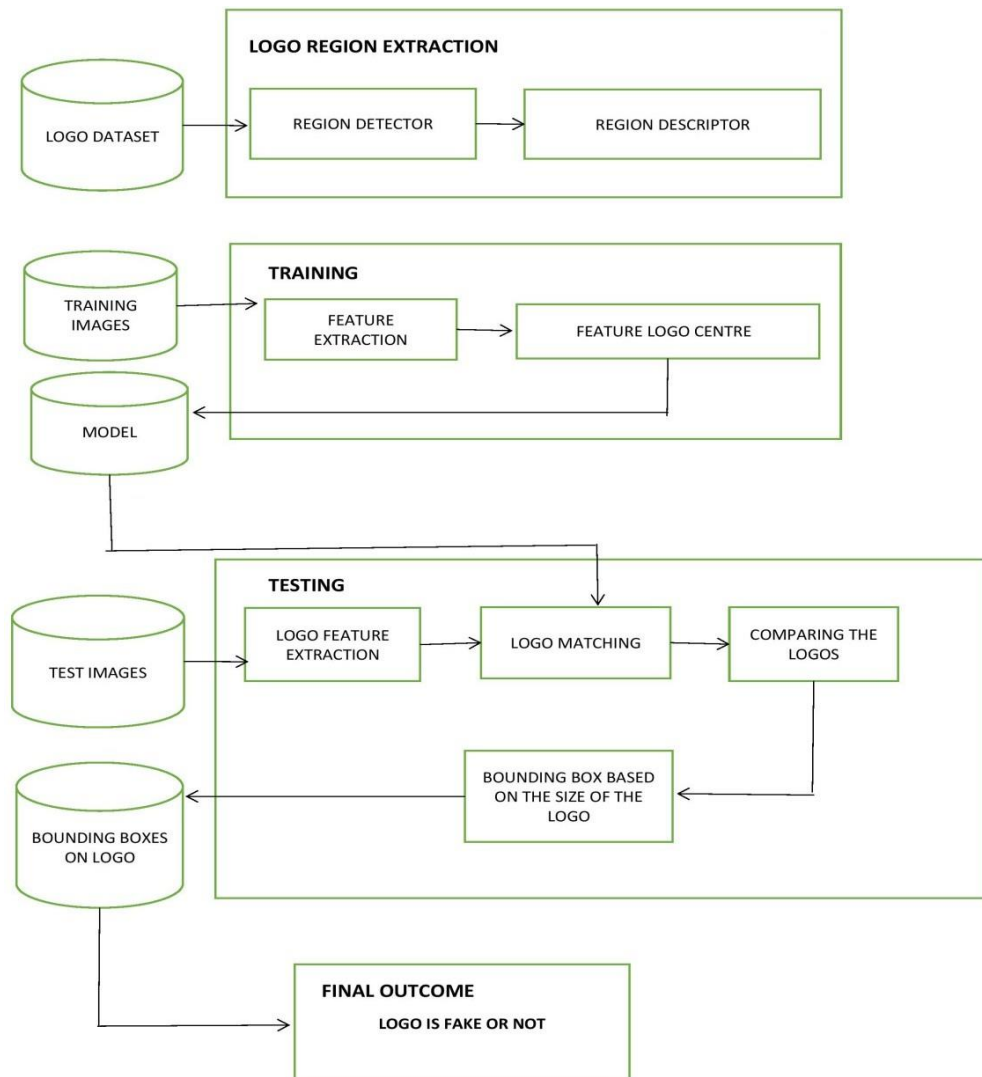


Fig 3.3a Architecture Diagram

The software system design, program methodologies, and system approaches employed in the creation of programming are described in this work. Accepting the data and commands, executing different commands and getting the required output. In this system the input is the logo of the product detected in real time and the output is whether the logo is fake or real. The system first compares the input with the trained dataset from the admin side. For comparing the images in the dataset the Tensorflow module is used which is one of the main model of the deep learning library .For real time input OpenCv is used. OpenCv is the python library helps to access the camera. For the real time detection and analyzing the originality of logo the Yolo algorithm is been used.

3.4 USECASE DIAGRAM

The use case for the proposed system. System modeling has been done through a use case that represents a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

The overall system architecture for the proposed system is shown in Figure 3.4a as below

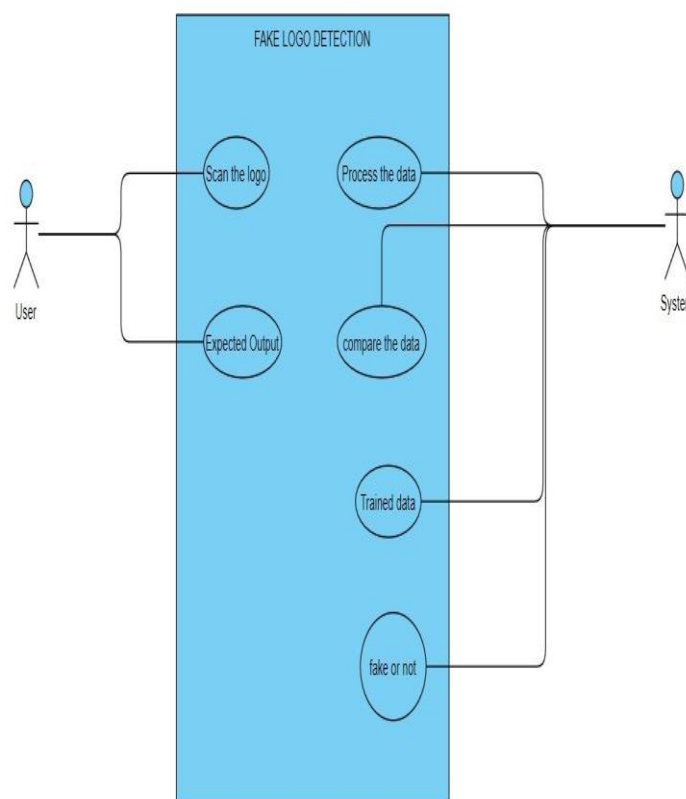


Fig 3.4a Use Case Diagram

3.5 CONCLUSION

In this chapter, the system is divided into various modules for better understanding and the working of several aspects of the application. Also, the UML diagrams are drawn up for the proposed system to highlight the interactions between the modules and to analyze the efficiency of their capabilities. As a result, the proposed Augmented Reality app is secure and can streamline effective and efficient work for the Customers.

4. SYSTEM IMPLEMENTATION

4.1 INTRODUCTION

For the implementation of the proposed system, the Python has been chosen for several reasons. First, it provides all the necessary tools to create interactive environments in a relatively simple way. Second, it has a complete and well documented. And finally, it can run on different platforms such as PCs, consoles, mobile devices or web.

4.2 HARDWARE AND SOFTWARE SPECIFICATIONS

4.2.1 HARDWARE REQUIREMENTS

Processor: Intel i5/i7/i9 (8th Gen or above)

Download speed of 10 Mbps or greater

RAM: 8 GB (or above)

Hard disk: 20GB (or above)

4.2.2 SOFTWARE TOOLS REQUIRED

Python 3.7 or above

Tensor flow OpenCV

SYSTEM ENVIRONMENT FOR PYTHON

PYTHON

Python is a **high-level, interpreted, interactive and object-oriented scripting language**. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **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 is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

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.

