

IMPLEMENTATION OF DENSE OBJECT DETECTION IN SELF DRIVING APPLICATIONS

Submitted in partial fulfillment of the requirements for
the award of

Bachelor of Technology degree in Information Technology
by

MUKKAPATI V D N SAI BABA

(Reg. No. 37110463)

G.S.TRIVIKRAM NAIDU (Reg. No. 37110788)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC JEPPIAAR
NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600

119

MARCH – 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" Grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119.

Phone: 044 - 2450 3150 / 51 / 52 / 54 / 55 Fax: 044 - 2450 2344

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **MUKKAPATI V D N SAI BABA** (Reg. No. 37110463) and **G.S.TRIVIKRAM NAIDU** (Reg. No. 37110788) who carried out the project entitled "**IMPLEMENTATION OF DENSE OBJECT DETECTION IN SELF DRIVING APPLICATIONS**" under my supervision from **NOVEMBER 2020** to **March 2021**.

Internal Guide

Dr.Y.Bevis Jinila, M.E., Ph.D.,

Head of the Department

Dr. S. Vigneswari, M.E., Ph.D.,

Submitted for Viva voice Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **MUKKAPATI V D N SAI BABA** hereby declare that the Project Report entitled **“IMPLEMENTATION OF DENSE OBJECT DETECTION IN SELF DRIVING APPLICATIONS”** is done by me under the guidance of **Dr.Y.BEVISH JINILA, M.E.,Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: Chennai

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

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

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D.,** and **Dr. L. Lakshmanan, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.Y.Bevis Jinila, M.E., Ph.D., Associate Professor** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

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

ABSTRACT

The focal loss for dense object detection to date are based on two stages of approval. Except if there are numerous tests on facial sketch mixes, the past measurable data can't resolve certain non-facial highlights. In the assortment of data like glasses, beautifiers, and hair, the information is dismissed. In this project, we embraced a facial acknowledgment structure, which relies upon outward appearances, cosmetics, and eye to eye direction. Then, utilizing short photographs to see the pictures in the data set is in fact beneficial. The police can discover covered up suspects, yet in addition encourage seeing, and the head can likewise draw dependent on the associated pictures and make drawings that are viewed as viable, which will be gainful. The proposed structure is separated into two phases, specifically: a mix of produced drawings and draft endorsement. The strategy for counterfeit work of art craftsmanship relies upon the immediate assurance of the neighbors between the picture and the artistic creation, which is affected by the idea of the immediate position of the region. In the portrayal of Principal component analysis (PCA), the edge picture can be created from the inherent face. Since Eigen's face is enrolled from the readiness community, we can demonstrate that the created face picture can likewise be communicated as an immediate blend of test planning.

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO.
3.1	Water Fall Model	10
3.2	RAD Model	12
3.3	Flow Diagram	15

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO.
	ABSTRACT	iv
	LIST OF FIGURES	v
1	INTRODUCTION	1
2	LITERATURE SURVEY	
	2.1 SURVEY 1	5
	2.2 SURVEY 2	6
	2.3 SURVEY 3	7
	2.4 SURVEY 4	8
3	SYSTEM ANALYSIS AND DESIGN	9
4	EXPERIMENTAL RESULTS	
	4.1 Hardware Requirements	11
	4.2 Software Requirements	11
	4.3 About Python	12
5	Conclusion	29
	References	31

CHAPTER 1

INTRODUCTION

Especially in automotive, need of Dense object detection increase due to Computer vision in vehicles for safety and also the growth of Autonomous vehicles. Here we are going to show how Dense Object Detection Implemented in Automotive field.

1.1 Overview

The proposed system consists of two ultrasonic sensors and a microcontroller equipped with data logging shield. The data were recorded and then analyzed using a computer program written in Java. Actual experiments that have been conducted indicates that this system can give a correct number of vehicles in a given interval of time.

1.2 Scope of the Project

The main contributions of this project therefore are:

- Data Analysis

- Dataset Preprocessing

- Training the Model

- Testing of Dataset

1.3 Domain Overview

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and

hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

Image recognition technology has a great potential of wide adoption in various industries. In fact, it's not a technology of the future, but it's already our present. Such corporations and startups as Tesla, Google, Uber, Adobe Systems etc heavily use image recognition. To prove that the technology marches around the world let's look at the recent statistics. Researchers predict that the global market of image recognition will reach \$38.92 billion by 2021. That's quite a sound figure! So no wonder that more and more so-called imagetech applications that leverage image recognition emerge for various purposes and business verticals. What helps the expansion of image recognition technology today? It's open-source tools that make programming easier, while computing more affordable. Open-source frameworks and libraries today make it possible for companies to benefit from image recognition technology exponentially. For instance, such massive open databases as Pascal VOC and ImageNet give access to millions of tagged images.

