International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Solar Cell Surface Defect Detection Based on Optimized Yolov5

V Sai Tarun

B. Tech Graduate, Sreenidhi Institute of Science and Technology

ABSTRACT

A solar cell defect detection method with an improved YOLO v5 algorithm is proposed for the characteristics of the complex solar cell image background, variable defect morphology, and large-scale differences. First, the deformable convolution is incorporated into the CSP module to achieve an adaptive learning scale and perceptual field size; then, the feature extraction capability of the model is enhanced by introducing the ECA-Net attention mechanism; finally, the model network structure is improved and one tiny defect prediction head is added to improve the accuracy of target detection at different scales. To further optimize and improve the YOLO v5 algorithm, this paper uses Mosaic and MixUp fusion data enhancement, K-meansCC clustering anchor box algorithm, and CIOU loss function to enhance the model performance. The experimental results show that the improved YOLO v5 algorithm achieves 89.64% mAP for the model trained on the solar cell EL image dataset, which is 7.85% higher than the mAP of the original algorithm, and the speed reaches 36.24 FPS, which can complete the solar cell defect detection task more accurately while meeting the real-time requirements.

In propose work we have modified YoloV5 but not experimented with advance YOLO family other version as like Yolo6, 7 or 8. So as extension we have trained same dataset with Yolov6 and its giving better prediction accuracy compare to Optimized Yolov5.

1. INTRODUCTION

At the present stage, under the dual pressure of environmental pollution and the increasingly prominent traditional energy crisis, people have turned their attention to the development and utilization of new energy sources [1]. Due to the advantages of a wide range of applications, low cost, safety, and reliability, solar energy has become one of the mainstream new energy sources with high-speed development. Solar panels are important components of photovoltaic power generation, silicon crystal plates are fragile and fragile, and defects are easily produced by improper operation in production and installation [2], these defects cannot only affect the efficiency of solar cell power generation but also seriously threaten people's life and property safety [3]. Therefore, the study of solar cell defect detection methods is of great significance [4]. Electroluminescence (EL) imaging involves injecting a forward bias current into the PV module to put it in an excited state and then using a silicon charge-coupled device (CCD) or an InGaAs camera to capture the infrared light generated by the solar cell in the excited state for imaging. With the advantages of nondestructive and noncontact, electroluminescence imaging cannot only effectively detect tiny cracks, finger interruption, and other process defects that cannot be observed by conventional imaging systems, but also avoid blurring of imaging caused by lateral thermal propagation [5], [6]. Based on its excellent performance, electroluminescence imaging has become the main way of solar cell defect detection.

1.1 OBJECTIVE

The objective of this study is to enhance solar cell defect detection through a refined YOLOv5 algorithm, incorporating deformable convolution in the CSP module for adaptive learning scales and perceptual field sizes. The addition of the ECA-Net attention mechanism improves feature extraction, while a modified network structure with an additional tiny defect prediction head enhances detection accuracy across scales. Mosaic and MixUp fusion, K-meansCC clustering anchor box algorithm, and CIOU loss function contribute to algorithm optimization. While achieving a significant 7.85% improvement in mean Average Precision (mAP) over the original YOLOv5, the study highlights the potential of further experimentation with advanced YOLO versions for solar cell defect detection.

1.2 PROBLEM STATEMENT

Solar cell defect detection poses challenges due to complex image backgrounds, variable defect morphologies, and large-scale differences. Existing methods, including YOLOv5, encounter limitations in adaptive learning scales and perceptual field sizes, affecting accuracy. This study addresses these issues by proposing an improved YOLOv5 algorithm, incorporating deformable convolution in the CSP module, introducing the ECA-Net attention mechanism, and modifying the network structure. However, the study has not explored advanced YOLO versions like YOLOv6, 7, or 8. The problem statement revolves around the need to enhance solar cell defect detection methods to achieve higher accuracy and real-time performance, considering the unique characteristics of solar cell images.

1.3 SOFTWARE REQUIREMENTS

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

Platform – In computing, a platform describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer's architecture, operating system, or programming languages and their runtime libraries.

Operating system is one of the first requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of same line of operating systems, although some measure of backward compatibility is often maintained. For example, most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, software designed using newer features of Linux Kernel v2.6 generally does not run or compile properly (or at all) on Linux distributions using Kernel v2.2 or v2.4.

APIs and drivers – Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms.

Web browser – Most web applications and software depending heavily on Internet technologies make use of the default browser installed on system. Microsoft Internet Explorer is a frequent choice of software running on Microsoft Windows, which makes use of ActiveX controls, despite their vulnerabilities.

1. Software : Anaconda
2. Primary Language : Python
3. Frontend Framework : Flask
4. Back-end Framework : Jupyter Notebook

5. Database : Sqlite3
6. Front-End Technologies : HTML, CSS, JavaScript and Bootstrap4

1.4 HARDWARE REQUIREMENTS

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

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

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

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

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

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

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

1. **Operating System : Windows Only**
2. **Processor : i5 and above**
3. **Ram : 8gb and above**
4. **Hard Disk : 25 GB in local drive**

2. FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

2.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

LITERATURE SURVEY

3. LITERATURE SURVEY

3.1 Deep CNN-based visual defect detection: Survey of current literature:

https://ouci.dntb.gov.ua/en/works/l1wXbky7/

ABSTRACT: In the past years, the computer vision domain has been profoundly changed by the advent of deep learning algorithms and data science. The defect detection problem is of outmost importance in high-tech industries such as aerospace manufacturing and is extensively employed using automated industrial quality control systems. Defect inspection methods can be mainly grouped into manual inspection, traditional computer vision, and modern computer vision inspection. Initially developed two decades ago, the CNN algorithms recently became popular for solving complex machine vision problems, as big datasets and computationally potent hardware became widely available. Deep learning-based methods form the foundation for modern automatic optical inspection methods and can be grouped based on their network connections into two categories: dense networks and sparse networks. Another method for grouping considers the type of learning: supervised learning used primarily for defect classification and segmentation, and unsupervised learning models, which have the potential to overcome the challenges of supervised models such as labeling images and annotating pixels. In addition, pixel-level based segmentation techniques are considered to cover the state-of-the-art methodologies for the automatic optical inspection. Still, both supervised and unsupervised models pose challenges in regards to model training and attaining the expected detection accuracy. Identified open

challenges include algorithmic, application, and data processing challenges. By addressing these challenges, in the future, the demand for automated optical inspection is expected to only grow in both industry practice and academic research.

3.2 Cross-convolutional-layer Pooling for Image Recognition:

https://arxiv.org/pdf/1510.00921.pdf#:~:text=We%20name%20our%20method%20cross,successive%20convolutional%20layer%20as%20guidance.

ABSTRACT: Recent studies have shown that a Deep Convolutional Neural Network (DCNN) trained on a large image dataset can be used as a universal image descriptor and that doing so leads to impressive performance for a variety of image recognition tasks. Most of these studies adopt activations from a single DCNN layer, usually a fully-connected layer, as the image representation. In this paper, we proposed a novel way to extract image representations from two consecutive convolutional layers: one layer is used for local feature extraction and the other serves as guidance to pool the extracted features. By taking different viewpoints of convolutional layers, we further develop two schemes to realize this idea. The first directly uses convolutional layers from a DCNN. The second applies the pre-trained CNN on densely sampled image regions and treats the fully-connected activations of each image region as a convolutional layer's feature activations. We then train another convolutional layer on top of that as the pooling-guidance convolutional layer. By applying our method to three popular visual classification tasks, we find that our first scheme tends to perform better on applications which demand strong discrimination on lower-level visual patterns while the latter excels in cases that require discrimination on category-level patterns. Overall, the proposed method achieves superior performance over existing approaches for extracting image representations from a DCNN. In addition, we apply cross-layer pooling to the problem of image retrieval and propose schemes to reduce the computational cost. Experimental results suggest that the proposed method achieves promising results for the image retrieval task.

3.3 Exploiting Convolutional Neural Networks With Deeply Local Description for Remote Sensing Image Classification:

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8270691

ABSTRACT: The extraction of features from the fully connected layer of a convolutional neural network (CNN) model is widely used for image representation. However, the features obtained by the convolutional layers are seldom investigated due to their high dimensionality and lack of global representation. In this study, we explore the uses of local description and feature encoding for deeply convolutional features. Given an input image, the image pyramid is constructed, and different pretrained CNNs are applied to each image scale to extract convolutional features. Deeply local descriptors can be obtained by concatenating the convolutional features in each spatial position. Hellinger kernel and principal component analysis (PCA) are introduced to improve the distinguishable capabilities of the deeply local descriptors. The Hellinger kernel causes the distance measure to be sensitive to small feature values, and the PCA helps reduce feature redundancy. In addition, two aggregate strategies are proposed to form global image representations from the deeply local descriptors. The first strategy aggregates the descriptors of different CNNs by Fisher encoding, and the second strategy concatenates the Fisher vectors of different CNNs. Experiments on two remote sensing image datasets illustrate that the Hellinger kernel, PCA, and two aggregate strategies improve classification performance. Moreover, the deeply local descriptors outperform the features extracted from fully connected layers.

3.4 Convolutional Recurrent Deep Learning Model for Sentence Classification:
https://ieeexplore.ieee.org/document/8314136

ABSTRACT: As the amount of unstructured text data that humanity produces overall and on the Internet grows, so does the need to intelligently to process it and extract different types of knowledge from it. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been applied to natural language processing systems with comparative, remarkable results. The CNN is a noble approach to extract higher level features that are invariant to local translation. However, it requires stacking multiple convolutional layers in order to capture long-term dependencies, due to the locality of the convolutional and pooling layers. In this paper, we describe a joint CNN and RNN framework to overcome this problem. Briefly, we use an unsupervised neural language model to train initial word embeddings that are further tuned by our deep learning network, then, the pre-trained parameters of the network are used to initialize the model. At a final stage, the proposed framework combines former information with a set of feature maps learned by a convolutional layer with long-term dependencies learned via long-short-term memory. Empirically, we show that our approach, with slight hyperparameter tuning and static vectors, achieves outstanding results on multiple sentiment analysis benchmarks. Our approach outperforms several existing approaches in term of accuracy; our results are also competitive with the state-of-the-art results on the Stanford Large Movie Review data set with 93.3% accuracy, and the Stanford Sentiment Treebank data set with 48.8% fine-grained and 89.2% binary accuracy, respectively. Our approach has a significant role in reducing the number of parameters and constructing the convolutional layer followed by the recurrent layer as a substitute for the pooling layer. Our results show that we were able to reduce the loss of detailed, local information and capture long-term dependencies with an efficient framework that has fewer parameters and a high level of performance.

3.5 A Novel Text Structure Feature Extractor for Chinese Scene Text Detection and Recognition:
https://ieeexplore.ieee.org/document/7870570

ABSTRACT: Scene text information extraction plays an important role in many computer vision applications. Most features in existing text extraction algorithms are only applicable to one text extraction stage (text detection or recognition), which significantly weakens the consistency in an end-to-end system, especially for the complex Chinese texts. To tackle this challenging problem, we propose a novel text structure feature extractor based on a text structure component detector (TSCD) layer and residual network for Chinese texts. Inspired by the three-layer Chinese text cognition model of a human, we combine the TSCD layer and the residual network to extract features suitable for both text extraction stages. The specialized modeling for Chinese characters in the TSCD layer simulates the key structure component cognition layer in the psychological model. And the residual mechanism in the residual network simulates the key bidirectional connection among the layers in the psychological model. Through the organic combination of the TSCD layer and the residual network, the extracted features are applicable to both text detection and recognition, as humans do. In evaluation, both text detection and recognition models based on our proposed text structure feature extractor achieve great improvements over baseline CNN models. And an end-to-end Chinese text information extraction system is experimentally designed and evaluated, showing the advantage of the proposed feature extractor as a unified feature extractor.