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 down sides 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 ever used to implement smart phone-based applications. One such application is called Carbonnelle.

There as on 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. Under developed Database Access Layers

Compared to more widely used technologies like JDBC (Java Data Base Connectivity) and ODBC (Open Data Base 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.

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 white space. The object-oriented approach and language constructs 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. 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 Python3.

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 Python3. [Download the Python Cheat sheet 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

Step1: 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.

Step2: Click on the Download Tab.



Step3: 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

Step4: Scroll down the page until you find the Files option.

Step5: Here you see a different version of python along with the operating system.

Files

Version	Operating System	Description	MD5 Sum	File Size	PGP
Gzipped source tarball	Source release		68111671e5b2db4ae77b9ab01b7079be	13017643	PGP
XZ compressed source tarball	Source release		d33e4a8e6097051c3eca45ee3604803	17131432	PGP
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daf1a4c2c8a0ce0e6	34898416	PGP
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773bf5ea938b241f	28882845	PGP
Windows .hug file	Windows		86399573a2e9682ac58cade8b4f7ad2	8131761	PGP
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	980938f828ee0b8a8e2184a6728a2	7504381	PGP
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76d9b0b35c3a583e563400	26881348	PGP
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28c31c6088b673aee951a3b351b4bd2	1362904	PGP
Windows x86 embeddable zip file	Windows		9fab38d198a1d79fda9413574139d8	6741628	PGP
Windows x86 executable installer	Windows		33c882942a5446a3d8451479394789	25663848	PGP
Windows x86 web-based installer	Windows		1b670cfad117d85c30983ea371d87c	1324608	PGP

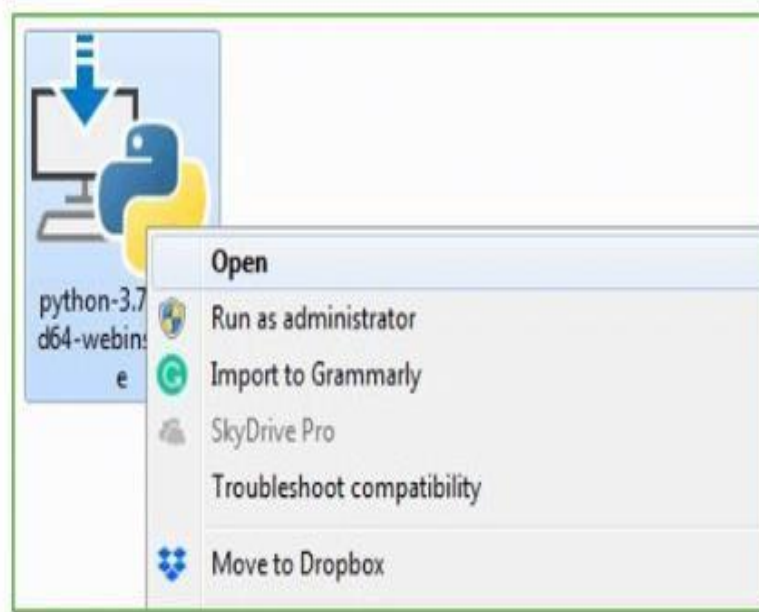
To download Windows 32-bit python, you can select any one from the three options: Windowsx86 embeddable zipfile, Windowsx86 executable installer or Windowsx86 web-based installer.

•To download Windows 64-bit python, you can select any one from the three options: Windowsx86-64 embeddable zipfile, Windowsx86-64 executable installer or Windowsx86-64web-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 these cond 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

Step1: Go to Download and Open the downloaded python version to carry out the installation process.



Step2: Before you click on Install Now, Make sure to put a tick on Add Python3.7 to PATH.



Step3: Click on Install NOW After the installation is successful. Click on Close.



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

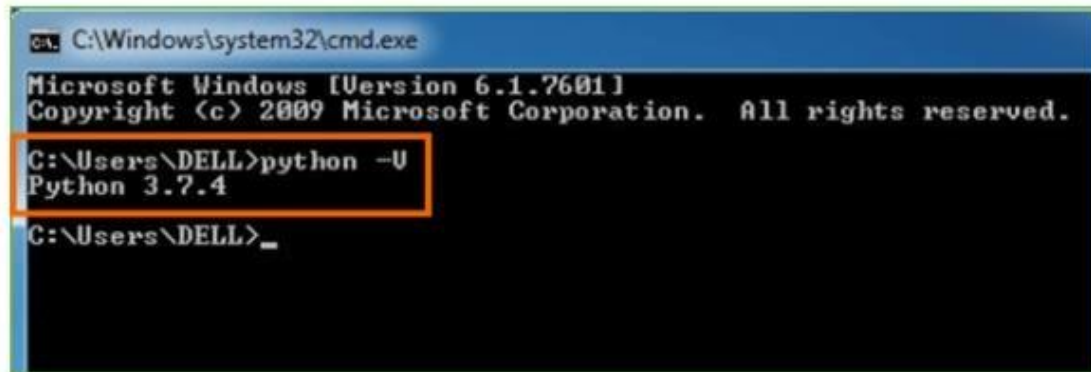
Step1: Click on Start

Step2: In the Windows Run Command, type “cmd”.



Step3: Open the Command prompt option.

Step4: Let us test whether the python is correctly installed. Type **python -V** and press Enter.



```
CA. 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>_
```

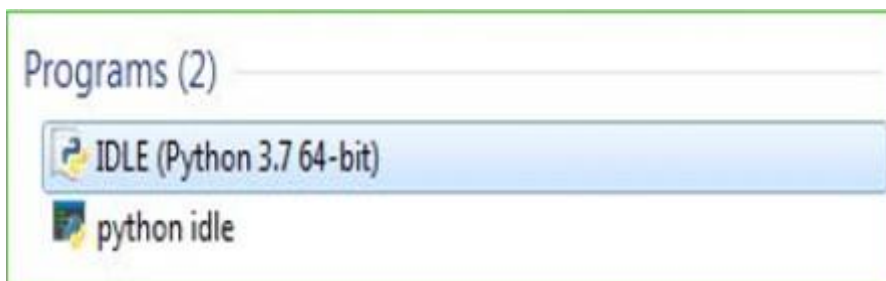
Step5: You will get the answer as 3.7.4

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

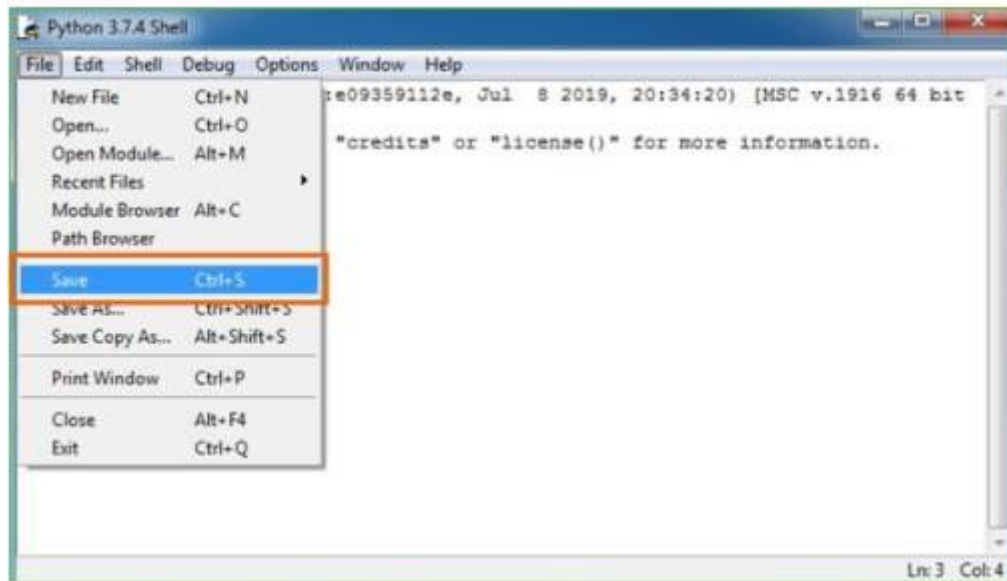
Step1: Click on Start

Step2: In the Windows Run command, type “python idle”.