In fact, they help companies and image tech startups to develop and improve their own machine learning apps and algorithms. For a real-time image recognition professional

developers also use an open-source, cross-platform library called OpenCV. It's considered the first point of addressing and the most authoritative resource for image recognition professionals. Another great set of libraries for computer vision include OpenNN, VXL and many other. But before jumping into more details on image recognition and processing it should be mentioned that images can be used in different ways. In mobile, web and software development images serve for a multitude of reasons, including: Object recognition Pattern recognition Locating duplicates (exact or partial) Image search by fragments Camera's image processing Improving mobile apps UX Augmented reality Possible applications of images, imagetech and image recognition technology in particular are unlimited. With the advancements in the technology new opportunities for businesses and startups in every niche arise. Thus image analysis is taken to a whole new level thanks to open-source solutions and deep learning tools out there. Image Recognition Image recognition or a computer vision is a technical discipline that deals with searching the ways to automate all the job that a human visual system can do.

TensorFlow by Google, DeepFace by Facebook, Project Oxford by Microsoft are great examples of deep learning image recognition systems. On the other hand, hosted APIs such as Google Cloud Vision, Clarifai, Imagga allow businesses to save some money on the costly computer vision development teams. Advantages of open-source services mentioned are numerous. They conduct image recognition computing in the cloud making your imagetech business operations more efficient and much cheaper. Also in-house developers at your company can integrate their APIs into your apps without problems. Moreover, these open APIs developers can use to even start a new business in the field of image recognition. Image Processing Techniques Generally image processing consists of several stages: image import, analysis, manipulation and image output. There are two methods of image processing: digital and analogue. In

particular, digital image processing and its techniques is what this article is about. Computer algorithms play a crucial role in digital image processing. Developers use multiple algorithms to solve different tasks, including digital image detection, analysis, reconstruction, restoration, image data compression, image enhancement, image estimation and image spectral estimation.

Major techniques of digital image processing are as follows: Image Editing, which basically means altering digital images by means of graphic software tools. Image Restoration, which refers to the estimation of a clean original image out of the corrupt image taken in order to get back the information lost. Independent Component Analysis, which separates a multivariate signal computationally into additive subcomponents. Anisotropic Diffusion, which is often known as Perona-Malik Diffusion, makes it possible to reduce image noise without having to remove important parts of the image. Linear Filtering. It's another digital image processing technique, which refers to processing time-varying input signals and producing output signals that are subject to the constraint of linearity. Neural Networks, which are computational models widely used in machine learning for solving various tasks. Pixelation, which often refers to turning printed images into digitized ones (such as GIF). Principal Components Analysis, a digital image processing technique that can be used for feature extraction. Partial Differential Equations, which also is dealing with effectively de-noising images. Hidden Markov Models, a technique used for image analysis in two dimensions. Wavelets, which stands for a mathematical function that's used in image compression. Self-organizing Maps, a digital image processing technique for classifying images into a number of classes.

CHAPTER 2

LITERATURE SURVEY

Literature Survey 1	
Title	Cooperative Deep Q-Learning With Q-Value Transfer for Multi-Intersection Signal Control
Authors	HONGWEI GE, YUMEI SONG, CHUNGUO WU, JIANKANG REN, AND GUOZHEN TAN
Published Year	Aug 2019
Efficiency	The adaptive control of signallights can optimize the traf_c of regional road network,reduce congestion and carbon dioxide emissions
Drawbacks	Traffic congestion has become a major strategic problem facing the sustainable and harmonious development of cities.
Description	This paper proposes,The problem of adaptive traf_c signal control in the multi-intersection system has attracted the attention of researchers. Among the existing methods, reinforcement learning has shown to be effective. However, the complex intersection features, heterogeneous intersection structures, and dynamic coordination for multiple intersections pose challenges for reinforcement learning-based algorithms. This paper proposes a cooperative deep Q-network with Q-value transfer (QT-CDQN) for adaptive multi-intersection signal control. In QT-CDQN, a multi-intersection traf_c network in a region is modeled as a multi-agent reinforcement learning system. Each agent searches the optimal strategy to control an intersection by a deep Q-network that takes the discrete state encoding of traf_c information as the network inputs. Towork cooperatively, the agent considers the in_uence of the latest actions of its adjacencies in the process of policy learning. Especially, the optimal Q-values of the neighbor agents at the latest time step are transferred to the loss function of the Q-network. Moreover, the strategy of the target network and the mechanism of experience replay are used to improve the stability of the algorithm. The advantages of QT-CDQN lie not only in the effectiveness and scalability for the multi-intersection system but also in the versatility to deal with the heterogeneous intersection structures. The experimental studies under different road structures show that the QT-CDQN is competitive in terms of average queue length, average speed, and average waiting time when compared

	with the state-of-the-art algorithms. Furthermore, the experiments of recurring congestion and occasional congestion validate the adaptability of the QT-CDQN to dynamic traf_c environments.
Literature Survey 2	
Title	Deep Reinforcement Learning for Traffic Light Optimization
Authors	Mustafa Coskun, Abdelkader Baggag, Sanjay Chawla
Published Year	May 2018
Efficiency	It is able to decrease travel time in a micro-simulator environment.
Drawbacks	high variance problem policy and reward difference can be very high
Description	This system proposes, Deep Reinforcement Learning has the potential of practically addressing one of the most pressing problems in road traffic management, namely that of traffic light optimization (TLO). The objective of the TLO problem is to set the timings (phase and duration) of traffic lights in order to minimize the overall travel time of the vehicles that traverse the road network. In this paper, we introduce a new reward function that is able to decrease travel time in a micro-simulator environment. More specifically, our reward function simultaneously takes the traffic flow and traffic delay into account in order to provide a solution to the TLO problem. We use both Deep Q-Learning and Policy Gradient approaches to solve the resulting reinforcement learning problem.