4. SYSTEM ANALYSIS

4.1 EXISTING SYSTEM:

Traditional visual inspection requires operation and maintenance engineers to carry instruments to inspect solar cells one by one, which is a high workload, low efficiency, and overly dependent on the subjective experience of O&M engineers, and the inspection accuracy cannot be guaranteed. To automatically and accurately identify defects in images, researchers have proposed traditional computer vision based on manual feature extraction and classifiers. Tsai et al. proposed a method for detecting defects in polysilicon solar cells based on the Fourier image reconstruction technique, which removes possible defects in EL images by setting the frequency components of line and strip defects to 0. Demant et al. proposed a classification recognition method based on local descriptors and support vector machines, which achieves effective detection of photoluminescence (PL) images and infrared (IR) images of small-grain silicon wafers. However, traditional computer vision relies on manual extraction of descriptors, which requires a large number of parameter adjustments and has poor robustness and generalization capabilities.

Disadvantages:

1. However, most convolutional neural networks are designed for natural scene images, and the direct application of mature deep learning models to detect surface defects in solar cell EL images has inapplicable problems
2. solar cell defect detection is susceptible to interference from complex backgrounds
3. solar cells have diversity in the shape of the same class of defects
4. as network training progresses and downsampling continues, micro defect features such as cracks and finger interruption tend to disappear

4.2 Proposed System:

After the above analysis and demonstration, the firstorder detector YOLO v5 plays an important role in target detection with powerful real-time processing capability and low hardware requirements, which can be ported to mobile devices for real-time monitoring. Based on this, this paper proposes an improved YOLO v5 model for three different characteristics of solar cell surface defects, namely, cracks, black core, and finger interruption. In the design of the improved YOLO v5 network, deformable convolution is introduced into the CSP module to achieve effective extraction of defects of different sizes and shapes; and the ECA-Net attention module is introduced in the Neck part to achieve improved detection performance through cross-channel interaction; meanwhile, the model structure is optimized and the prediction head is added to achieve four-scale feature defect detection and improve the detection accuracy of tiny defects. Finally, the detection effect of the improved model in this paper is objectively evaluated through experiments such as ablation experiments and a comparison of mainstream methods, and the results show that the improved model improves the detection accuracy of solar cell defects while ensuring the real-time detection.

In this work has modified YoloV5, but has not experimented with other advanced versions of the Yolo family, such as Yolo6, 7, or 8. Consequently, we trained Yolov6 on the same dataset as an extension, and it produced predictions with a higher accuracy than Optimized Yolov5.

Advantages:

1. Part of the conventional convolution in the CSP module is replaced with deformable convolution to realize the detection network adaptive learning of feature point receptive fields and effectively extract defect features of different sizes and shapes

2. Adding the Neck part to the ECA-Net of the deep convolutional network to achieve considerable performance improvement by adding only a small number of parameters through a local cross-information interaction strategy without dimensionality reduction.
3. Improving the network structure and optimizing the parameters of the YOLO v5 detection model, and increasing the number of prediction heads from 3 to 4. The new prediction heads use shallow features to achieve the detection of micro defects, making the improved model more applicable to solar cell surface defect detection.
4. Using the K-meansCC algorithm for anchor box clustering, the improved clustering anchor box size is more in line with the data set, effectively reducing the impact of initial points on the clustering results and speeding up the convergence of network training; at the same time replacing the loss function of the detection network with the complete loss function (CIOU), making the improved prediction box more in line with the real box.
5. The mosaic data augmentation and Mixup data augmentation are proportionally fused for data expansion, which effectively reduces the memory loss of data augmentation while satisfying the demand for data expansion.

4.3 FUNCTIONAL REQUIREMENTS

1. Data Collection
2. Data Pre-processing
3. Training and Testing
4. Modiling
5. Predicting

4.4 NON FUNCTIONAL REQUIREMENTS

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

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

- Environmental requirement

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

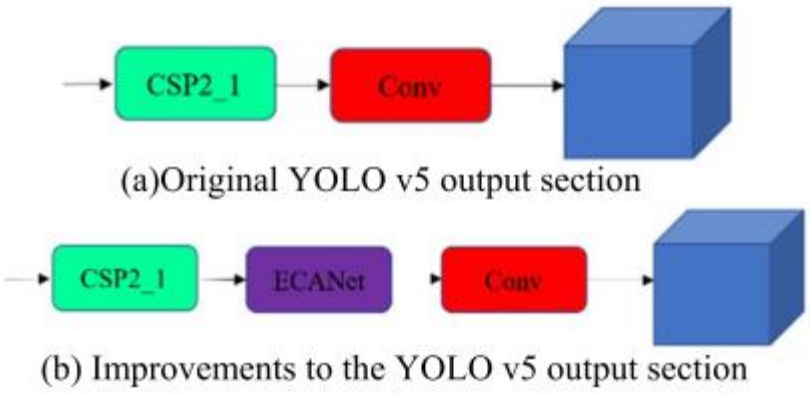
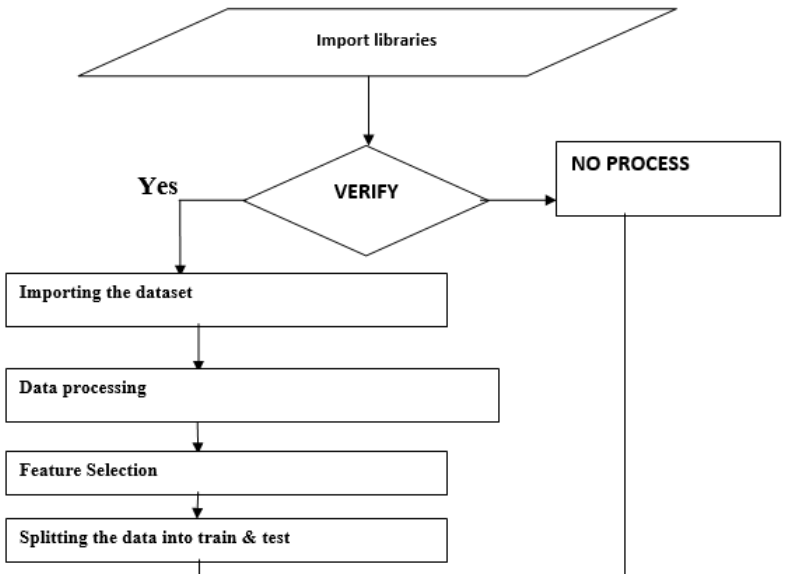


