**YOLO-Extract Improved YOLOv5 for Aircraft Object Detection in Remote Sensing Images**

**ABSTRACT**

**ABSTRACT**

Remote sensing targets include small and dense target forms, as well as complicated target backgrounds, as compared to natural images. As a consequence, detection accuracy is insufficient and the target location cannot be exactly discovered.We proposes a new algorithm called YOLO-extract for detecting aircraft objects in remote sensing images. The YOLO-extract algorithm is based on the YOLOv5 algorithm and optimizes its model structure to improve detection accuracy and identify target location accurately. The algorithm integrates a new feature extractor with stronger feature extraction ability into the network and borrows the idea of residual network to integrate Coordinate Attention into the network. The mixed dilated convolution is combined with the redesigned residual structure to enhance the feature and location information extraction ability of the shallow layer of the model and optimize the feature extraction ability of the model for different scale targets. Finally, Focal-α EIoU Loss is designed to replace CIoU Loss, which makes the model bounding box regression faster and the loss lower. The experimental results on the test data set show that compared with the YOLOv5 algorithm, the YOLO-extract algorithm has a faster convergence speed, reduces the calculation amount and the number of parameters, but increases the mAP and the detection speed by 3 times.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| S.NO | CONTENT | PGNO |
| 1 | Introduction   * 1. Objective   2. Problem Statement   3. Software requirements   4. Hardware requirements |  |
| 2 | Feasibility study |  |
| 3 | Literature survey |  |
| 4 | System analysis  4.1 Existing system  4.1.1 Disadvantages of existing system  4.2 Proposed system  4.2.1 Advantages of proposed system  4.3 Functional requirements  4.4 Non-Functional requirements |  |
| 5 | System design  5.1 System architecture  5.2 UML diagrams |  |
| 6 | Implementation  6.1 Modules  6.2 Sample code |  |
| 7 | Software environment |  |
| 8 | System testing  8.1 Testing strategies  8.2 Test cases |  |
| 9 | Screens |  |
| 10 | Conclusion |  |
| 11 | References |  |

**INTRODUCTION**

**1. INTRODUCTION**

The massive adoption of the Internet and its convenient use as a connectivity medium has driven a significant evolution of communication networks. Consequently, networks have turned into heterogeneous, dynamic, and systematically complex systems. Thus, designing, deploying, managing, and maintaining networks based on traditional techniques is notoriously difficult to perform. one of the most important advantages of machine learning (ML) is the capability to address complex problems that require classification, regression, and decision-making with results close to or even better than those obtained by human beings. ML techniques have gained a greater maturity in specific domains such as computer vision or natural language understanding. In others, such as communication networks, the application of ML approaches is still at an early stage. Revising the state of the art reveals the lack of publicly available and rich network traffic datasets for the network research community. The selection and adequate representation of network traffic features, which is related to feature exploration and engineering, determine the effectiveness of ML to a large extent. Thus, considering the most relevant features for the inference process will guarantee a proper model performance.

**1.1 Objective:**

The objective of this research is to enhance the accuracy and efficiency of aircraft object detection in remote sensing images. This is achieved by proposing and implementing the YOLO-extract algorithm, an optimized version of YOLOv5. The algorithm focuses on improving detection accuracy and precise target location identification in the presence of small and dense target forms and complex backgrounds commonly encountered in remote sensing imagery.

**1.2 Problem Statement:**

Remote sensing applications often face challenges in accurately detecting aircraft objects due to their small size and intricate backgrounds. Current detection methods like YOLOv5 struggle to provide sufficient accuracy and location precision. To address this issue, our research introduces the YOLO-extract algorithm, which optimizes YOLOv5 by incorporating a stronger feature extractor, Coordinate Attention, mixed dilated convolution, and Focal-α EIoU Loss. The problem is to enhance detection accuracy and speed in remote sensing images while reducing computational complexity and parameters, ultimately achieving a 3-fold improvement in mAP and detection speed compared to YOLOv5.

**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**

**FEASIBILITY STUDY**

**2. FEASIBILITY STUDY**

**Feasibility Study**

A feasibility study evaluates a project's or system's practicality. As part of a feasibility study, the objective and rational analysis of a potential business or venture is conducted to determine its strengths and weaknesses, potential opportunities and threats, resources required to carry out, and ultimate success prospects. Two criteria should be considered when judging feasibility: the required cost and expected value.

**Types Of Feasibility Study**

A feasibility analysis evaluates the project’s potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions. There are five types of feasibility study—separate areas that a feasibility study examines, described below.

**1. Technical Feasibility**

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. As an exaggerated example, an organization wouldn’t want to try to put Star Trek’s transporters in their building—currently, this project is not technically feasible.

**2. Economic Feasibility**

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent [project assessment](https://www.simplilearn.com/risk-assessment-project-management-article) and enhances project credibility—helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide.