Title	Deep Reinforcement Learning for Traffic Light Optimization
Authors	Mustafa Coskun, Abdelkader Baggag, Sanjay Chawla
Published Year	May 2018
Efficiency	It is able to decrease travel time in a micro-simulator environment.
Drawbacks	high variance problem policy and reward difference can be very high
Description	This system proposes, Deep Reinforcement Learning has the potential of practically addressing one of the most pressing problems in road traffic management, namely that of traffic light optimization (TLO). The objective of the TLO problem is to set the timings (phase and duration) of traffic lights in order to minimize the overall travel time of the vehicles that traverse the road network. In this paper, we introduce a new reward function that is

	able to decrease travel time in a micro-simulator environment. More specifically, our reward function simultaneously takes the traffic flow and traffic delay into account in order to provide a solution to the TLO problem. We use both Deep Q-Learning and Policy Gradient approaches to solve the resulting reinforcement learning problem.
Literature Survey 3	
Title	Self-driving Cars Using CNN and Q-learning
Authors	Syed Owais Ali Chishti, Sana Riaz, Muhammad Bilal Zaib, Mohammad Nauman
Published Year	October 2018
Efficiency	prospective of this technology is clear as it will dramatically change transportation by minimizing traffic jam, increasing efficiency and allowing faster speed
Drawbacks	Autonomous car's software is forced during an unavoidable crash to choose between multiple harmful courses of action, and possibly insufficient adaptation to respond on spoken commands
Description	This paper proposes, DrivingMatter is an experiment carried out to understand the deeper side of an autonomous car. In 1900s, idea was to drive car on Moon from Earth. This was initial motivation which grew from there and now expanding to complex system of roads in the real world. A book-sized Raspberry Pi based autonomous car is built to carry out the experiment on hardware. Software side was accomplished by developing a Python based library for controlling and communicating with car over a network or locally within the car. For environment learning two methodologies are practiced; Supervised learning: Drove the car on an environment/road and collected 3,000+ data-points. Based on this a CNN model was trained which achieved 73% test 89% train accuracy. Reinforcement learning: Car is trained for three different road signs; Stop, No left, and Traffic light using DQN with existing CNN model. These road signs are detected in the environment using OpenCV cascade classifiers
Literature Survey 4	
Title	Deep Q Learning with LSTM for Traffic Light Control
Authors	Chung-Jae Choe, Seungho Baek, Bongyoung Woon, and Seung-Hyun Kong
Published Year	July 2018

Efficiency	TLC, which deals with complex traffic environments, reinforcement learning can effectively reduce traffic congestion
Drawbacks	Since the traffic light is not controlled simultaneously, there is a time lag between the data collection and the TLC operation.
Description	This method proposes, Most Conventional traffic light control (TLC) techniques do not provide enough efficiency to control dynamic traffic situations in real-time. Recently, DQN (Deep Q Network) algorithm is considered for TLC at the intersection because of its optimization technique for complex problems, where key features of the intersection traffic, such as vehicle positions and velocities, are obtained from the intersection by the camera installed at well above the ground. However, the general DQNbased TLC algorithms have failed to utilize the fact that vehicle trajectories are continuous, which can be very useful in sensing real-time traffic. To utilize the continuous vehicle motion for TLC improvement, we propose DRQN-TLC (Deep Recurrent Q Network for TLC) algorithm that is based on LSTM (Long-Short Term Memory) with DQN. The superior performance of the proposed algorithm is demonstrated with the simulation; the proposed algorithm reduces the average traveling time by 23% and the overall vehicle waiting time by 10% when compared with the general DQN-based TLC algorithm.

CHAPTER 3

SYSTEM ANALYSIS AND DESIGN

3.1 Overview

One of the primary choices we face for our project implementations is “Which development methodology should we use?” There are various software development methodologies available in the ICT industry. They are

- Agile Software Development
- Crystal Methods
- Dynamic Systems Development Model (DSDM)
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Joint Application Development (JAD)

- Lean Development (LD)
- Rapid Application Development (RAD)
- Rational Unified Process (RUP)
- Scrum
- Spiral
- Systems Development Life Cycle (SDLC)
- Waterfall (a.k.a. Traditional)

Among all these methodologies, we have decided to compare the two most popular and widely used methodologies:

- Waterfall Model
- RAD Model

3.2 Waterfall Model

It is called as traditional approach. Waterfall is a linear method (sequential model) for any software application development. Here application development is segregated into a sequence of pre-defined phases.