Step3: Click on IDLE (Python3.764-bit) and launch the program

Step4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



Step5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step6: Now for e.g. **enterprint**

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

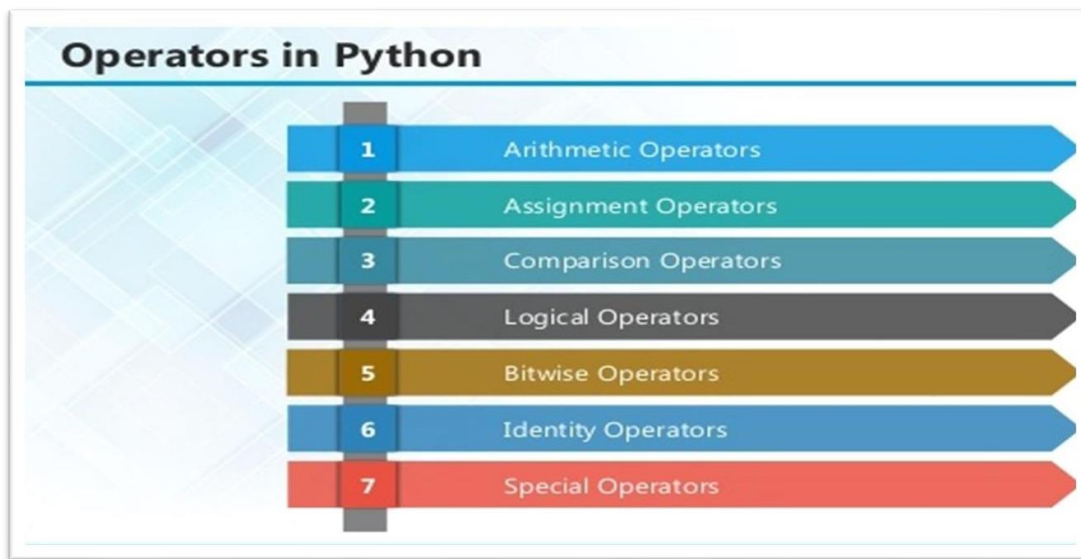
Python's features include:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases :** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Python has a big list of good features:

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.



ARITHMETIC OPERATORS

Operator	Description	Example
+Addition	Adds values on either side of the operator.	a+b= 30
-Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a*b=200
/Division	Divides left hand operand by right hand operand	b /a= 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b%a=0
** Exponent	Performs exponential(power) calculation on operators	a**b=10 to the power20

//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	$9//2=4$ and $9.0//2.0=4.0$, $-11//3=-4$, $-11.0//3=-4.0$
----	--	---

ASSIGNMENT OPERATOR

Operator	Description	Example
=	Assigns values from right side operands to left side operand	$c=a+b$ assigns value of a + b into c
+=AddAND	It adds right operand to the left operand and assign the result to left operand.	$c += a$ is equivalent to $c = c + a$
-=SubtractAND	It subtracts right operand from the left operand and assign the result to left operand	$c -= a$ is equivalent to $c = c - a$
*= Multipl y AND	It multiplies right operand with the left operand and assign the result to left operand	$c *= a$ is equivalent to $c = c * a$
/=DivideAND	It divides left operand with the right operand and assign the result to left operand	$c /= a$ is equivalent to $c = c / a$

IDENTITY OPERATOR

is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	X is y, here is results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	X is not y, here is not results in 1 if id(x) is not equal to id(y)

COMPARISON OPERATOR

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b) (means 0000 1100)
Binary OR	It copies a bit if it exists in either operand.	(a b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)

~Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a)=- 61 (means 1100 0011 in 2's to.
-------------------------	--	---

LOGICAL OPERATOR

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
Or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

Membership Operators

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	X in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not find a variable in the specified sequence and false otherwise.	X not in y, here not in results in a 1 if x is not a member of sequence y.

LIST

The list is a most versatile data type available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example –

```
list1=['physics','chemistry',1997,2000];  
list2=[1,2,3,4,5];  
list3=["a","b","c","d"]
```

Python includes following list methods

SN	Methods with Description
1	<u>list.append (obj)</u> Appends object obj to list
2	<u>list.count (obj)</u> Returns count of how many times obj occurs in list
3	<u>list. Extend (seq)</u> Appends the contents of seq to list
4	<u>list.index (obj)</u> Returns the lowest index in list that obj appears
5	<u>list.insert (index,obj)</u> Inserts object obj in to list at off set index
6	<u>list.pop(obj=list[-1])</u> Removes and returns last object or obj from list

7	<u>list.remove(obj)</u> Removes object obj from list
8	<u>list.reverse()</u> Reverses objects of list in place
9	<u>list.sort([func])</u> Sorts objects of list, use compare function if given

TUPLES

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally we can put these comma-separated values between parentheses also. For example –

```
tup1=('physics','chemistry',1997,2000);
```

Built-in Tuple Functions

S.NO.	Function with Description
1	<u>cmp(tuple1,tuple2)</u> : Compares elements of both tuples.
2	<u>len(tuple)</u> : Gives the total length of the tuple.
3	<u>max(tuple)</u> : Returns item from the tuple with max value.
4	<u>min(tuple)</u> :Returns item from the tuple with m in value.
5	<u>tuple(seq)</u> :Converts a list into tuple.

4.3 TECHNOLOGIES USED

4.3.1 OPENCV

OpenCV is the open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, logos, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To identify image pattern and its various features we use vector space and perform mathematical operations on these features. When OpenCV was designed the main focus was real-time applications for computational efficiency.

4.3.1.1 OpenCV Functionality

1. Image/video I/O, processing, display
2. Object/feature detection
3. Geometry-based monocular or stereo computer vision
4. Computational photography
5. Machine learning & clustering
6. CUD Acceleration

4.3.1.2 Applications of OpenCV

1. Face recognition
2. Automated inspection and surveillance
3. Number of people count (foot traffic in a mall, etc)
4. Vehicle counting on highways along with their speeds
5. Interactive art installations
6. Anomaly (defect) detection in the manufacturing process (the odd defective products)
7. Street view image stitching
8. Video/image search and retrieval
9. Robot and driver-less car navigation and control
10. Object recognition
11. Medical image analysis
12. Movies–3D structure from motion
13. TV Channels advertisement recognition

4.3.2 TENSORFLOW

Tensorflow is an open-source library for numerical computation and large-scale machine learning that ease Google Brain TensorFlow, the process of acquiring data, training models, serving predictions and refining future results. TensorFlow is at present the most popular software library. There are several real-world applications of deep learning that makes TensorFlow popular. Being an Open-Source library for deep learning and machine learning, TensorFlow finds a role to play in text-based applications, image recognition, voice search and many more. Deep Face, Facebook's image recognition system, uses TensorFlow for image recognition. It is used by Apple's Siri for voice recognition. Every Google app that you use has made good use of TensorFlow to make your experience better. The TensorFlow Object Detection API is an open-source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models.