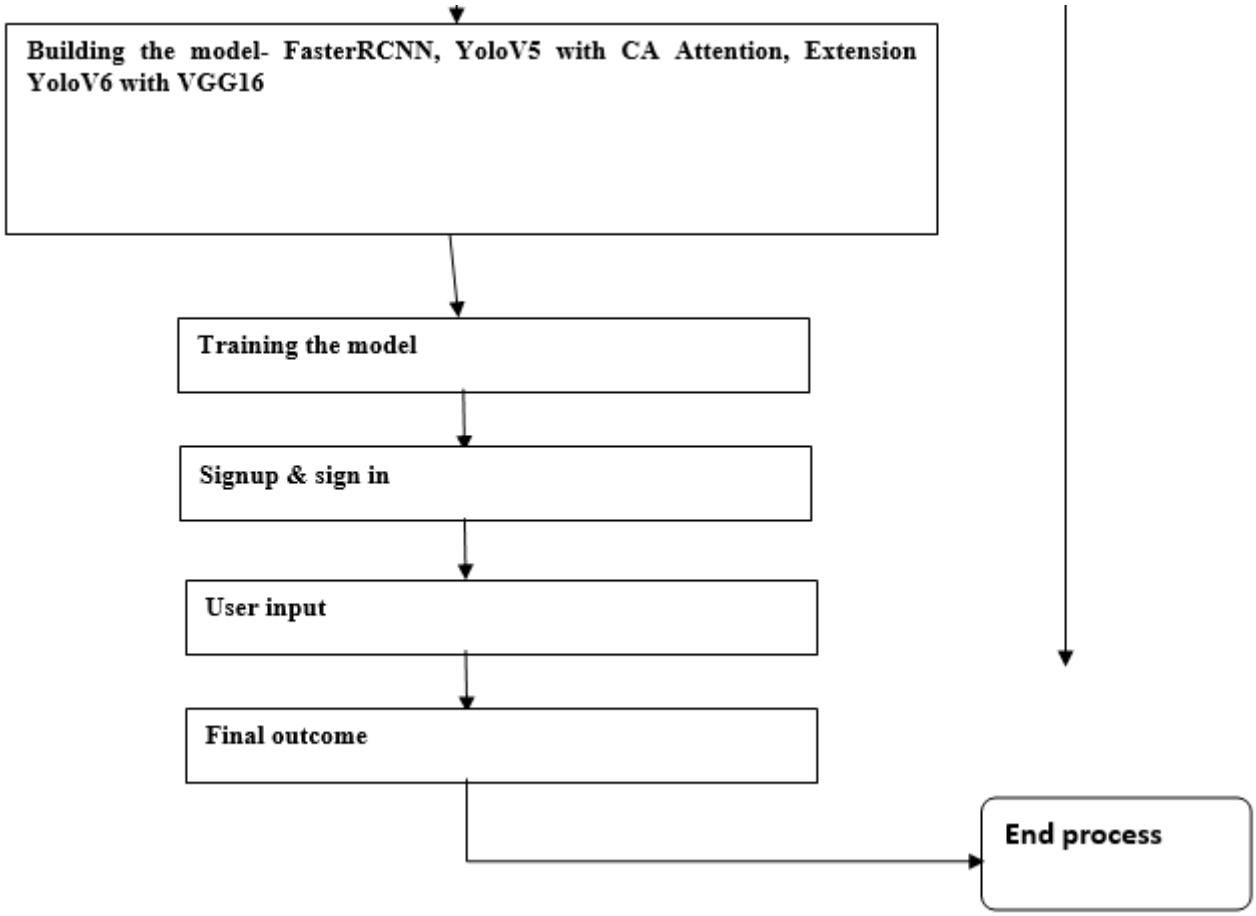
Fig.5.1.1 System architecture

DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

5.2 UML DIAGRAMS





UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

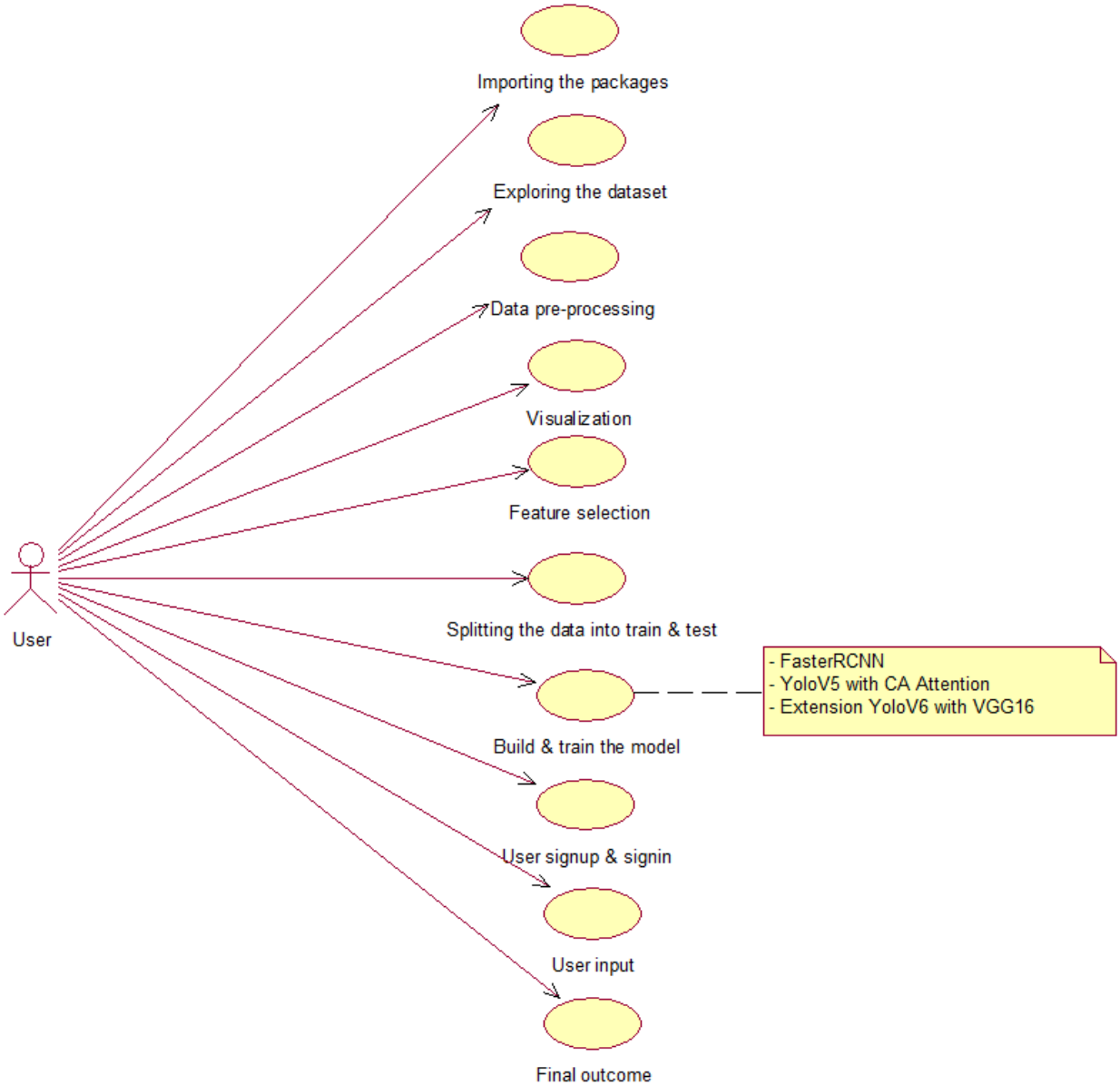
The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.

6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

Use case diagram:

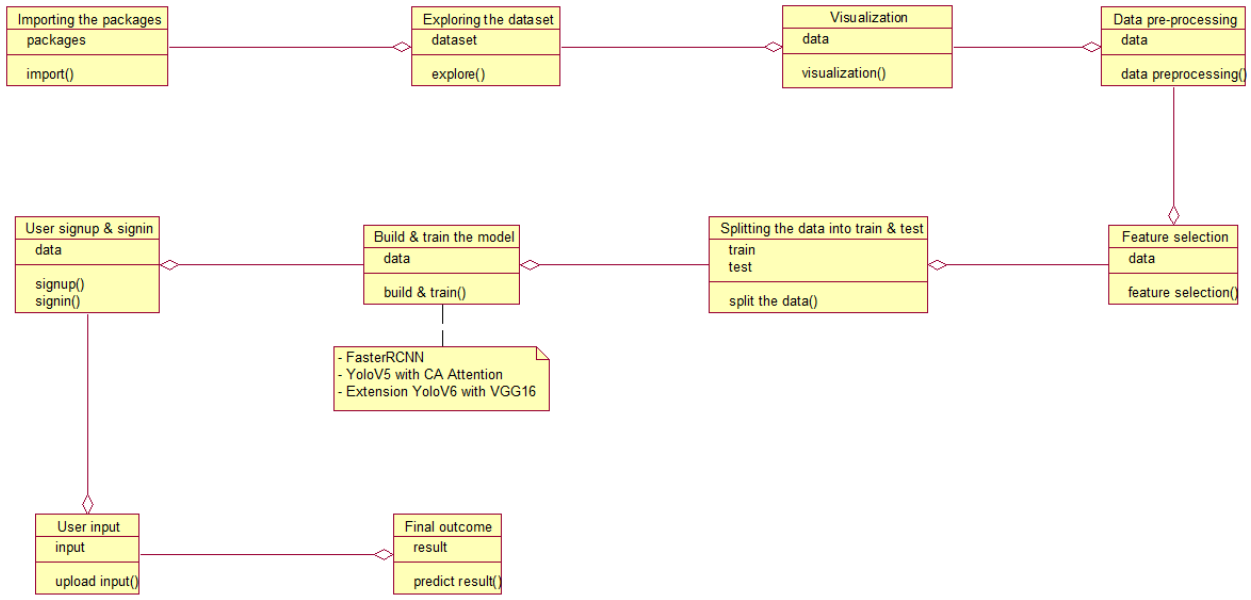
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies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



Class diagram:

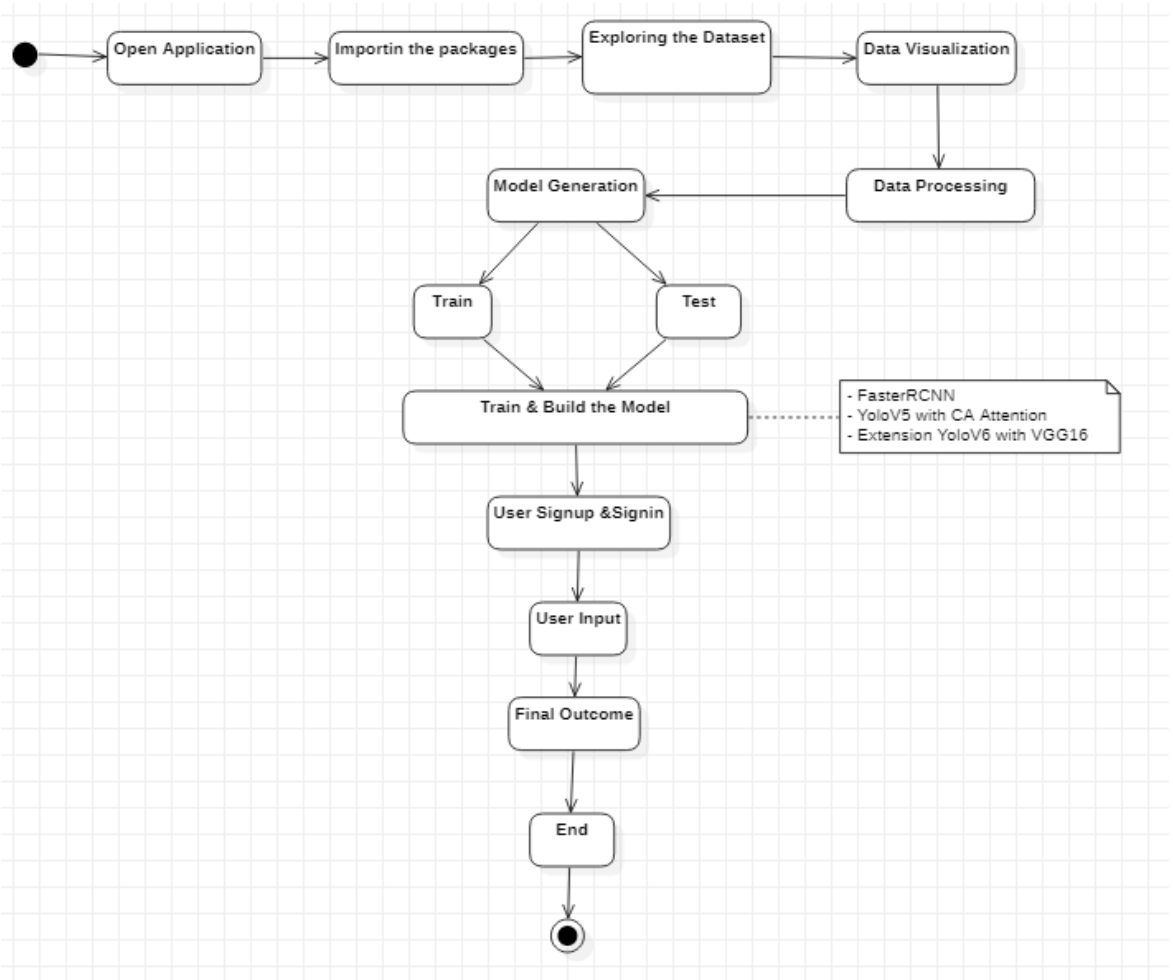
The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided

by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.



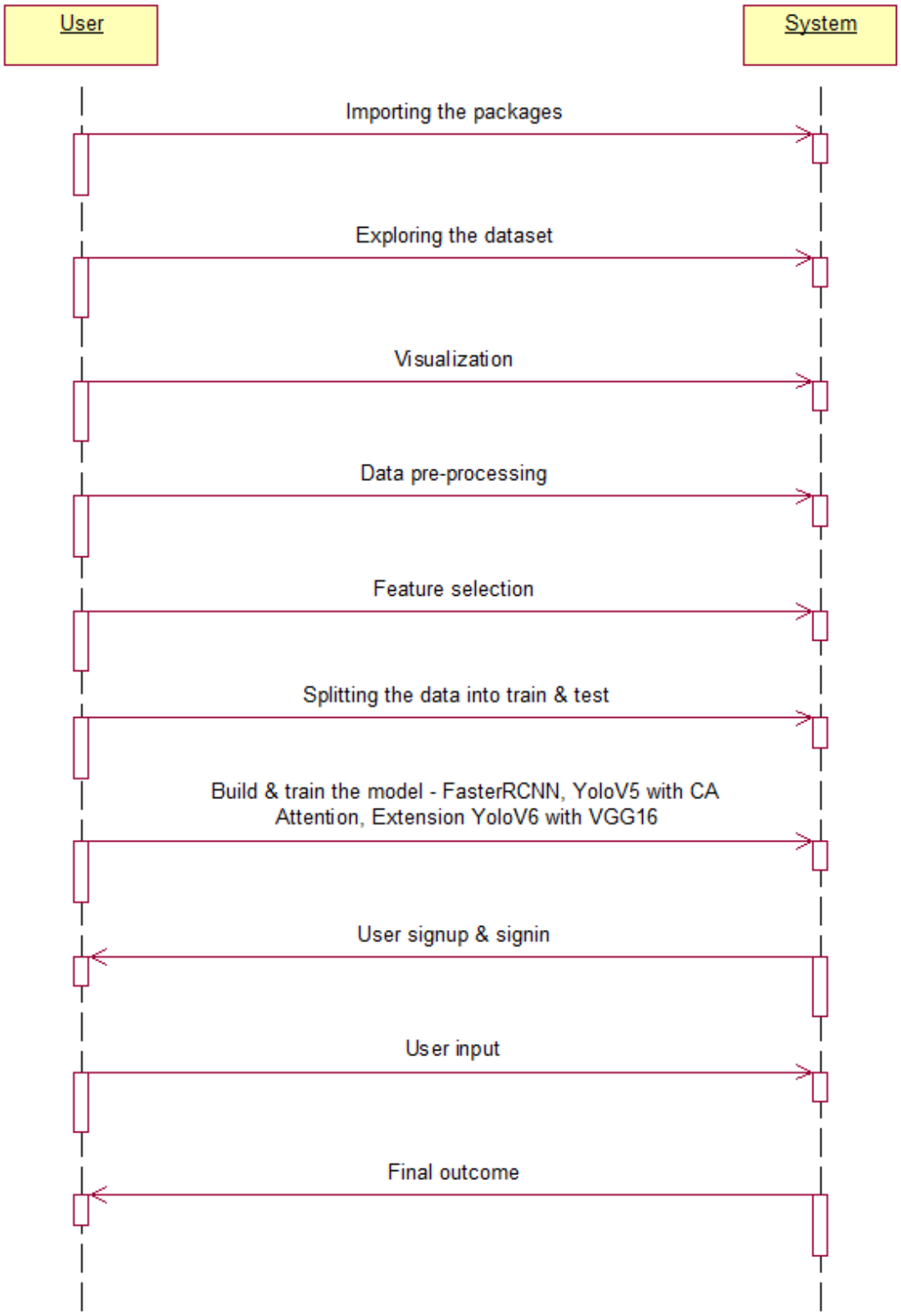
Activity diagram:

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.