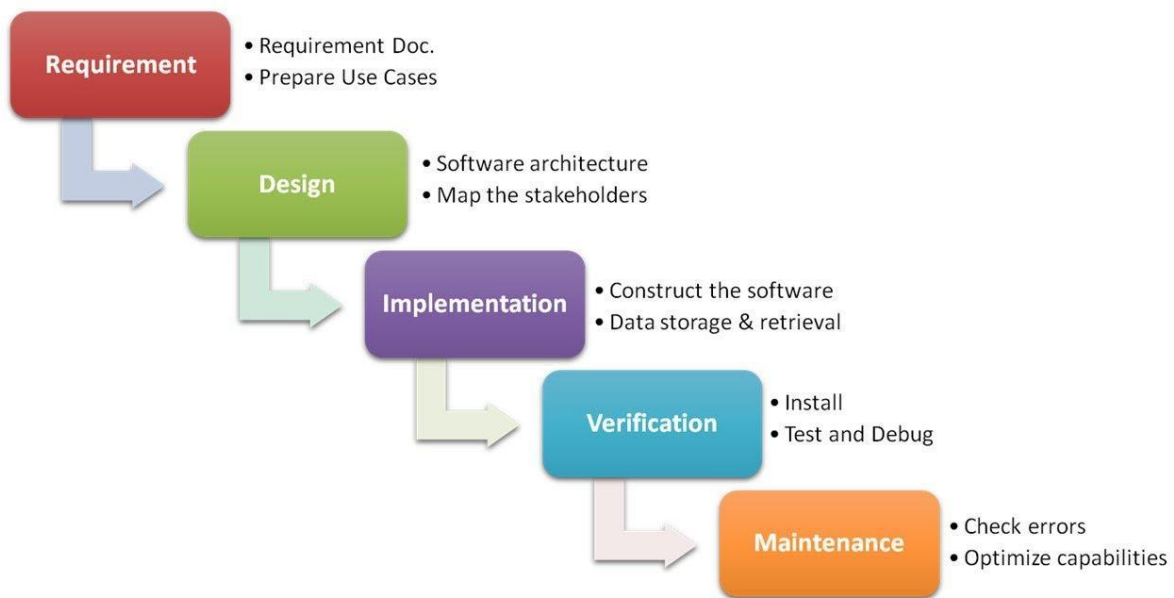


Fig. 3.1 Waterfall Model

1. Requirement gathering and documentation

In this stage, you should gather comprehensive data approximately what this challenge requires. You may gather this data in a diffusion of ways, from interviews to questionnaires to interactive brainstorming. By means of the stop of this section, the task requirements have to be clean, and also you have to have a necessities file that has been allotted on your team.

2. System design

Using the well-known requirements, your team designs the solutions. During this phase, no development will be happening. But the project team starts specification such as programming language or hardware requirements.

3. Implementation

During this phase software development coding will be happening. Web application programmers take data from the previous stage and create a functional product. Web application programmers write source code in small pieces, which are integrated at the end of this phase or the beginning of the next.

4. Testing

Once all coding is done, testing of the product can begin. Testers methodically find and report any problems. If serious issues arise, your project may need to return to phase one for reevaluation.

5. Delivery/deployment

In this phase, the solution is complete, and your project team submits the deliverables to be deployed or released.

6. Maintenance

The final solution has been implemented to the client and is being used. As troubles arises, the project team might also want to create patches and updates might also to deal with them. Again, huge troubles may also necessitate a return to segment one.

4.3 RAD Model

Agile is an iterative, team-based approach to development. This method highlights the speedy delivery of an software application in comprehensive functional components.

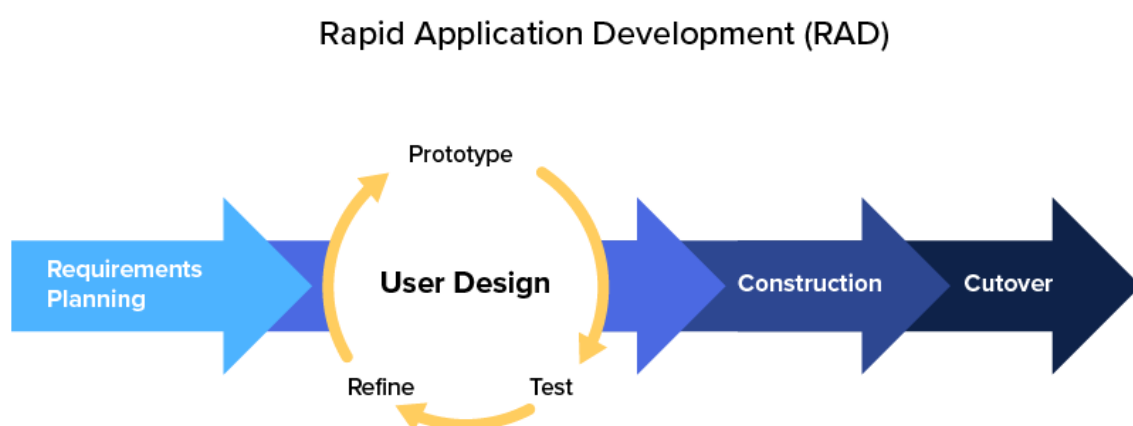


Fig 3.2 RAD Model

Phase 1: Requirements planning

This phase is equivalent to a project scoping meeting. Although the planning phase is condensed compared to other project management methodologies, this is a critical step for the ultimate success of the project.