There are already pre-trained models in their framework which are referred to as Model Zoo.

It includes a collection of pre-trained models trained on various datasets such as the

- COCO (Common Objects in Context) dataset
- The KITTI dataset
- The Open Images Dataset.

As you may see below there are various models available what is different in these models. These various models have different architecture and thus provide different accuracies but there is a trade-off between speed of execution and the accuracy in placing bounding boxes. Tensorflow bundles together Machine Learning and Deep Learning models and algorithms. It uses Python as a convenient front-end and runs it efficiently in optimized C++. TensorFlow is at present the most popular software library. There are several real-world applications of deep learning that makes TensorFlow popular. Being an Open-Source library for deep learning and machine learning, TensorFlow finds a role to play in text-based applications, image recognition, voice search and many more. DeepFace application Facebook's image recognition system, uses TensorFlow for image recognition. It is used by Apple's Siri for voice recognition. Every Google app that you use has made good use of TensorFlow to make your experience better.

4.3.3 YOLO

The YOLO framework (You Only Look Once) on the other hand, deals with object detection in a different way. It takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes. The biggest advantage of using YOLO is its superb speed; it's incredibly fast and can process 45 frames per second. YOLO also understands generalized object representation.

This is one of the best algorithms for object detection and has shown a comparatively similar performance to the R-CNN algorithms. Compared to the approach taken by object detection algorithms before YOLO, which repurpose classifiers to perform detection, YOLO proposes the use of an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. Following a fundamentally different approach to object detection, YOLO achieves state-of-the-art results beating other real-time object detection algorithms by a large margin. In addition to increased accuracy in predictions and a better Intersection over Union in bounding boxes (compared to real-time object detectors), YOLO has the inherent advantage of speed. While algorithms like Faster RCNN work by detecting possible regions of interest using the Region Proposal Network and then perform recognition on those regions separately, YOLO performs all of its predictions with the help of a single fully connected layer. Methods that use Region Proposal Networks thus end up performing multiple iterations for the same image, while YOLO gets away with a single iteration. The YOLO algorithm works by dividing the image into N grids, each having an equal dimensional region of $S \times S$. Each of these N grids is responsible for the detection.

SAMPLE LOGO DATASETS NAME:

- Samsung
- Pepsi
- Lays
- Mars
- Oreo
- Heinz
- Marvel
- PlayStation
- Chevrolet
- Burger King
- Hp
- Fila
- Microsoft
- Chrome
- NASA
- Reebok
- Oral b
- Cowbell
- Peak milk
- Twitter
- Google
- Adidas
- Android
- Nutella
- Puma
- Pringles
- Sprite
- Tesla
- Netflix
- Zara

5. TESTING

5.1 INTRODUCTION

Testing is performed to identify errors. It is an integral part of the entire development and maintenance process. The goal of the testing during this phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example, the design must not have any logic faults the design is detected before coding commences, otherwise, the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as a walkthrough.

5.2 LISTOFMODULES

5.2.1 TRAININGMODULE:

One of the best ways to improve deep learning performance is to improve the data. Custom Dataset created with the help of sample images which then augmented using data augmentation techniques to add more data to the dataset. In the proposed system the logo dataset consists of two classes Positives and Negatives.

The positive dataset consists of the whole set of images of the original brands. Only original brands logos are present in the positives. There are multiple images being collected for each and every brand for the sake of getting the accurate results. The intensities of images taken are gray scale and normal rgb colored images.

The Negative dataset consists of the whole set of images of the original brands. Only Fake brand logos are present in the Negatives .There are multiple images being collected for each and every brand for the sake of getting the accurate results. The intensities of images taken are grayscale and normal rgb colored images.

The data training can be carried out on the various training techniques and the training time .the epoch time and the way of grouping the batches is very much important while training the dataset.

An epoch is a term indicates the number of passes of the entire training dataset the machine learning algorithm has completed. The data can be grouped into multiple batches like 8, 16, 32, 64 etc. The no of batches is very much related to the epoch time. If the number of batches are high the data training will be fast and it consumes less epoch time. These were the processes that happen while training the dataset.

DATASET COLLECTION:

Gather a diverse dataset containing images of both genuine and fake logos. This dataset should cover a wide range of logo variations, lighting conditions, backgrounds, and potential sources of forgery.

DATA PREPROCESSING:

Perform preprocessing steps on the collected dataset to enhance the quality and consistency of the images. This may include resizing the images, normalizing pixel values, and removing noise or artifacts.

FEATURE EXTRACTION:

Extract relevant features from the logo images that can differentiate between genuine and fake logos. Commonly used features include texture, shape, color

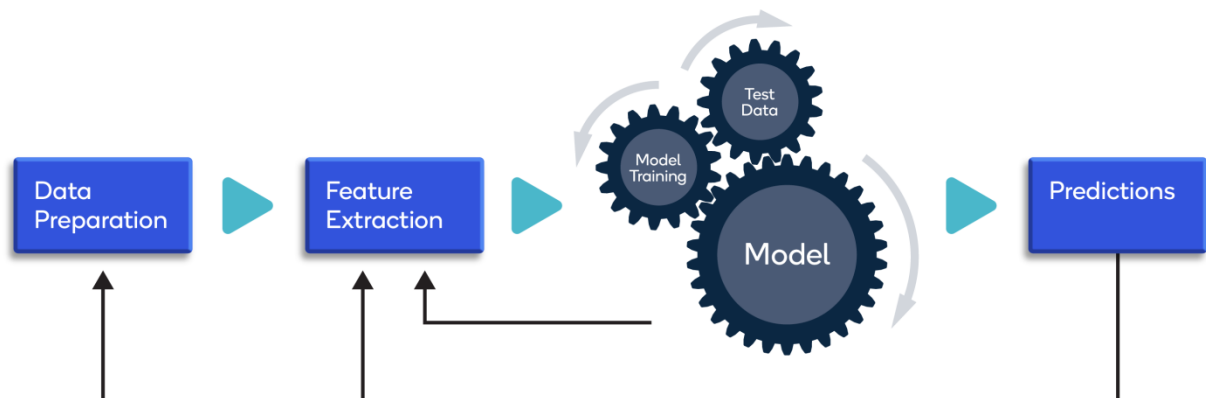




Fig 5.2.1a User Interface

```

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
[INFO] training network...
Epoch 1/30
82/82 [=====] - 3s 36ms/step - loss: 1.3718 - accuracy: 0.9640 - val_loss: 0.5692 - val_accuracy: 0.9630
Epoch 2/30
82/82 [=====] - 3s 33ms/step - loss: 1.3986 - accuracy: 0.8760 - val_loss: 0.5656 - val_accuracy: 0.9269
Epoch 3/30
82/82 [=====] - 2s 29ms/step - loss: 1.2917 - accuracy: 0.8122 - val_loss: 0.4466 - val_accuracy: 0.8299
Epoch 4/30
82/82 [=====] - 2s 29ms/step - loss: 1.2122 - accuracy: 0.6934 - val_loss: 0.5321 - val_accuracy: 0.6651
Epoch 5/30
59/82 [=====>.....] - ETA: 0s - loss: 1.0743 - accuracy: 0.7113