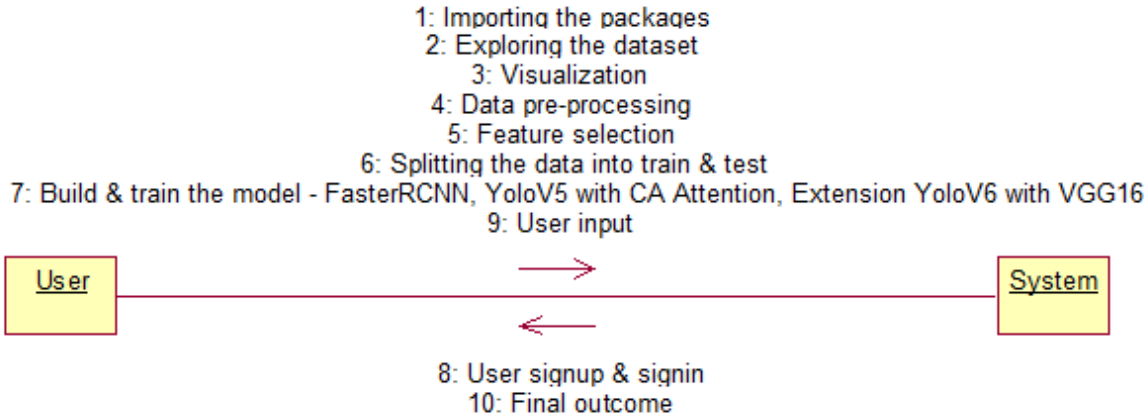
Sequence diagram:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".



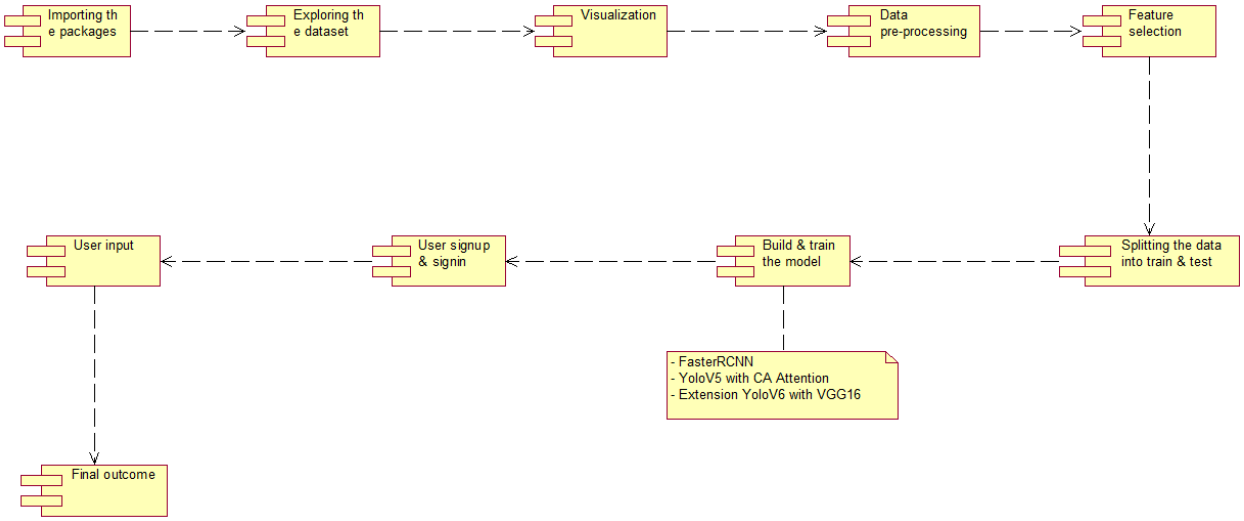
Collaboration diagram:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.



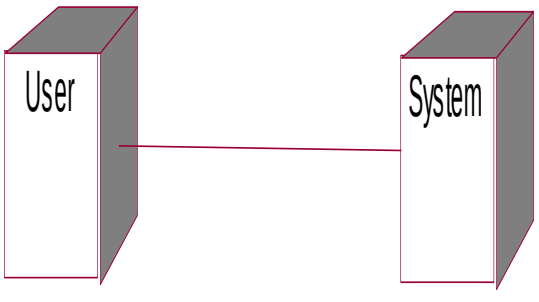
Component diagram:

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.



Deployment diagram:

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.



6. IMPLEMENTATION

MODULES:

- Importing required packages for image processing and machine learning.
- Setting bounding box or defect probability for anomaly detection.
- Exploring the dataset by extracting feature data from images.
- Visualizing dataset using OpenCV for better understanding and analysis.
- Preprocessing the dataset by normalizing and shuffling images for training.
- Performing feature selection to enhance model efficiency and accuracy.
- Splitting the data into training and testing sets for evaluation.
- Creating a function to calculate metrics scores for model assessment.
- Building models including FasterRCNN, YoloV5 with CA Attention, and YoloV6 with VGG16 extension.
- Training and comparing the performance of the constructed models.
- Making predictions using the trained models for anomaly detection.
- User signup & login: Using this module will get registration and login
- User input: Using this module will give input for prediction
- Prediction: final predicted displayed

Note: As an extension we applied an ensemble method combining the predictions of multiple individual models to produce a more robust and accurate final prediction.

However, we can further enhance the performance by exploring other ensemble techniques such as YoloV6, which got 98% of accuracy.

Algorithms:

Faster R-CNN:

Faster R-CNN, or Faster Region-based Convolutional Neural Network, is an object detection algorithm that introduces the Region Proposal Network (RPN) for efficient region proposals. By combining deep convolutional networks and region-based detection, Faster R-CNN achieves high accuracy in locating and classifying objects within images. It excels in real-time applications due to its balanced speed and precision, making it a popular choice for computer vision tasks.

YoloV5 with CA Attention:

YoloV5, or You Only Look One-level, enhanced with Channel Attention (CA) mechanism, refines object detection by focusing on important features. CA Attention dynamically weights channel-wise information, improving the model's ability to capture context and details. This attention mechanism enhances YoloV5's performance, particularly in scenarios with complex backgrounds or overlapping objects, ensuring robust and accurate object localization and classification.

Extension YoloV6 with VGG16:

The extended YoloV6, incorporating the VGG16 architecture, combines the strengths of YOLO's efficiency with the deep feature extraction capabilities of VGG16. VGG16's multiple convolutional layers enable YoloV6 to capture intricate hierarchical features, enhancing object recognition accuracy. This extension adapts the YOLO framework to leverage VGG16's powerful feature representation, resulting in improved performance for object detection tasks, especially in scenarios with diverse and challenging visual elements. The fusion of YOLO and VGG16 architectures in YoloV6 creates a potent model for comprehensive and precise object detection in various applications.

6.2 SAMPLE CODE:

7. SOFTWARE ENVIRONMENT

MACHINE LEARNING:

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*. Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Challenges in Machines Learning:-

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

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

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

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

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

Issue of over fitting & under fitting − If the model is over fitting or under fitting, it cannot be represented well for the problem.

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

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

DEEP LEARNING

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don’t need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs). These neural networks are

inspired by the structure and function of the human brain's biological neurons, and they are designed to learn from large amounts of data.

What is Anaconda for Python?

Anaconda Python is a free, open-source platform that allows you to write and execute code in the programming language Python. It is by continuum.io, a company that specializes in Python development. The Anaconda platform is the most popular way to learn and use Python for scientific computing, data science, and machine learning. It is used by over thirty million people worldwide and is available for Windows, macOS, and Linux.

People like using Anaconda Python because it simplifies package deployment and management. It also comes with a large number of libraries/packages that you can use for your projects. Since Anaconda Python is free and open-source, anyone can contribute to its development.

What is Anaconda for Python?

Anaconda software helps you create an environment for many different versions of Python and package versions. Anaconda is also used to install, remove, and upgrade packages in your project environments. Furthermore, you may use Anaconda to deploy any required project with a few mouse clicks. This is why it is perfect for beginners who want to learn Python.

Now that you know what Anaconda Python is, let's look at how to install it.

How to install Anaconda for Python?

To install Anaconda, just head to the Anaconda Documentation website and follow the instructions to download the installer for your operating system. Once the installer successfully downloads, double-click on it to start the installation process.

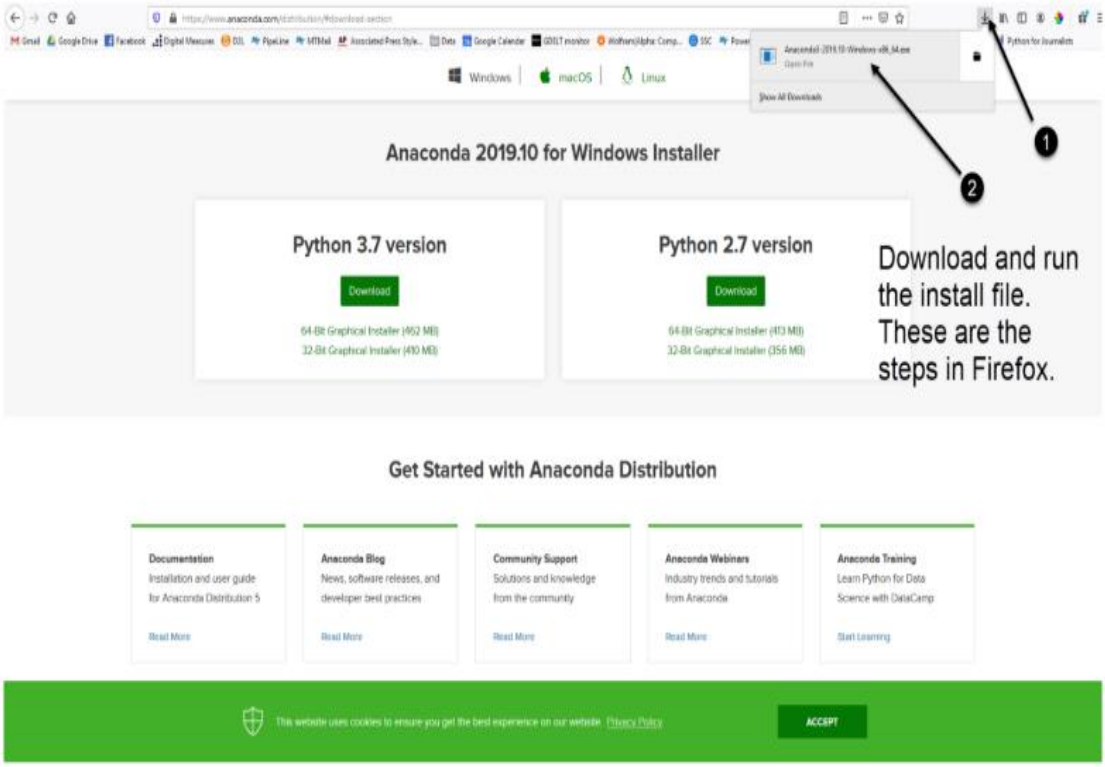
Follow the prompts and agree to the terms and conditions. When you are asked if you want to "add Anaconda to my PATH environment variable," make sure that you select "yes." This will ensure that Anaconda is added to your system's PATH, which is a list of directories that your operating system uses to find the files it needs.