During this stage, web application programmers, clients (software users), and team members communicate to determine the goals and expectations for the project as well as current and potential issues that would need to be addressed during the build.

A basic breakdown of this stage involves:

- Researching the current problem

- Defining the requirements for the project

- Finalizing the requirements with each stakeholder's approval

It is important that everyone has the opportunity to evaluate the goals and expectations for the project and weigh in. By getting approval from each key stakeholder and web application programmer, teams can avoid miscommunications and costly change orders down the road.

Phase 2: User design

Once the project is scoped out, it's time to jump right into development, building out the user design through various prototype iterations.

This is the meat and potatoes of the RAD methodology—and what sets it apart from other project management strategies. During this phase, clients work hand in hand with web application programmers to ensure their needs are being met at every step in the design process. It's almost like customizable software development where the users can test each prototype of the product, at each stage, to ensure it meets their expectations.

All the bugs and kinks are worked out in an iterative process. The web application programmer designs a prototype, the client (user) tests it, and then they come together to communicate on what worked and what didn't.

This method gives web application programmers the opportunity to tweak the model as they go until they reach a satisfactory design.

Both the software web application programmers and the clients learn from the experience to make sure there is no potential for something to slip through the cracks.

Phase 3: Rapid construction

Phase 3 takes the prototypes and beta systems from the design phase and converts them into the working model.

Because the majority of the problems and changes were addressed during the thorough iterative design phase, web application programmers can construct the final working model more quickly than they could by following a traditional project management approach.

The phase breaks down into several smaller steps:

Preparation for rapid construction

Program and application development

Coding

Unit, integration, and system testing

The software development team of programmers, coders, testers, and web application programmers work together during this stage to make sure everything is working smoothly and that the end result satisfies the client's expectations and objectives.

This third phase is important because the client still gets to give input throughout the process. They can suggest alterations, changes, or even new ideas that can solve problems as they arise.

Phase 4: Cutover

This is the implementation phase where the finished product goes to launch. It includes data conversion, testing, and changeover to the new system, as well as user training.

All final changes are made while the coders and clients continue to look for bugs in the system.

Benefits of RAD methodology

RAD is one of the most successful software development programs available today, with numerous benefits for both software development teams as well as their clients.

Here are just a few advantages:

- RAD lets you break the project down into smaller, more manageable tasks.

- The task-oriented structure allows project managers to optimize their team's efficiency by assigning tasks according to members' specialties and experience.

- Clients get a working product delivered in a shorter time frame.

- Regular communication and constant feedback between team members and stakeholders increases the efficiency of the design and build process.

Architecture Diagram

An architecture diagram is a graphical representation of a set of concepts, that are part of an

architecture, including their principles, elements and components.

What is a diagram? What are the types of diagrams for architecture? The Dragon1 open EA Method makes it very clear: if a diagram does not show a concept, principle or part of a principle, it is NOT an architecture diagram, because it does not show (a part of) the architecture.

There are many kinds of architecture diagrams, like a software architecture diagram, system architecture diagram, application architecture diagram, security architecture diagram, etc.

For system developers, they need system architecture diagrams to understand, clarify, and communicate ideas about the system structure and the user requirements that the system must support. It's a basic framework can be used at the system planning phase helping partners understand the architecture, discuss changes, and communicate intentions clearly

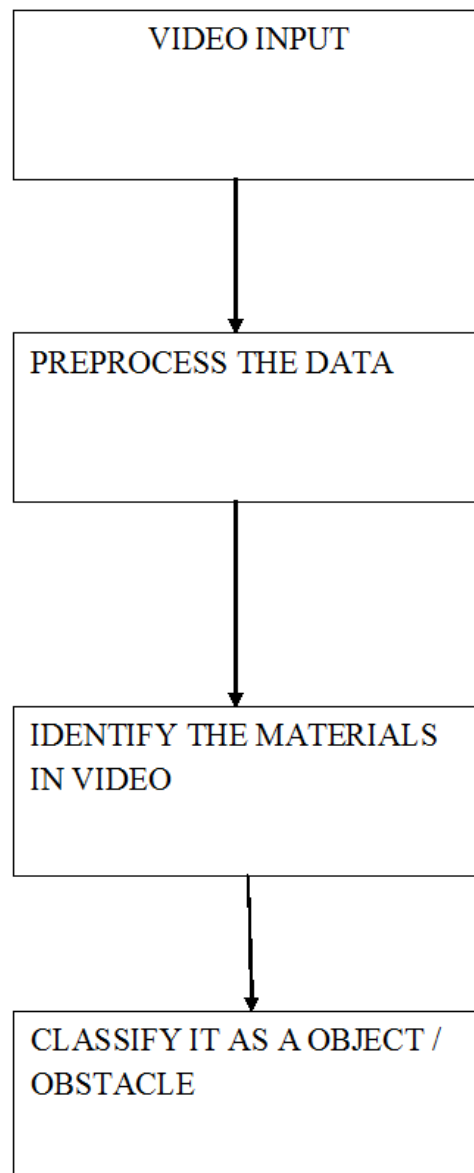


Fig. 3.3 Flow Diagram

EXPERIMENTAL RESULTS

Clearly defined requirements are essential signs on the road that leads to a successful project. They establish a formal agreement between a client and a provider that they are both working to reach the same goal. High-quality, detailed requirements also help mitigate financial risks and keep the project on a schedule. According to the Business Analysis Body of Knowledge definition, requirements are a usable representation of a need.