```

Fig5.2.1b Training Model

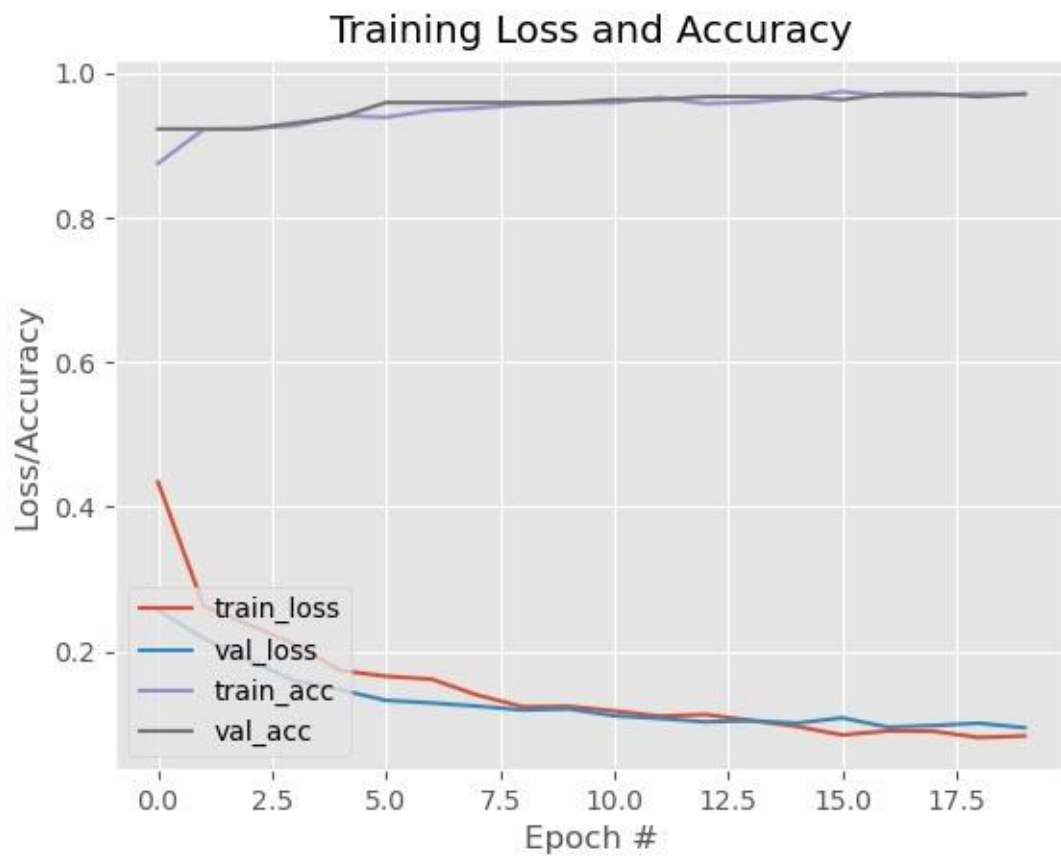


Fig 5.2.1c Training Graph

After the completion of the training module the detection takes place. In this module the main aim is to detect the region of the logo. Once the training is completed the model file is generated which is primarily used to compare the logo detected in real-time with the logo in the dataset. The primary component required for the detection is the camera .when the camera is turned on the logo is shown in front of the camera and it starts detecting it and it analyze the results about the logo. These were the processes that take place in the detection module of the fake logo identification system.

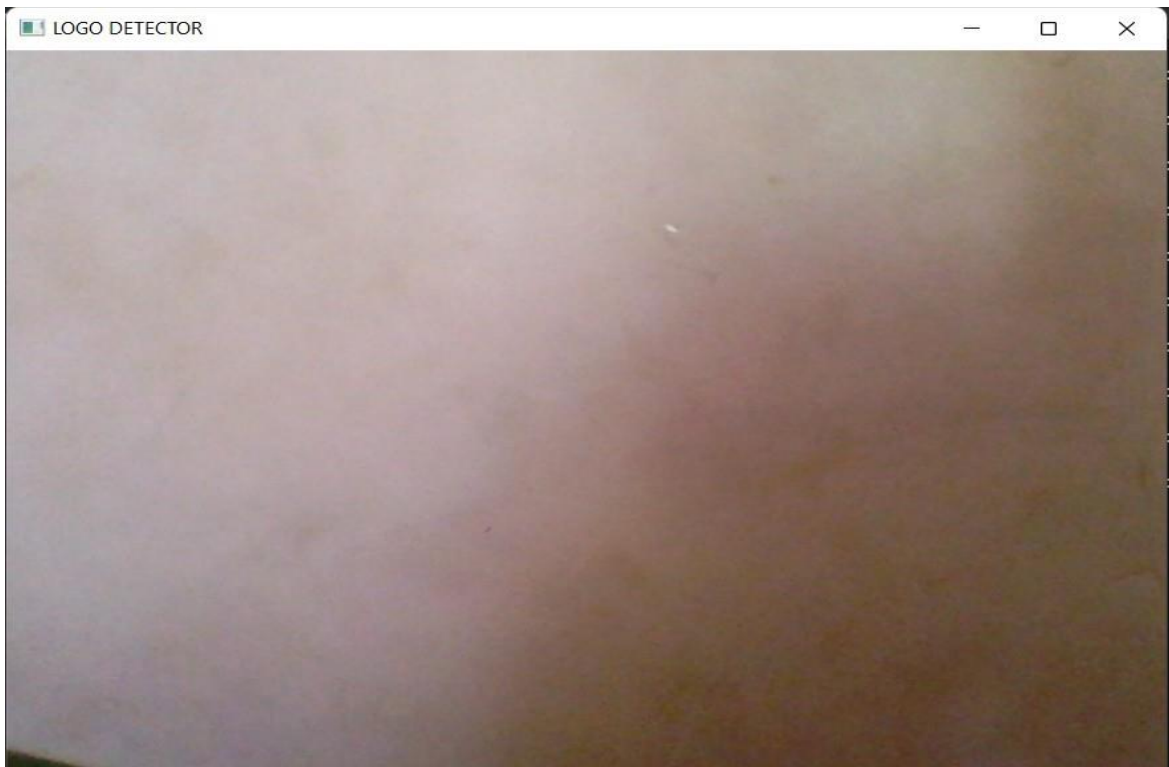


Fig 5.2.2a Detection Camera

5.2.3 THE OUTPUT MODULE

This is the final phase in which the output is shown to the user after analyzing all the factors. The output is shown in a bounding box labeled as “Original Brand logo” And “Fake Brand Logo”. So in this phase the output user got will be useful for him to buy the original product and prevent themselves from buying it. The system differentiates the original and fake ones by comparing with the elements in the dataset.