### **3. Legal Feasibility**

This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws,[data protection](https://www.simplilearn.com/understanding-data-security-rar30-article) acts or social media laws. Let’s say an organization wants to construct a new office building in a specific location. A feasibility study might reveal the organization’s ideal location isn’t zoned for that type of business. That organization has just saved considerable time and effort by learning that their project was not feasible right from the beginning.

### **4. Operational Feasibility**

This assessment involves undertaking a study to analyze and determine whether—and how well—the organization’s needs can be met by completing the project. Operational feasibility studies also examine how a [project plan](https://www.simplilearn.com/project-management-plans-in-project-environment-rar79-article) satisfies the requirements identified in the requirements analysis phase of system development.

### **5. Scheduling Feasibility**

This assessment is the most important for [project success](https://www.simplilearn.com/how-to-make-a-project-successful-article); after all, a project will fail if not completed on time. In scheduling feasibility, an organization estimates how much time the project will take to complete.

When these areas have all been examined, the feasibility analysis helps identify any constraints the proposed project may face, including:

* Internal Project Constraints: Technical, Technology, Budget, Resource, etc.
* Internal Corporate Constraints: Financial, Marketing, Export, etc.
* External Constraints: Logistics, Environment, Laws, and Regulations, etc.

**SYSTEM ANALYSIS**

**4.SYSTEM ANALYSIS**

**4.1 EXISTING SYSTEM:**

In previous studies they introduced a new model for detecting remote sensing targets at different scales. The model is based on You Only Look Once (YOLO)-V3 and uses DenseNet (Densely Connected Network) to enhance feature extraction capability. The detection scales were increased to four based on the original YOLO-V3. The experiment are performed on RSOD (Remote Sensing Object Detection) dataset and UCS-AOD (Dataset of Object Detection in Aerial Images) datasets. In another study they introduced the YOLOv5, the smooth Kullback-Leibler divergence loss function was used to replace the cross entropy loss function, and the CS and Glass module was designed to replace the residual module.

**4.1.1 DISADVANTAGES OF EXISTING SYSTEM:**

* Compared with natural images, remote sensing targets have small and dense target shapes as well as complex target backgrounds.
* As a result, insufficient detection accuracy and target location cannot be accurately identified.
* YOLO-V3 with DenseNet may not have incorporated the latest advancements or updates made to the YOLOv3 model, as it seems to be a previous iteration of the YOLO model.
* YOLOv5's feature extraction might be limited in capturing complex patterns in remote sensing images.
* YOLOv5 might not utilize the Coordinate Attention mechanism, which might result in less precise focus on important spatial locations in the input images, affecting target localization accuracy.
* Although YOLOv5 introduces the smooth Kullback-Leibler divergence loss function to replace the cross-entropy loss, it might not be as effective in bounding box regression.

# 4.2 Proposed System:

We proposes a new algorithm called YOLO-extract for detecting aircraft objects in remote sensing images based on the YOLOv5 algorithm. The YOLO-extract algorithm is based on the YOLOv5 algorithm and optimizes its model structure to improve detection accuracy and identify target location accurately. The algorithm integrates a new feature extractor with stronger feature extraction ability into the network and borrows the idea of residual network to integrate Coordinate Attention mechanism into the network. The mixed dilated convolution is combined with the redesigned residual structure to enhance the feature and location information extraction ability of the shallow layer of the model and optimize the feature extraction ability of the model for different scale targets. Finally, Focal-α EIoU Loss is designed to replace CIoU Loss, which makes the model bounding box regression faster and the loss lower.

# 4.2.1 Advantages of proposed system:

1. YOLO-extract optimizes the model structure of YOLOv5, which can lead to improved detection accuracy. The modifications made in YOLO-extract may enhance the overall performance of the model compared to the existing work.
2. YOLO-extract integrates a new feature extractor with stronger feature extraction capabilities. This enhancement can better capture complex patterns and characteristics of remote sensing images, making the model more effective in detecting aircraft objects.
3. By incorporating the Coordinate Attention mechanism into the network, YOLO-extract can better focus on important spatial locations in the input images, helping to improve the precision and accuracy of target location.
4. The combination of mixed dilated convolution with the redesigned residual structure improves feature and location information extraction in the shallow layers of the model.
5. The design of Focal-α EIoU Loss to replace CIoU Loss helps in faster bounding box regression during training and reduces the overall loss.