Creating requirements is a complex task as it includes a set of processes such as elicitation, analysis, specification, validation, and management. In this article, we'll discuss the main types of requirements for software products and provide a number of recommendations for their use.

Hardware Requirements

Hardware	Minimum Requirement
Disk Space	32 GB or more, 10 GB or more for Foundation Edition
Processor	1.4 GHz 64 bit
Memory	512 MB
Display	(800 × 600) Capable video adapter and monitor

Software Requirements

Backend Technologies

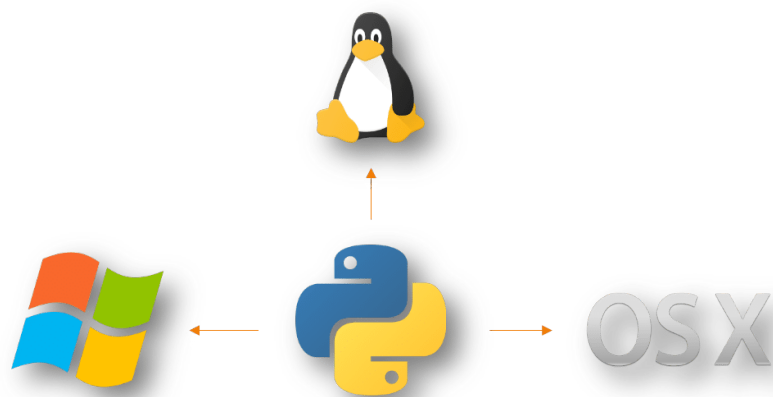
- ✓ Python
- ✓ Numpy
- ✓ Sci-learn
- ✓ Eclipse IDE

Frontend Technologies

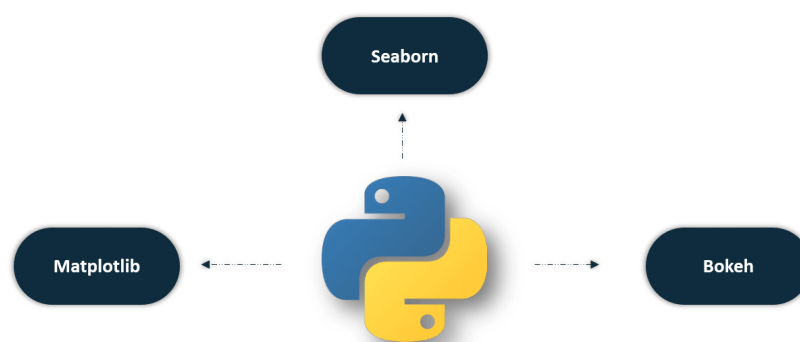
- ✓ Web Technologies
- ✓ Bootstrap

About Python

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assure that Python will work on all these operating systems.



Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations.



In addition, Python is the most popular language for machine learning and deep learning. As a matter of fact, today, all top organizations are investing in Python to implement machine learning in the back-end.

Python Line Structure

Python coding style comprises physical lines as well as logical lines or statements. A physical line in a Python program is a sequence of characters, and the end of the line terminates the line sequence as opposed to some other languages, such as C and C++ where a semi-colon is used to mark the end of the statement. A logical line, on the other hand, is composed of one or more physical lines. The use of a semi-colon is not prohibited in Python, although it's not mandatory. The NEWLINE token denotes the end of the logical line. A logical line that only contains spaces, comments, or tabs are called blank lines and they are ignored by the interpreter.

As we saw that in Python, a new line simply means that a new statement has started. Although, Python does provide a way to split a statement into a multiline statement or to join multiple statements into one logical line. This can be helpful to increase the readability of the statement. Following are the two ways to split a line into two or more lines:

Explicit Line Joining

In explicit line joining, we use a backward slash to split a statement into a multiline statement.

Implicit Line Joining

Statements that reside inside [], {}, or () parentheses can be broken down into two or more physical lines without using a back slash.

Multiple Statements on a Single Line

In Python, it is possible to club multiple statements in the same line using a semi-colon; however, most programmers do not consider this to be a good practice as it reduces the readability of the code.

Whitespaces and Indentation

Unlike most of the programming languages, Python uses indentation to mark a block of code.

According to Python coding style guideline or PEP8, we should keep an indent size of four.

Most of the programming languages provide indentation for better code formatting and don't enforce to have it. But in Python it is mandatory. This is why indentation is so crucial in Python.