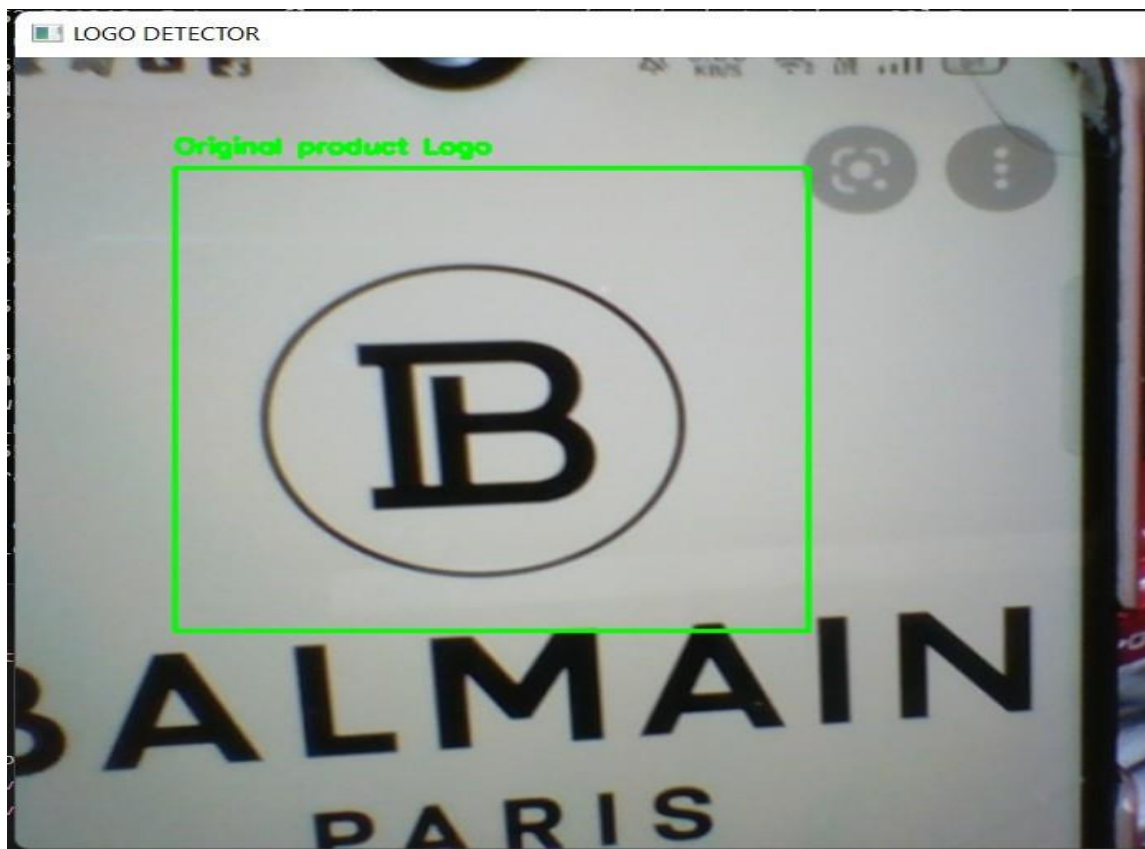
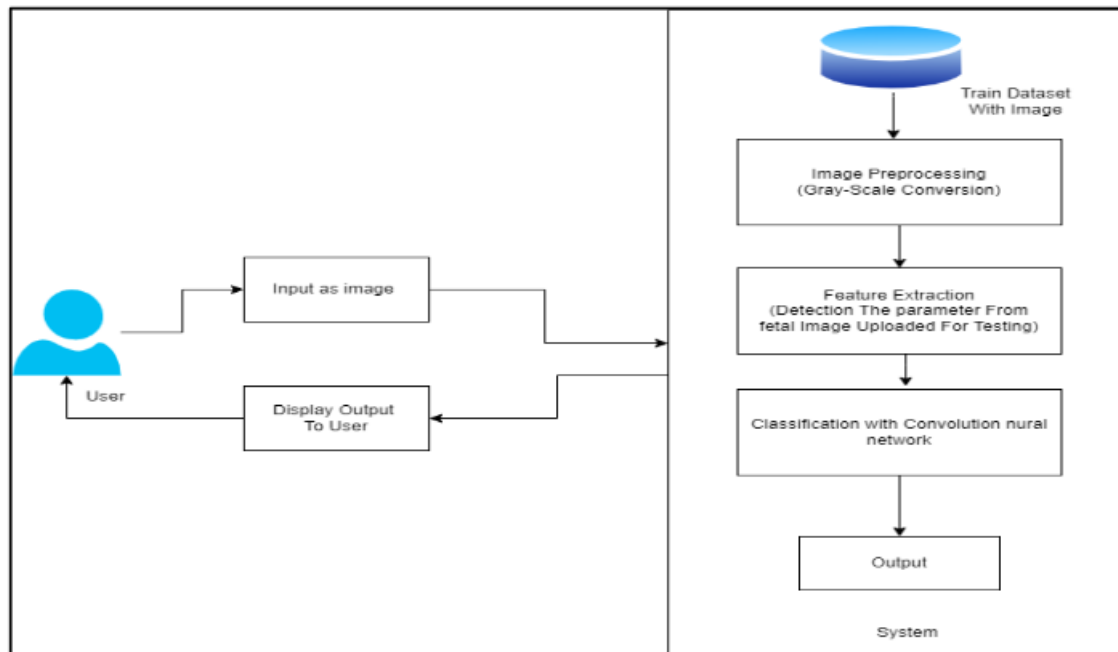