Once the installation is complete, you will be asked if you want to "enable Anaconda as my default Python." We recommend selecting "yes" to use Anaconda as your default Python interpreter.

Python Anaconda Installation

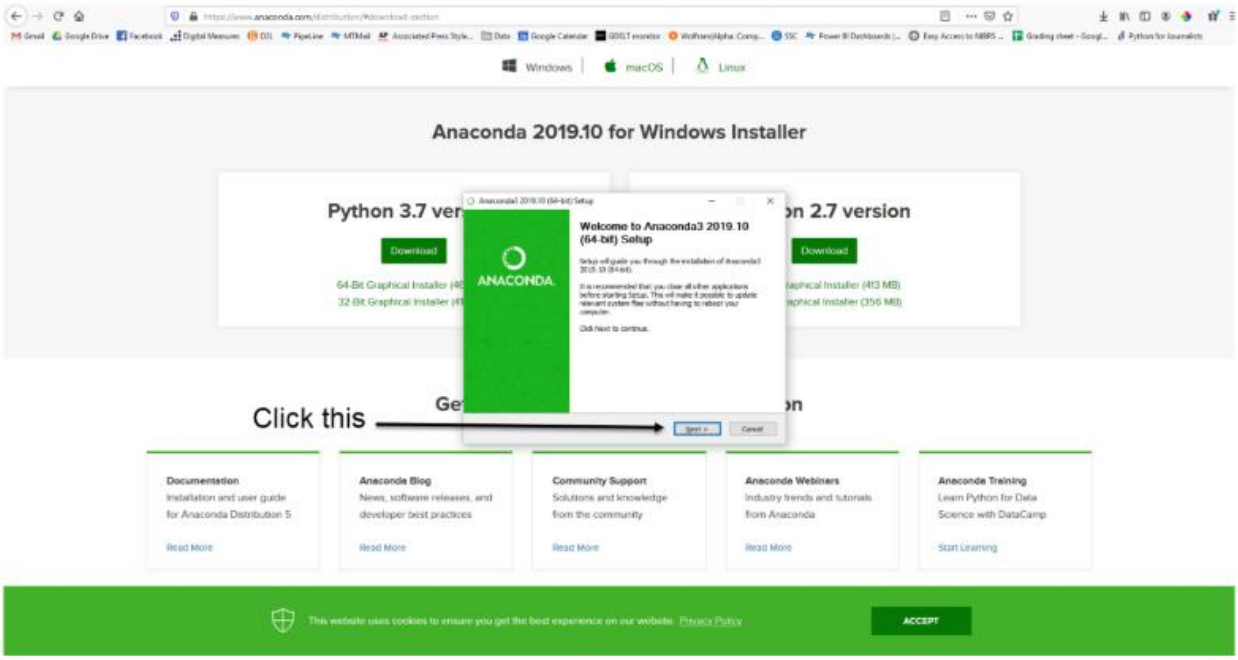
Next in the Python anaconda tutorial is its installation. The latest version of Anaconda at the time of writing is 2019.10. Follow these steps to download and install Anaconda on your machine:

1. Go to this link and download Anaconda for Windows, Mac, or Linux: – Download anaconda

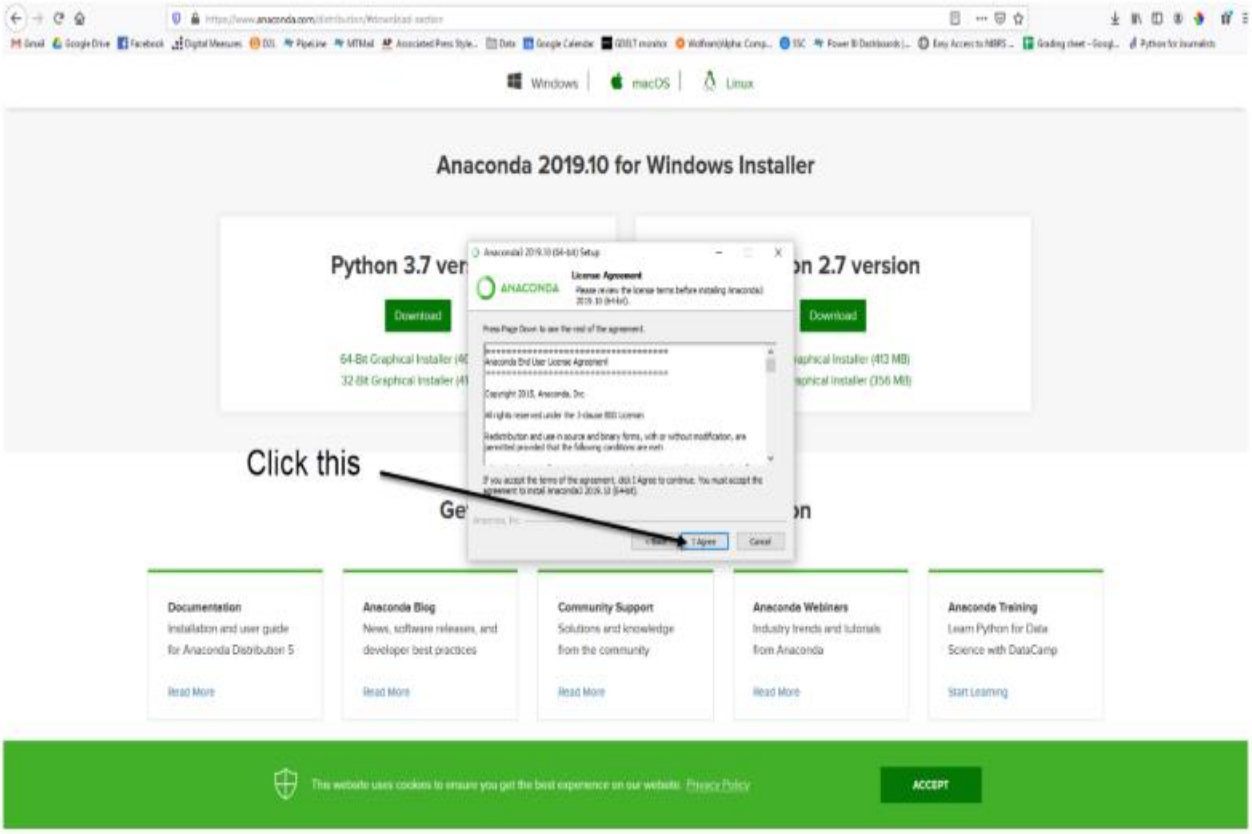


You can download the installer for Python 3.7 or for Python 2.7 (at the time of writing). And you can download it for a 32-bit or 64-bit machine.

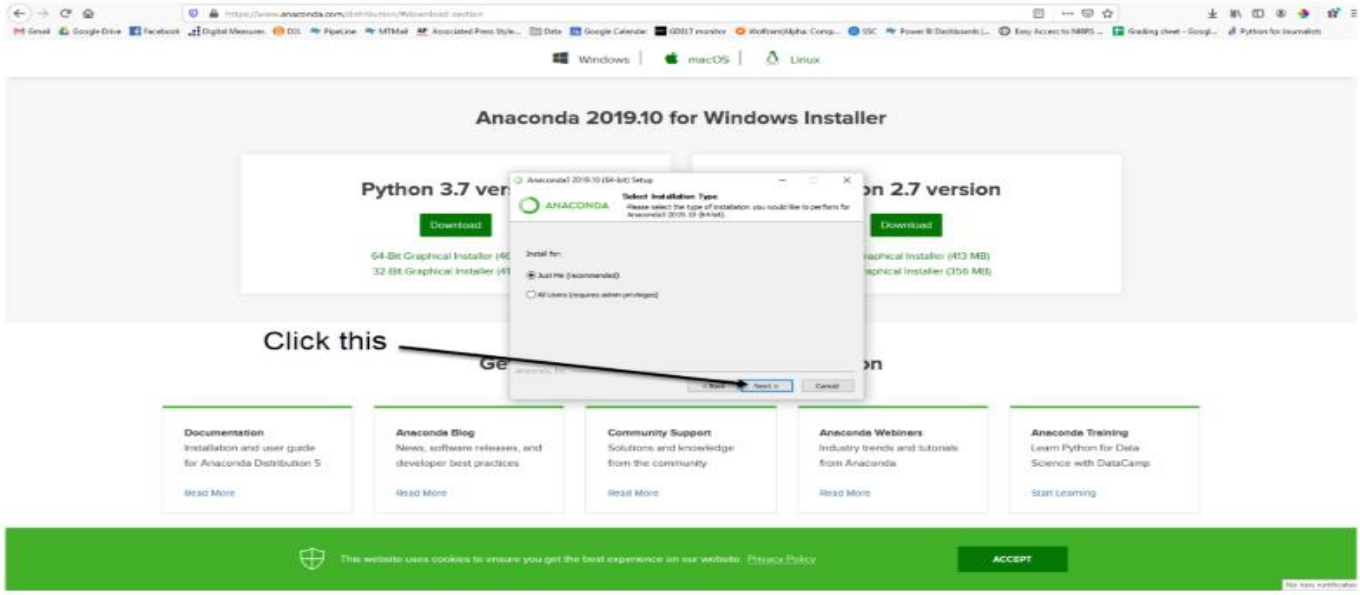
2. Click on the downloaded .exe to open it. This is the Anaconda setup. Click next.