Comments in any programming language are used to increase the readability of the code. Similarly, in Python, when the program starts getting complicated, one of the best ways to maintain the readability of the code is to use Python comments. It is considered a good practice to include documentations and notes in the python syntax since it makes the code way more readable and understandable to other programmers as well, which comes in handy when multiple programmers are simultaneously working on the same project.

The code can only explain how it does something and not why it does that, but Python comments can do that. With Python comments, we can make documentations for various explanations in our code itself. Comments are nothing but tagged lines of codes which increase the readability of a code and make it self-explanatory. There are different ways of creating comments depending on the type of comment we want to include in our code.

Following are different kinds of comments that can be included in our Python program:

1. Single Line Comments
2. Docstring Comments
3. Multiline Comments

Single line Python comments are marked with `#` character. These comments end at the end of the physical line, which means that all characters starting after the `#` character (and lasts till the end of the line) are part of the comment.

Python has the documentation strings (or docstrings) feature which is usually the first statement included in functions and modules. Rather than being ignored by the Python

Interpreter like regular comments, docstrings can actually be accessed at the run time using the dot operator.

It gives programmers an easy way of adding quick notes with every Python module, function, class, and method. To use this feature, we use triple quotes in the beginning of the documentation string or comment and the closing triple quotes at the end of the documentation comment. Docstrings can be one-liners as well as multi-liners.

Unlike some programming languages that support multiline comments, such as C, Java, and more, there is no specific feature for multiline comments in Python. But that does not mean that it is totally impossible to make multiline comments in Python. There are two ways we can include comments that can span across multiple lines in our Python code.

Python Block Comments: We can use several single line comments for a whole block. This type of comment is usually created to explain the block of code that follows the Block comment. Python Block comment is the only way of writing a real comment that can span across multiple lines. It is supported and preferred by Python's PEP8 style guide since Block comments are ignored by Python interpreter or parser. However, nothing is stopping programmers from using the second 'non-real' way of writing multiline comments in Python which is explained below.

Using Docstrings: Docstrings are largely used as multiline comments in Python by many programmers since it is the closest thing to having a multiline comment feature in Python. While it is not wrong to use docstrings when we need to make multiline comments, it is important to keep in mind that there is a significant difference between docstrings and comments. Comments in Python are totally ignored by the Python Interpreter, while docstrings, when used inside the Python function, can be accessed at the run time.

Python Data Types

One of the most crucial part of learning any programming language is to understand how data is stored and manipulated in that language. Users are often inclined toward Python because of its ease of use and the number of versatile features it provides. One of those features is dynamic typing.

In Python, unlike statically typed languages like C or Java, there is no need to specifically declare the data type of the variable. In dynamically typed languages such as Python, the interpreter itself predicts the data type of the Python Variable based on the type of value assigned to that variable.

Advantages of Python

- Universal Language Construct
- Support both High Level and Low Level Programming
- Language Interoperability
- Fastest Development life cycle therefore more productive coding environment
- Less memory used because a single container hold multiple data types and each type doesn't require its own function
- Learning Ease and open source development
- Speed and user-friendly data structure
- Extensive and extensible libraries.
- Simple & support IoT
- And many more

Based on the 3 languages that are described above, we decided to use Python as programming language for developing this e-Voting web based application. The important motives are

- Easy to learn, even non experienced programmers can use it. Ex: spacing and tabbing instead of extra syntax
- Interactive mode
- Large and comprehensive standard libraries
- Python programs resemble to that of pseudo-code. This makes it a basis and a must have for beginner programmers due to its extreme ease a difficulty when compared to C++ Java, Perl, and so forth

About Pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures.

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' DataFrames and

plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and DataFrames.

1. Series: a pandas Series is a one dimensional data structure (“a one dimensional ndarray”) that can store values — and for every value it holds a unique index, too.
2. DataFrame: a pandas DataFrame is a two (or more) dimensional data structure — basically a table with rows and columns. The columns have names and the rows have indexes.

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame’s rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.

Now, DataFrames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of potentially different types.

In general, you could say that the Pandas DataFrame consists of three main components: the data, the index, and the columns.

Firstly, the DataFrame can contain data that is:

- a Pandas DataFrame
- a Pandas Series: a one-dimensional labeled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a DataFrame.

- a NumPy ndarray, which can be a record or structured
- a two-dimensional ndarray
- dictionaries of one-dimensional ndarray's, lists, dictionaries or Series.

Some of the key features of Python Pandas are as follows:

It provides DataFrame objects with default and customized indexing which is very fast and efficient.

- There are tools available for loading data of different file formats into in-memory data objects.
- It is easy to perform data alignment and integrated handling of missing data in Python Pandas.
- It is very simple to perform pivoting and reshaping of data sets in Pandas.
- It also provides indexing, label-based slicing, and sub-setting of large data sets.
- We can easily insert and delete columns from a data structure.
- Data aggregation and transformations can be done using group by.
- High-performance merging and joining of data can be done using Pandas.
- It also provides time series functionality.
- Inserting and deleting columns in data structures.
- Merging and joining data sets.