Fig 5.2.3a Detecting Original Balmain Logo



SYSTEM ARCHITECTURE

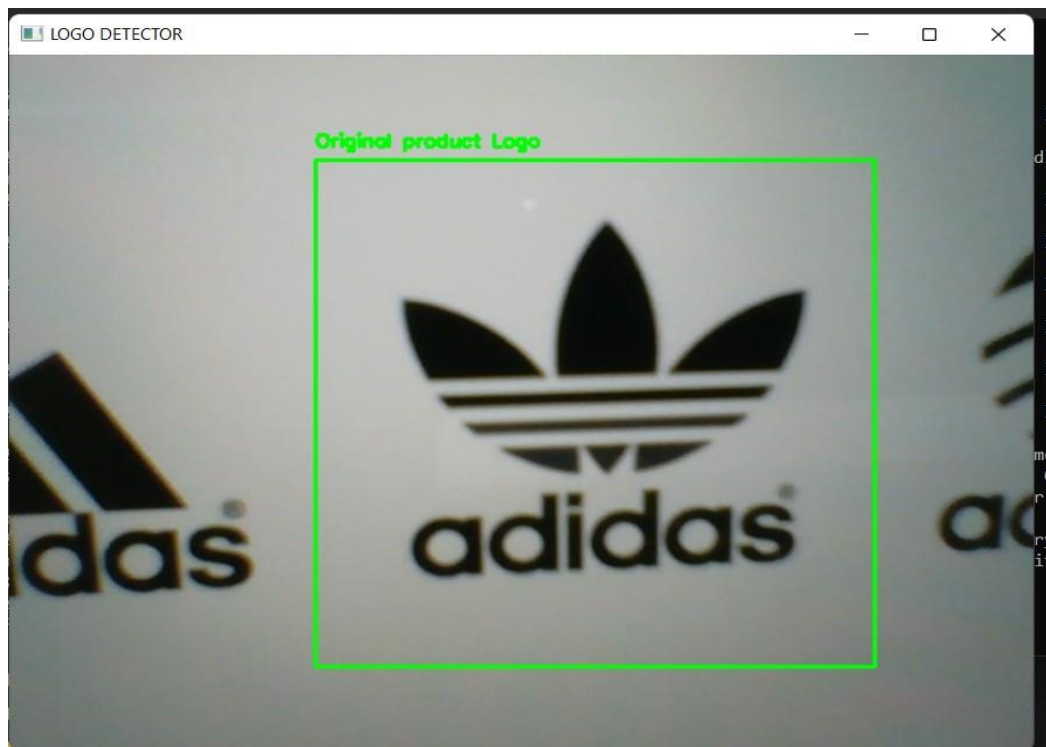


Fig5.2.3b Detecting Original Adidas Logo

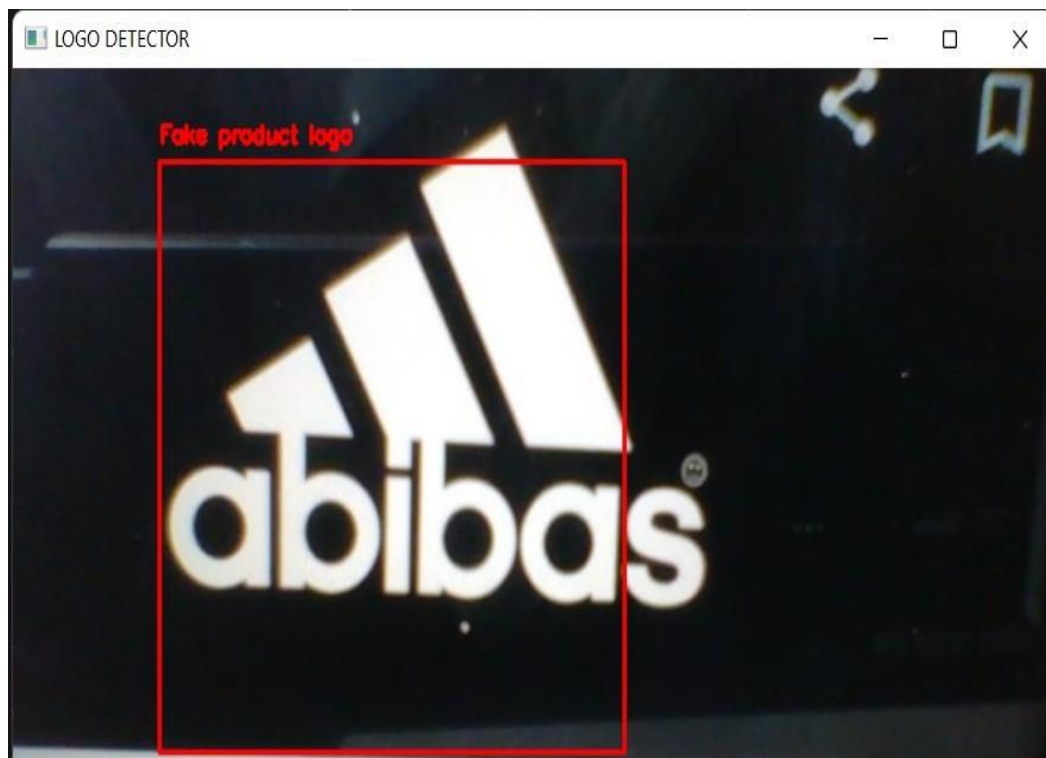


Fig 5.2.3c Detecting Fake Adidas Logo

- User

- Upload Image

The user can click and upload images in .jpg or .png format of the logo they want to detect.

- Logo Detection

Once the logo is uploaded, the system will analyze the image based on parameters such as dimensions, color, text, etc.

After examining these parameters, the system generates a score of less than 90% means that the logo is fake.

6. CONCLUSION

6.1 CONCLUSION

As Logos are the common mode of identification of the original product, it is very important to find the originality of the logo of the product while purchasing. But road maintenance is a very critical task and includes numerous manpower. In This Project work we presented a new “Fake Logo Detection System” Which is definitely the need of an hour for the people and the brand owners. This System provides its own way to find the logo and display in a labeled structure using the yolo algorithm that is used in many real time detection systems. Yolo Algorithm makes the system more efficient than the other existing system. Finally, this System Creates great impact on the market due to its importance to the users and the product owners.

6.1 FUTURE ENHANCEMENTS

The proposed fake Logo Detection System aims to help consumers distinguish fake from the original product. Using this system, a consumer can verify whether a product real or forged. This System can also be helpful for brands struggling to fight against forged products. In future it can be developed in a way users can complain the original brand about the fraudulent activities, so that the brand owners can take sufficient action to reduce the fraudulent activities and prevent the people from buying the fake products and also they can prevent themselves from damaging their credibility.

7. APPENDIX

7.1 APPENDIX 1

```
fromsklearn.preprocessing import LabelEncoder from
sklearn.model_selection import train_test_split from
sklearn.metrics import classification_report
fromkeras.preprocessing.imageimportimg_to_array from
keras.utils import np_utils
fromlenet.nn.convimportLeNet from
imutils import paths
importimutils

importmatplotlib.pyplotasplt import
numpy as np
importargparse
import cv2
import os

ap = argparse.ArgumentParser() ap.add_argument('-d','-
-dataset',required=True,
    help='path to the input dataset of Logos')
ap.add_argument('-m','--model',required=True,
    help='pathtooutputmodel')
args = vars(ap.parse_args()) data
= []
labels=[]
forimagePathinsorted(list(paths.list_images(args['dataset']))): image = cv2.imread(imagePath)
image=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY) image = imutils.resize(image, width=28)
#28 x 28 x 1 image = img_to_array(image)

data.append(image)
label=imagePath.split(os.path.sep)[-3]

label='OriginalProductLogo'iflabel=='positives'else'FakeProductLogo' labels.append(label)
data=np.array(data,dtype='float')/255.0 labels =
np.array(labels)
le=LabelEncoder().fit(labels)

labels=np_utils.to_categorical(le.transform(labels),2) classTotals =
labels.sum(axis=0)
classWeight=dict()
```

```

for i in range(0, len(classTotals)):

classWeight[i]=classTotals.max()/classTotals[i]

(train,test,train,test)=train_test_split(data,labels,test_size=0.20,stratify=labels, random_state=42)
print('[INFO]compilingmodel...')

model = LeNet.build(width=28, height=28, depth=1, classes=2)
model.compile(loss=['binary_crossentropy'],optimizer='adam',metrics=['accuracy'])

print('[INFO] training network...')
H=model.fit(trainX,trainY,validation_data=(testX,testY),class_weight=classWeight, batch_size=64,
epochs=30, verbose=1)
print('[INFO]evaluatingnetwork...')
predictions = model.predict(testX, batch_size=64)
print(classification_report(testY.argmax(axis=1),predictions.argmax(axis=1),
target_names=le.classes_))
print('[INFO]serializingnetwork') model.save(args['model'])

plt.style.use('ggplot') plt.figure()
plt.plot(np.arange(0, 30), H.history['loss'], label='train_loss') plt.plot(np.arange(0, 30),
H.history['val_loss'], label='val_loss') plt.plot(np.arange(0, 30), H.history['accuracy'],
label='accuracy') plt.plot(np.arange(0,30),H.history['val_accuracy'],label='val_accuracy')
plt.title('Training Loss and Accuracy')
plt.xlabel('Epoch #')
plt.ylabel('Loss/Accuracy')plt.legend()
plt.show()