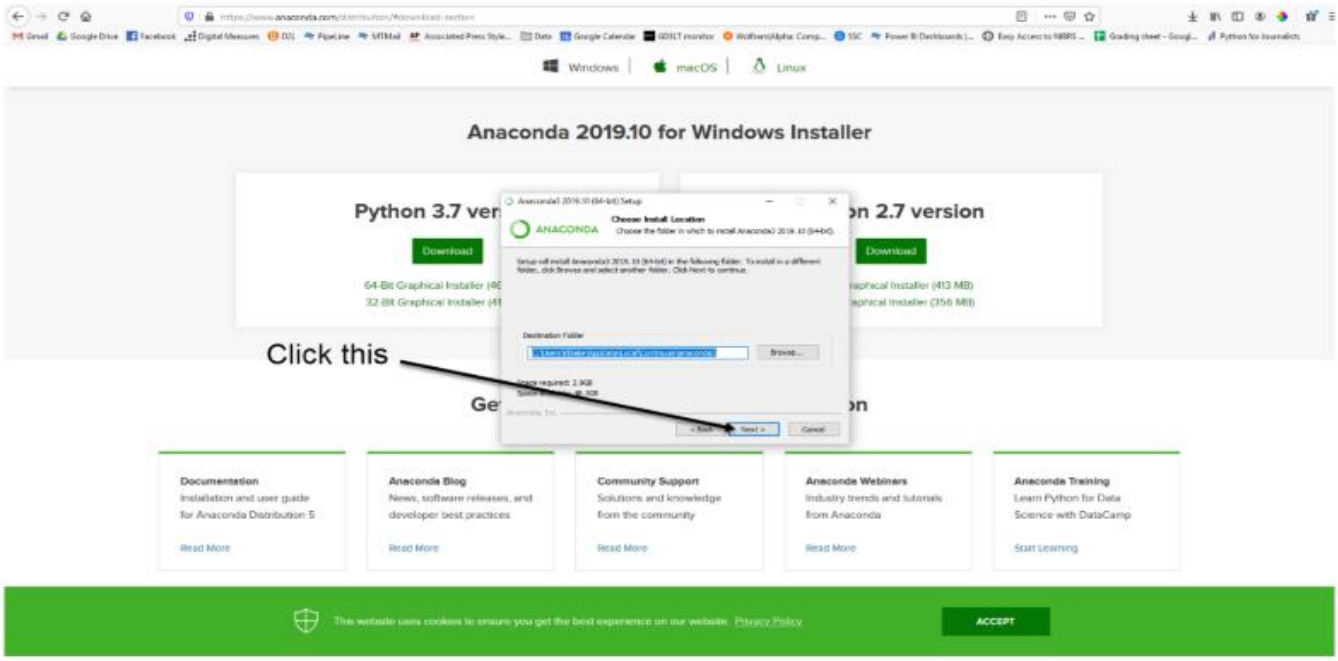
3. Now, you'll see the license agreement. Click on 'I Agree'.



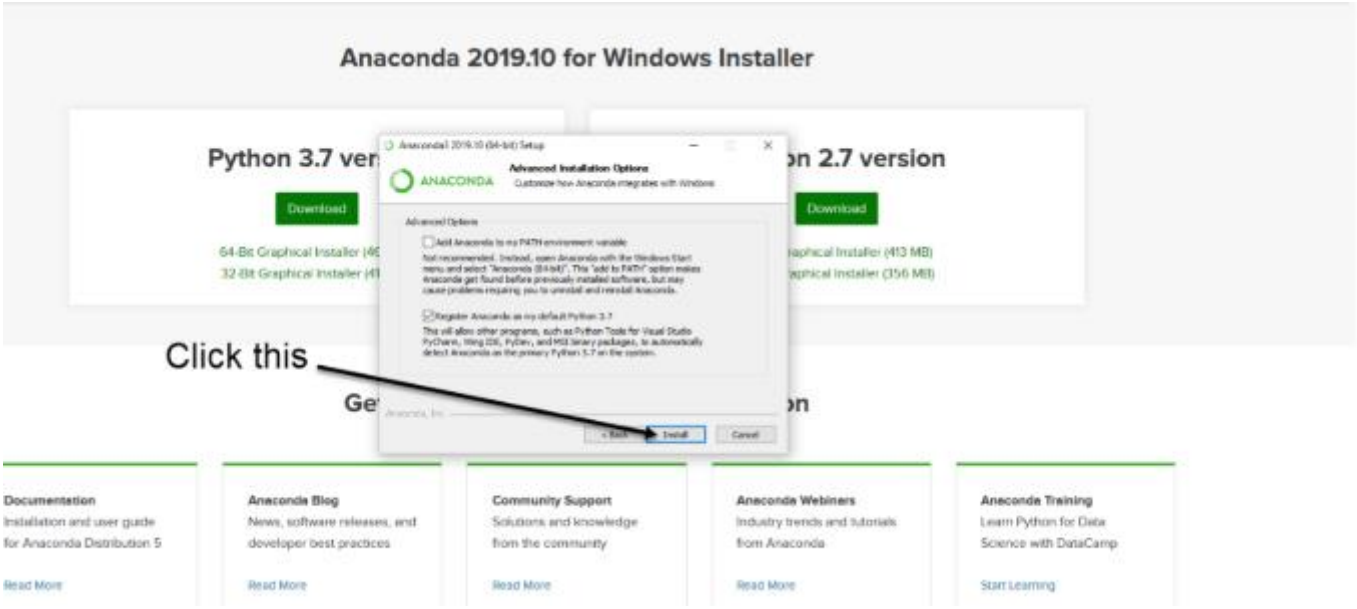
4. You can install it for all users or just for yourself. If you want to install it for all users, you need administrator privileges.



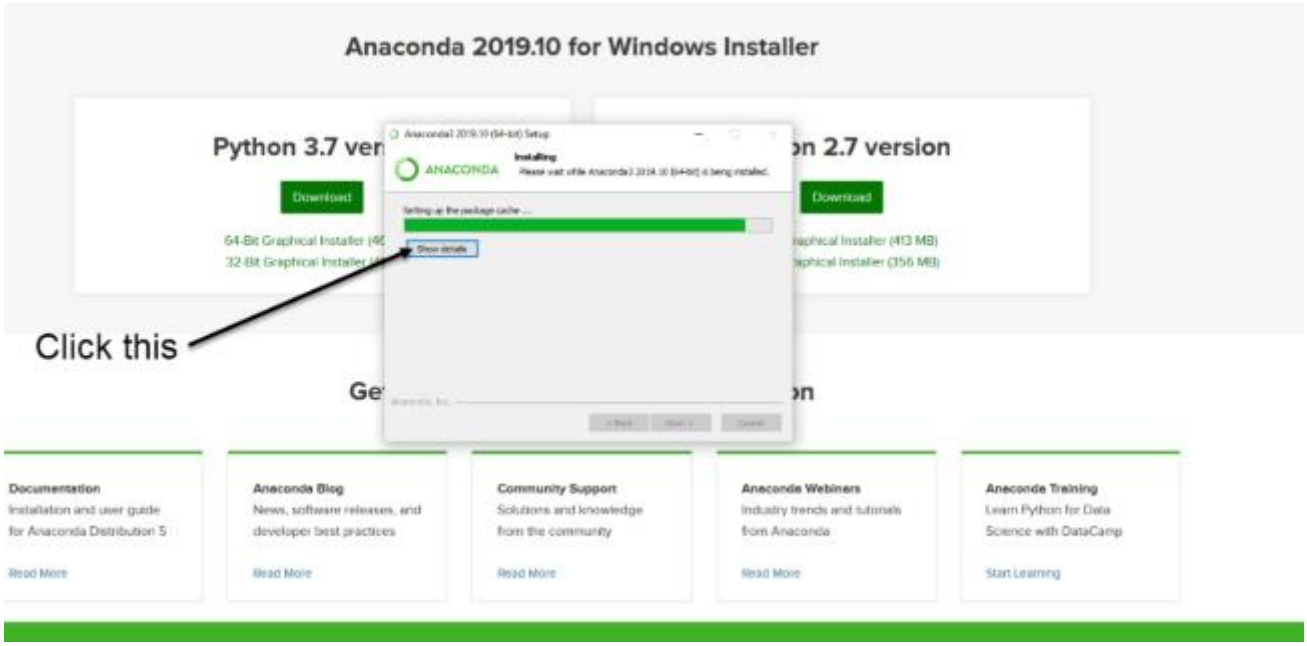
5. Choose where you want to install it. Here, you can see the available space and how much you need.



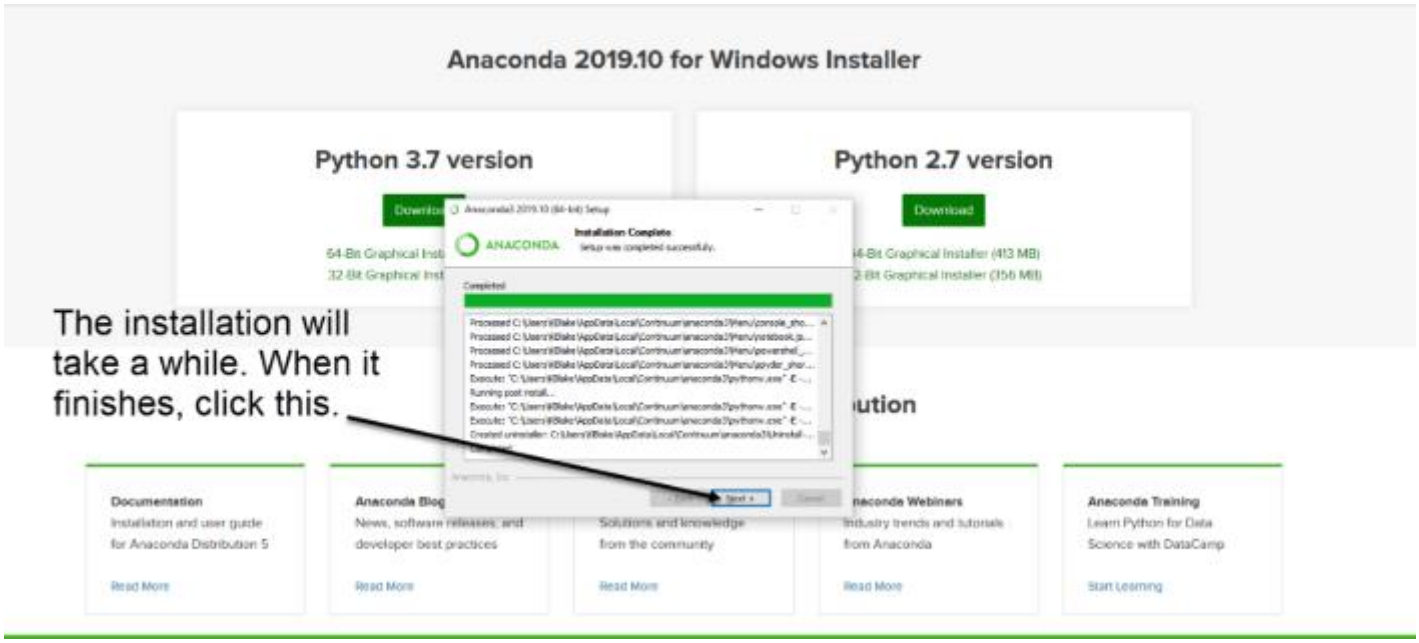
6. Now, you'll get some advanced options. You can add Anaconda to your system's PATH environment variable, and register it as the primary system Python 3.7. If you add it to PATH, it will be found before any other installation. Click on 'Install'.