- Reshaping and pivoting data sets.
- Aligning data and dealing with missing data.
- Manipulating data using integrated indexing for DataFrame objects.
- Performing split-apply-combine on data sets using the group by engine.
- Manipulating high-dimensional data in a data structure with a lower dimension using hierarchical axis indexing.
- Subsetting, fancy indexing, and label-based slicing data sets that are large in size.
- Generating data range, converting frequency, date shifting, lagging, and other time-series functionality.
- Reading from files with CSV, XLSX, TXT, among other formats.
- Arranging data in an order ascending or descending.
- Filtering data around a condition.
- Analyzing time series.
- Iterating over a data set.

Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of

nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

an array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

- The data pointer indicates the memory address of the first byte in the array,
- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array, and
- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

Or, in other words, an array contains information about the raw data, how to locate an element and how to interpret an element.

With NumPy, we work with multidimensional arrays. We'll dive into all of the possible types of multidimensional arrays later on, but for now, we'll focus on 2-dimensional arrays. A 2-dimensional array is also known as a matrix, and is something you should be familiar with. In fact, it's just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we're able to extract an element from a matrix.

We can create a NumPy array using the `numpy.array` function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we'll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to be of the same type, so if we include the header row, all the elements in the array will be read in as strings. Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats.

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

1. NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
2. Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

Matplotlib

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an object oriented way. Furthermore it can be used with general-purpose GUI toolkits like wxPython, Qt, and GTK+. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

CHAPTER 5

CONCLUSION

In this project, we proposed a simple transfer learning neural network on video object Detection. This simple convolutional neural network imposes less computational cost. On the basis of the transfer learning neural network , we also analyzed different methods of learning rate set and different optimization algorithm of solving the optimal parameters of the influence on video object Detection. We also verify that the shallow network also has a relatively good recognition effect.

Contrary to traditional methods that use hand-crafted features, our algorithm learns hierarchical features directly from normalized image pixels. We employ ReLU and dropout to enhance the training of our CNN. Experiments on public datasets show that our algorithm achieved state of the art performance

REFERENCES

- T. Kumagai, M. Walberer, H. Nakamura, H. Endepols, M. Sué, S. Vollmar, S. Adib, G. Mies, T. Yoshimine, M. Schroeter, and R. Graf, "Distinct spatiotemporal patterns of spreading depolarizations during early infarct evolution: evidence from real-time imaging," *J. Cereb. Blood Flow Metab.*, vol. 31, no. 2, pp. 580-592, Feb. 2011.
- L. Shi, J. Qin, R. Reif, and R. K. Wang, "Wide velocity range Doppler optical microangiography using optimized step-scanning protocol with phase variance mask," *J. Biomed. Opt.*, vol. 18, no. 10, pp. 106015, Oct. 2013.
- J. Xu, S. Song, Y. Li, and R. K. Wang, "Complex-based OCT angiography algorithm recovers microvascular information better than amplitude- or phase-based algorithms in phase-stable systems," *Phys. Med. Biol.*, vol. 63, no. 1, pp. 015023, Dec. 2017.
- W. J. Choi, Z. Zhi, and R. K. Wang, "In vivo OCT microangiography of rodent iris," *Opt. Lett.*, vol. 39, no. 8, pp. 2455-2458, Apr. 2014.
- G. Llovera, S. Roth, N. Plesnila, R. Veltkamp, and A. Liesz, "Modeling stroke in mice: permanent coagulation of the distal middle cerebral artery," *J. Vis. Exp.*, no. 89, pp. e51729, Jul. 2014.
- Y. Li, U. Baran, and R. K. Wang, "Application of thinned-skull cranial window to mouse cerebral blood flow imaging using optical microangiography," *PLoS One*, vol. 9, no. 11, pp. e113658, Nov. 2014.
- S. P. Chong, C. W. Merkle, D. F. Cooke, T. Zhang, H. Radhakrishnan, L. Krubitzer, and V. J. Srinivasan, "Noninvasive, in vivo imaging of subcortical mouse brain regions with 1.7 μm OCT," *Opt. Lett.*, vol. 40, no. 21, pp. 4911-4914, Nov. 2015.
- D. W. Wei, A. J. Deegan, and R. K. Wang, "Automatic motion correction for in vivo human skin OCTA through combined rigid and nonrigid registration," *J. Biomed. Opt.*, vol. 22, no. 6, pp. 66013, Jun. 2017.

Frederick P Rivara, Nathan Kuppermann, and Richard G Ellenbogen, "Use of clinical prediction rules for guiding use of computed tomography in adults with head trauma," *JAMA*, vol. 314, no. 24, pp. 2629– 2631, 2015.

Shu-Ling Chong, Su Yah Chew, Jasmine Xun Yi Feng, Penny Yun Lin Teo, Sock Teng Chin, Nan Liu, and Marcus Eng Hock Ong, "A prospective surveillance of paediatric head injuries in singapore: a dual-centre study," *BMJ open*, vol. 6, no. 2, pp. e010618, 2016.