```

7.2 APPENDIX 2

```
from tensorflow.keras.preprocessing.image import img_to_array from
tensorflow.keras.models import load_model
import numpy as np
import imutils
import argparse
import cv2

ap = argparse.ArgumentParser() ap.add_argument('-c', '--
cascade', required=True,
help='path to where the logocascade resides')
ap.add_argument('-m', '--model', required=True,
help='path to the pre-trained logodetectorCNN')
ap.add_argument('-v', '--video',
help='path to the (optional) video file') args =
vars(ap.parse_args())
detector = cv2.CascadeClassifier
('C:\\FAKELOGODETECTOR\\FAKELOGO DETECTOR\\cascade.xml')
model = load_model(args['model']) if not
args.get('video', False):
print('[INFO] starting video capture...') camera =
cv2.VideoCapture(0)
else:
    camera = cv2.VideoCapture(args['video']) while
True:
    (grabbed, frame) = camera.read()
    if args.get('video') and not grabbed: break
    frame = imutils.resize(frame, width=700)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) frameClone =
    frame.copy()
    rectangle = detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=25, minSize=(200, 200),
    flags=cv2.CASCADE_SCALE_IMAGE)
    for (fX, fY, fW, fH) in rectangle:
        roii = gray[fY:fY+fH, fX:fX+fW] roii =
        cv2.resize(roii, (28, 28))
        roii = roii.astype('float')/255.0 roii =
        img_to_array(roii)
        roii = np.expand_dims(roii, axis=0) (negatives,
```

```

positives)=model.predict(roi)[0]
label='OriginalproductLogo'ifpositives>negativeselse"Fakeproductlogo" if label == 'Original
product Logo':
cv2.putText(frameClone, label,(fX,fY-10),cv2.FONT_HERSHEY_SIMPLE, 0.45, (0, 255, 0), 2)
cv2.rectangle(frameClone, (fX,fY),(fX+fW,fY+fH),(0,255,0),2) else:
cv2.putText(frameClone, label,(fX,fY-10),cv2.FONT_HERSHEY_SIMPLE, 0.45, (0, 0, 255), 2)
cv2.rectangle(frameClone, (fX,fY),(fX+fW,fY+fH),(0,0,255),2) cv2.imshow('LOGO DETECTOR',
frameClone)
ifcv2.waitKey(1)&0xFF==ord('q'): break
camera.release()
cv2.destroyAllWindows()

```


7.3 APPENDIX 3

```
fromtkinterimport* import os
fromdatetime import datetime; root=Tk()
root.configure(background="white") def
function1():
    os.system("pytrain_model.py-dLOGOs-mmodel.h5") def function2():
    os.system("pydetect_Logo.py-ccascade.xml-mmodel.h5") def function6():
root.destroy()

root.title("FAKE LOGO DETECTION SYSTEM")

Label(root,text="FAKELOGODETECTION
SYSTEM",font=("Blackletter",20),fg="white",bg="black",height=2).grid(row
=0,rowspan=2,columnspan=2,sticky=N+E+W+S, padx=5,pady=5) Button(root,
text="Trainlogo", font=("times new
roman",20),bg="#7F7F7F",fg='white',command=function1).grid(row=3,colu
mnspan=2,sticky=W+E+N+S, padx=5,pady=5)
Button(root, text="Detect
LOGO",font=("Decorative",20),bg="#3F3F3F",fg='white',command=function
2).grid(row=4,columnspan=2,sticky=N+E+W+S, padx=5,pady=5) Button(root,text="Exit",
font=('times new roman',20),bg="black",fg="white",command=function6).grid(row=9,columns
pan=2,sticky=N+E+W+S, padx=5,pady=5)
```

8. BIBLIOGRAPHY

- [1] Hang Su ,Xiatian Zhu andShaogang Gong . Deep Learning Logo Detection withData Expansion by SynthesisingContext . 2018.
- [2] SimoneBianco,MarcoBuzzelli,DavideMazziniandRaimondoSchettini.Deep Learning for Logo Recognition . 2017.
- [3] AayushGarg, Thilo Will, William Darling, WilliRichert and Clemens Marschner . Scalable Object Detection for Stylized Objects 2017.
HangSu, XiatianZhu, ShaogangGong.Open LogoDetectionChallenge.2018.
- [4] IstvanFehervari andSrikarAppalaraju .Scalable LogoRecognition using Proxies .2018.
- [5] JunjunHu,YanhaoZhu,BoZhao, JiexinZheng,ChenxuZhao,XiangyuZhu, Kangle Wu ,Darun Tang . Makeup216: Logo Recognition with Adversarial Attention Representations.
- [6] QitingYe, ZhaoLuo, XiabingXiao, ShimingGe.GeLoGO: Detecting TV logos from Web-Scale Videos. 2017.
- [7] Hendrick, Chih-Min , Wang,Aripriharta, Ciou-Guo , Jhe, Ping-Cheng , Tsu, Gwo-Jia, Jong . The Halal Logo Classification by Using NVIDIA DIGITS . 2019.
- [8] Umesh D. Dixit andM. S. Shirdhonkar . Automatic Logo Detection and Extraction using Singular Value Decomposition . 2016.
GuangyuZhuandDavidDoermann.AutomaticDocumentLogoDetection.
- [9] Nabin Sharma, RanjuMandal, Rabi Sharma, Umapada Pal and Michael Blumenstein . Signature and Logo Detection using Deep CNN for Document Image Retrieval.
- [10] Rahul kumar , Taro, Tomohiro, Takahiro, Yutaro, Xiang,and Yen – Wei . LogoNet: Layer-Aggregated Attention CenterNet for Logo Detection. 2021.
- [11] Kuan-HsienLiu1 , Member, IEEE, Fei Wang2 , Tsung-Jung Liu2 , Member, IEEE, and Guan-Hong Chen . Clothing Brand Prediction via Logo Recognition . 2020.
- [12] Mansi Mahendru1, Sanjay Kumar Dubey2.Real Time Object Detection with Audio Feedback using Yolo vs. Yolo_v3 . 2021.

- [13] Nhat-DuyNguyen^{1,2},ThuaNguyen^{1,2},TienDo^{1,2},ThanhDucNgo^{1,2},andDuy-DinhLe¹.U15-Logos: Unconstrained Logo Dataset with Evaluation by Deep learning Methods .
- [14] AndreyKuznetsov, AndreySavchenko.Sport Teams Logo Detection Based On Deep Local Features . 2019.
- [15] Md. BaharUllah . CPU Based YOLO: A Real Time Object Detection Algorithm .2020.
- Linghua Zhou, Weidong Min, Member, IEEE, Deyu Lin, Member, IEEE, Qing Han, Ruikang Liu.Detecting Motion Blurred Vehicle Logo in IoV Using Filter- DeblurGAN and VL-YOLO . 2020.
- Shuo Yang , Chunjuan Bo, Junxing Zhang, PengxiangGao, Yujie Li , and Seichi Serikawa . VLD-45: A Big Dataset for Vehicle Logo Recognition and Detection . 2021.