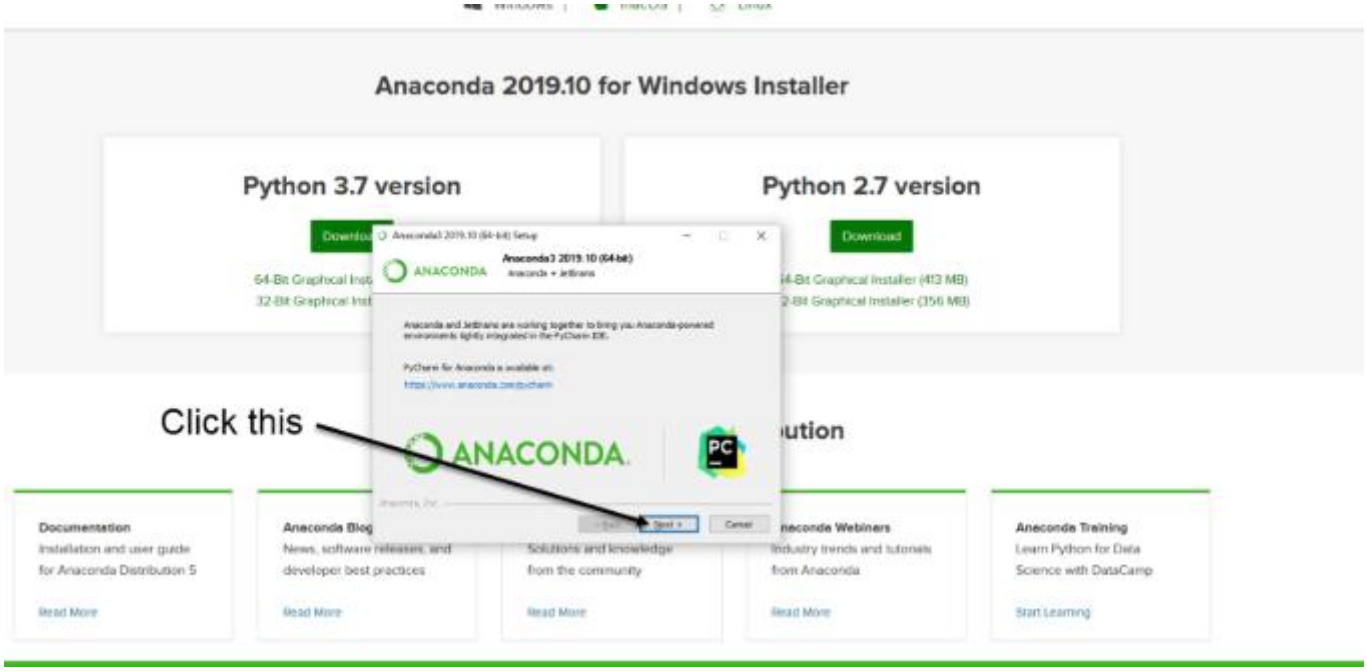
7. It will unpack some packages and extract some files on your machine. This will take a few minutes.



8. The installation is complete. Click Next.

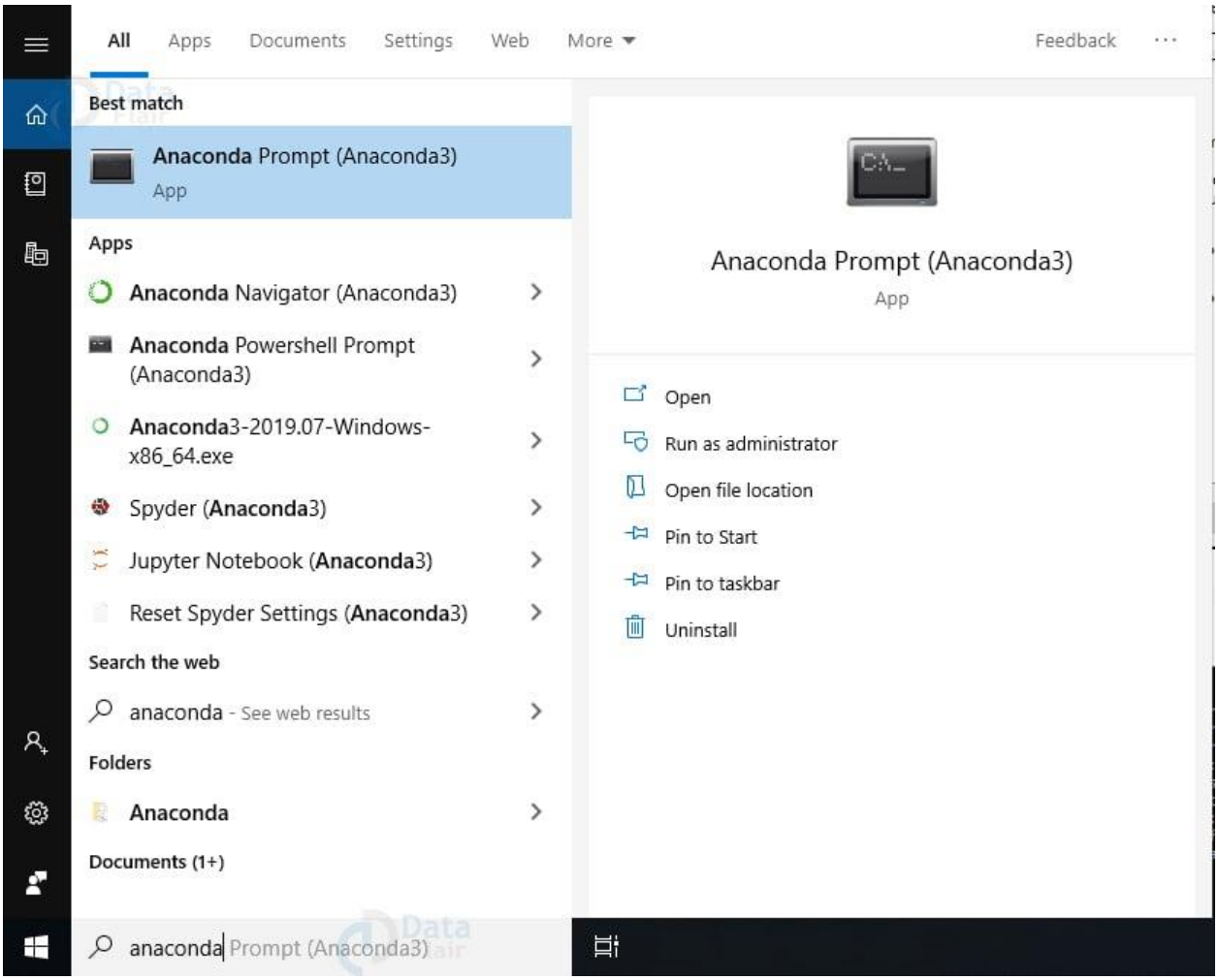


9. This screen will inform you about PyCharm. Click Next.



10. The installation is complete. You can choose to get more information about Anaconda cloud and how to get started with Anaconda. Click Finish.

11. If you search for Anaconda now, you will see the following options:



PYTHON LANGUAGE:

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, we don't need to declare the type of variable because it is a dynamically typed language. For example, x = 10 Here, x can be anything such as String, int, etc.

Features in Python:

There are many features in Python, some of which are discussed below as follows:

1. **Free and Open Source**

Python language is freely available at the official website and you can download it from the given download link below click on the Download Python keyword. Download Python Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

2. **Easy to code**

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

3. **Easy to Read**

As you will see, learning Python is quite simple. As was already established, Python's syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

4. **Object-Oriented Language**

One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.

5. **GUI Programming Support**

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.

6. **High-Level Language**

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

7. **Extensible feature**

Python is an Extensible language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

8. **Easy to Debug**

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program's issues once you understand how to interpret Python's error traces. Simply by glancing at the code, you can determine what it is designed to perform.

9. **Python is a Portable language**

Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

10. **Python is an integrated language**

Python is also an integrated language because we can easily integrate Python with other languages like C, C++, etc.

11. **Interpreted Language:**

Python is an Interpreted Language because Python code is executed line by line at a time. like other

languages C, C++, Java, etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called byte code.

12. Large Standard Library

Python has a large standard library that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as regular expressions, unit-testing, web browsers, etc.

13. Dynamically Typed Language

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

14. Frontend and backend development

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in Python like JavaScript. Backend is the strong forte of Python it's extensively used for this work cause of its frameworks like Django and Flask.

15. Allocating Memory Dynamically

In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value. Developers do not need to write int y = 18 if the integer value 15 is set to y. You may just type y=18.

LIBRARIES/PACKGES:-

Tensor flow

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

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

Numpy

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

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

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

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

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can

accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

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

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

Scikit – learn

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

8. SYSTEM TESTING

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing, for example, might check that every kind of user input produces the intended output across the application.

Phases of system testing:

A video tutorial about this test level. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

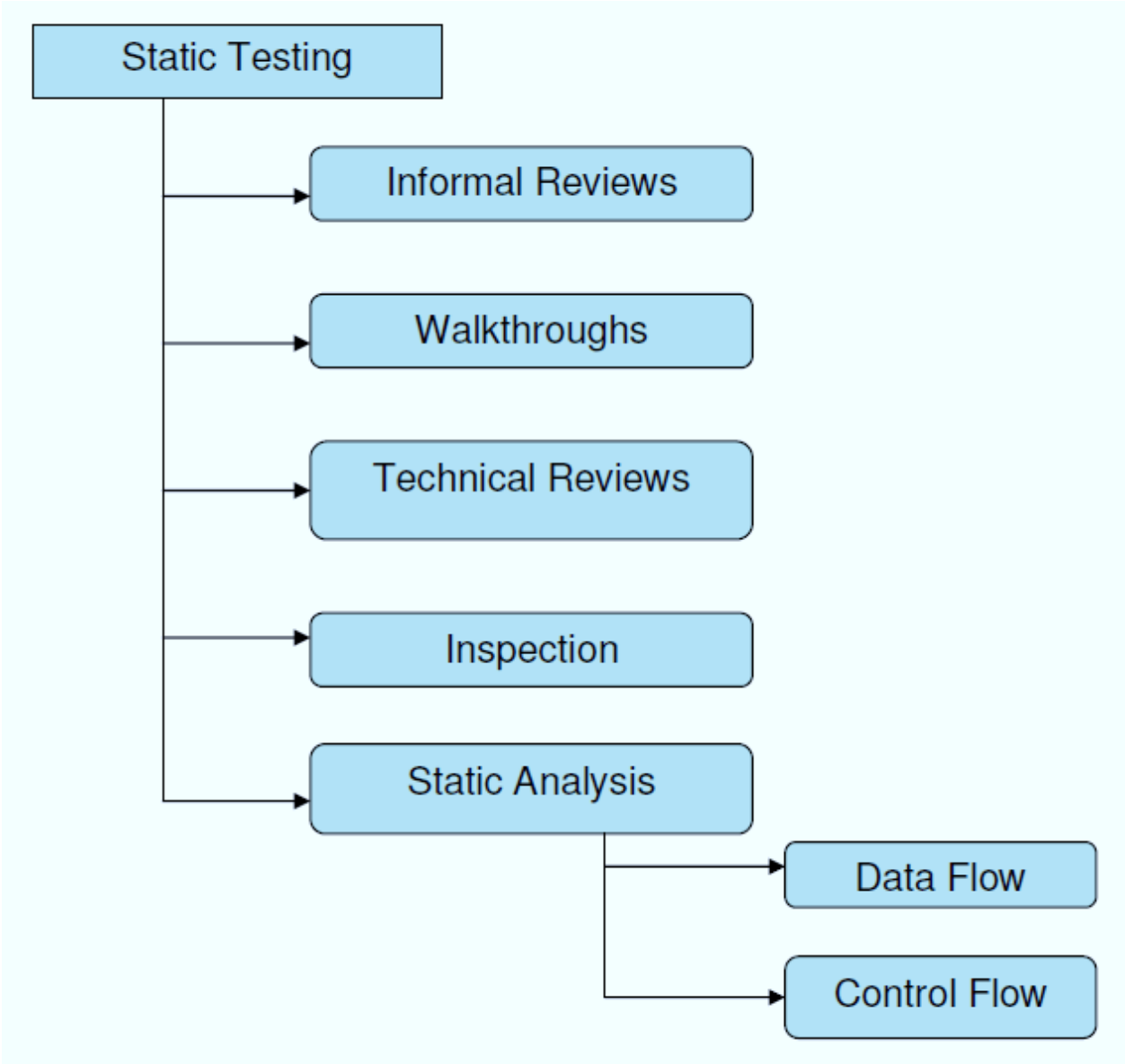
If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

8.1 Software Testing Strategies:

Optimization of the approach to testing in software engineering is the best way to make it effective. A software testing strategy defines what, when, and how to do whatever is necessary to make an end-product of high quality. Usually, the following software testing strategies and their combinations are used to achieve this major objective:

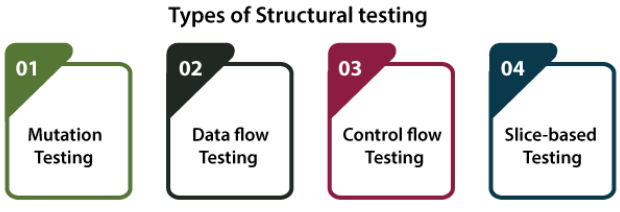
Static Testing:

The early-stage testing strategy is static testing: it is performed without actually running the developing product. Basically, such desk-checking is required to detect bugs and issues that are present in the code itself. Such a check-up is important at the pre-deployment stage as it helps avoid problems caused by errors in the code and software structure deficits.



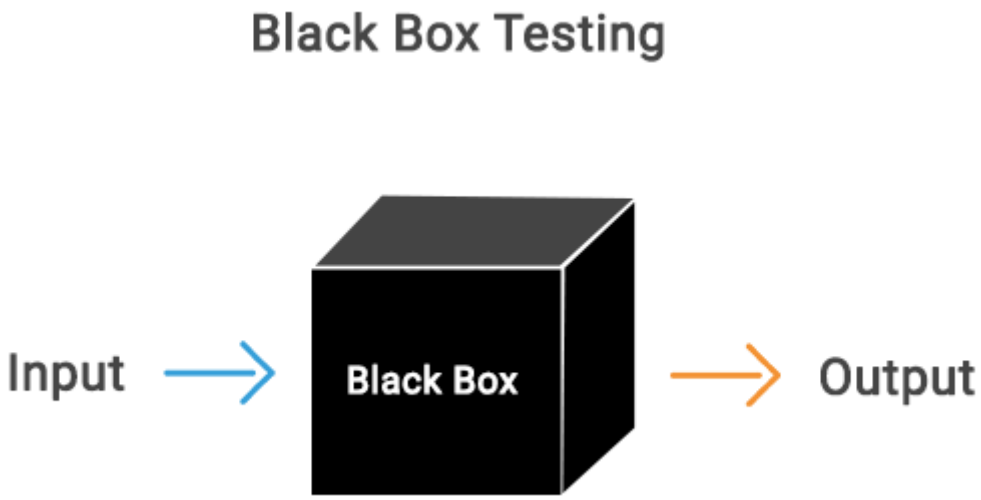
Structural Testing:

It is not possible to effectively test software without running it. Structural testing, also known as white-box testing, is required to detect and fix bugs and errors emerging during the pre-production stage of the software development process. At this stage, unit testing based on the software structure is performed using regression testing. In most cases, it is an automated process working within the test automation framework to speed up the development process at this stage. Developers and QA engineers have full access to the software's structure and data flows (data flows testing), so they could track any changes (mutation testing) in the system's behavior by comparing the tests' outcomes with the results of previous iterations (control flow testing).



Behavioral Testing:

The final stage of testing focuses on the software's reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black-box testing, presupposes running numerous tests, mostly manual, to see the product from the user's point of view. QA engineers usually have some specific information about a business or other purposes of the software ('the black box') to run usability tests, for example, and react to bugs as regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required. For example, you may need to fill 100 registration forms on the website to see how the product copes with such an activity, so the automation of this test is preferable.



8.2 TEST CASES:

S.NO	INPUT	If available	If not available
1	User signup	User get registered into the application	There is no process
2	User sign in	User get login into the application	There is no process
3	Enter input for prediction	Prediction result displayed	There is no process

SCREENS

7. SCREENSHOTS

SCREENS:

10. CONCLUSION

In this paper, an improved YOLO v5 target detection model is proposed for the characteristics of solar cell defects, introducing deformable convolutional CSP module, ECA-Net attention mechanism, improved network structure and adding prediction head to enhance the feature extraction capability to achieve defect detection at different scales. Meanwhile, in order to optimize and improve the model, this paper uses mosaic and MixUp scale fusion data enhancement, K-meansCC clustering anchor box algorithm, and invoking multi-model integration methods. The comparison experiments and ablation experiments show that the improved target detection model achieves an average accuracy of 89.64%, an improvement of 7.85% over the mAP of the original detection model, and a speed of 36.24 FPS, with significant enhancement effects. The next work direction is to reduce the complexity of the model and

achieve high detection speed by processing the detection model network pruning and distillation to achieve a lighter improvement of the model.

BIBILOGRAPHY

REFERENCES

1. S. B. Jha and R. F. Babiceanu, ‘‘Deep CNN-based visual defect detection: Survey of current literature,’’ Comput. Ind., vol. 148, Jun. 2023, Art. no. 103911, doi: 10.1016/j.compind.2023.103911.
2. L. Liu, C. Shen, and A. Van Den Hengel, ‘‘Cross-convolutional-layer pooling for image recognition,’’ 2015, arXiv:1510.00921.
3. N. Liu, L. Wan, Y. Zhang, T. Zhou, H. Huo, and T. Fang, ‘‘Exploiting convolutional neural networks with deeply local description for remote sensing image classification,’’ IEEE Access, vol. 6, pp. 11215–11228, 2018, doi: 10.1109/ACCESS.2018.2798799.
4. A. Hassan and A. Mahmood, ‘‘Convolutional recurrent deep learning model for sentence classification,’’ IEEE Access, vol. 6, pp. 13949–13957, 2018, doi: 10.1109/ACCESS.2018.2814818.
5. X. Ren, Y. Zhou, Z. Huang, J. Sun, X. Yang, and K. Chen, ‘‘A novel text structure feature extractor for Chinese scene text detection and recognition,’’ IEEE Access, vol. 5, pp. 3193–3204, 2017, doi: 10.1109/ACCESS.2017.2676158.
6. W. Liu, A. Dragomir, E. Dumitru, S. Christian, R. Scott, C.-Y. Fu, and C. B. Alexander, ‘‘SSD: Single shot MultiBox detector,’’ in Computer Vision—ECCV. Cham, Switzerland: Springer, 2016, pp. 21–37.
7. S. Ren, K. He, R. Girshick, and J. Sun, ‘‘Faster R-CNN: Towards real-time object detection with region proposal networks,’’ IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.
8. C. V. Dung and L. D. Anh, ‘‘Autonomous concrete crack detection using deep fully convolutional neural network,’’ Autom. Construct., vol. 99, pp. 52–58, Mar. 2019, doi: 10.1016/j.autcon.2018.11.028.
9. G. H. Yao and X. C. Wu, ‘‘Halcon-based solar panel crack detection,’’ in Proc. IEEE WCMEIM Shanghai, China, Nov. 2019, pp. 733–736.
10. V. S. Bharath, A. Haque, and M. A. Khan, ‘‘Fault classification for photovoltaic modules using thermography and image processing,’’ in Proc. IEEE Ind. Appl. Soc. Annu. Meeting, Baltimore, MD, USA, Sep. 2019, pp. 1–6.
11. H. Chen, H. Zhao, D. Han, W. Liu, P. Chen, and K. Liu, ‘‘Structureaware-based crack defect detection for multicrystalline solar cells,’’ Measurement, vol. 151, Feb. 2020, Art. no. 107170, doi: 10.1016/j.measurement.2019.107170.
12. J. Balzategui, L. Eciolaza, and N. Arana-Arexolaleiba, ‘‘Defect detection on polycrystalline solar cells using electroluminescence and fully convolutional neural networks,’’ in Proc. IEEE SII Honolulu, HI, USA, Jan. 2020, pp. 949–953.
13. A. V. Zubair, B. Yoann, D. Priya, S. Arcot, T. Thorsten, and H. Ziv, ‘‘Localization of defects in solar cells using luminescence images and deep learning,’’ in Proc. IEEE PVSC, Fort Lauderdale, FL, USA, Jun. 2021, pp. 745–749.
14. Y. Wang, L. Li, Y. Sun, J. Xu, Y. Jia, J. Hong, X. Hu, G. Weng, X. Luo, S. Chen, Z. Zhu, J. Chu, and H. Akiyama, ‘‘Adaptive automatic solar cell defect detection and classification based on

absolute electroluminescence imaging,'' Energy, vol. 229, Aug. 2021, Art. no. 120606, doi: 10.1016/j.energy.2021.120606.
15. A. S. Al-Waisy, D. A. Ibrahim, D. A. Zebari, S. Hammadi, H. Mohammed, M. A. Mohammed, and R. Damasevicius, ''Identifying defective solar cells in electroluminescence images using deep feature representations,'' PeerJ Comput. Sci., vol. 8, p. e992, May 2022, doi: 10.7717/peerj-cs.992.
16. M. Zhang and L. Yin, ''Solar cell surface defect detection based on improved YOLO v5,'' IEEE Access, vol. 10, pp. 80804–80815, 2022, doi: 10.1109/ACCESS.2022.3195901.
17. L. Li, Z. Wang, and T. Zhang, ''GBH-YOLOv5: Ghost convolution with BottleneckCSP and tiny target prediction head incorporating YOLOv5 for PV panel defect detection,'' Electronics, vol. 12, no. 3, p. 561, Jan. 2023, doi: 10.3390/electronics12030561.
18. S. Prabhakaran, R. A. Uthra, and J. Preetharoselyn, ''Deep learningbased model for defect detection and localization on photovoltaic panels,'' Comput. Syst. Sci. Eng., vol. 44, no. 3, pp. 2683–2700, 2023, doi: 10.32604/csse.2023.028898.
19. A. Chen, X. Li, H. Jing, C. Hong, and M. Li, ''Anomaly detection algorithm for photovoltaic cells based on lightweight multi-channel spatial attention mechanism,'' Energies, vol. 16, no. 4, p. 1619, Feb. 2023, doi: 10.3390/en16041619.
20. M. Bie, Y. Liu, G. Li, J. Hong, and J. Li, ''Real-time vehicle detection algorithm based on a lightweight you-only-look-once (YOLOv5n-L) approach,'' Exp. Syst. Appl., vol. 213, Mar. 2023, Art. no. 119108, doi: 10.1016/j.eswa.2022.119108.
21. T.-Y. Lin, D. Piotr, G. Ross, K. M. He, H. Bharath, and B. Serge, ''Feature pyramid networks for object detection,'' in Proc. IEEE CVPR, Honolulu, HI, USA, Jul. 2017, pp. 936–944.
22. S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, ''Path aggregation network for instance segmentation,'' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Honolulu, HI, USA, Jun. 2018, pp. 8759–8768.
23. D. Zou, Y. Cao, Y. Li, and Q. Gu, ''The benefits of mixup for feature learning,'' 2023, arXiv:2303.08433.
24. G. Song, Y. Liu, and X. Wang, ''Revisiting the sibling head in object detector,'' in Proc. IEEE/CVF sConf. Comput. Vis. Pattern Recognit. (CVPR), Seattle, WA, USA, Jun. 2020, pp. 11560–11569.