### **4.3 FUNCTIONAL REQUIREMENTS**

1.Data Collection

2.Data Preprocessing

3.Training And Testing

4.Modiling

5.Predicting

### **4.4 NON FUNCTIONAL REQUIREMENTS**

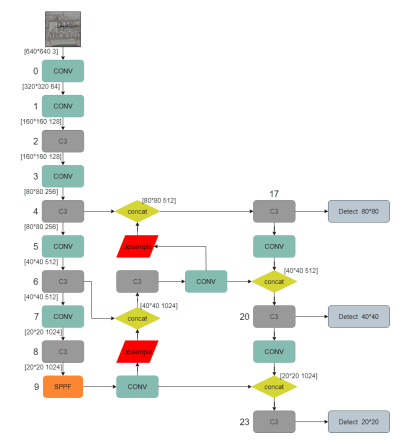
NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement,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

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

**SYSTEM DESIGN**

**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.

**Import libraries**

**VERIFY**

**NO PROCESS**

**Yes NO**

**Exploring the dataset**

**Image processing**

**Loading the pretrained model**

**Image processing**

**Data Augmentation**

**Building the model – YOLOV5 – YOLOV6 – YOLOV7 – YOLOV8**

**Training the model**

**Signup & signin**

**User input**

**Final outcome**

**End process**

**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.

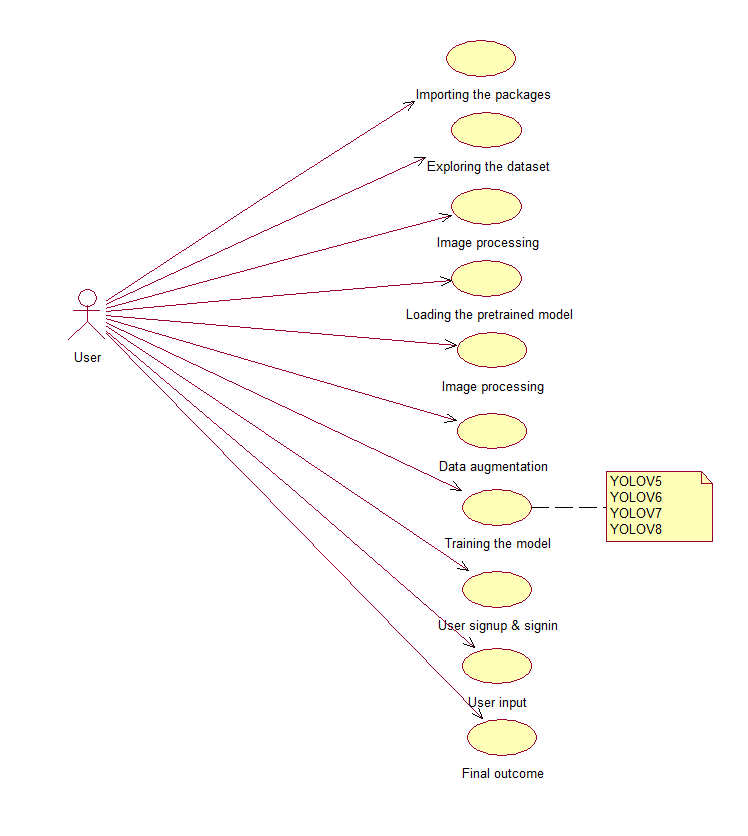
**GOALS:**

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.

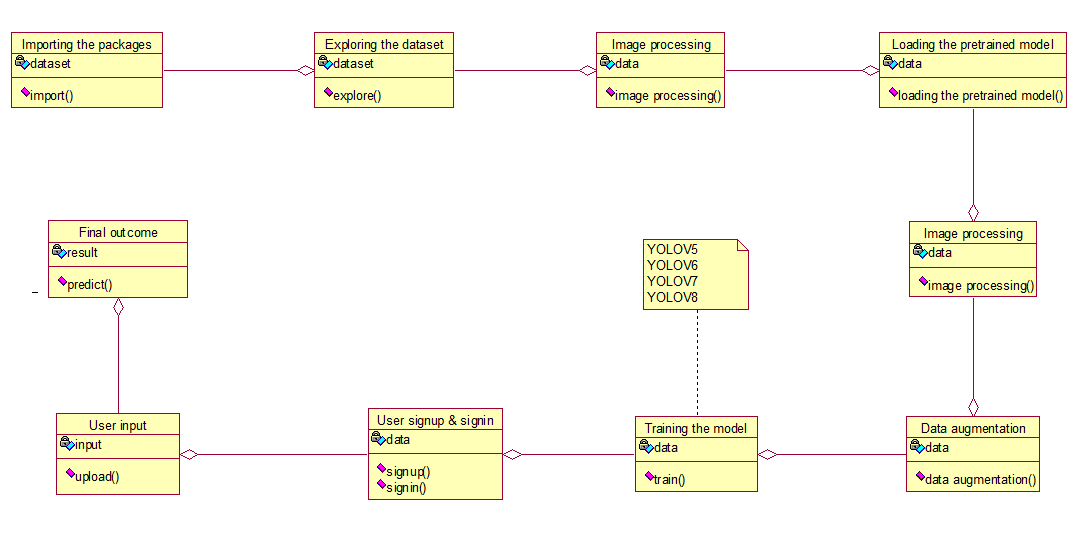
**Use case diagram:**

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.



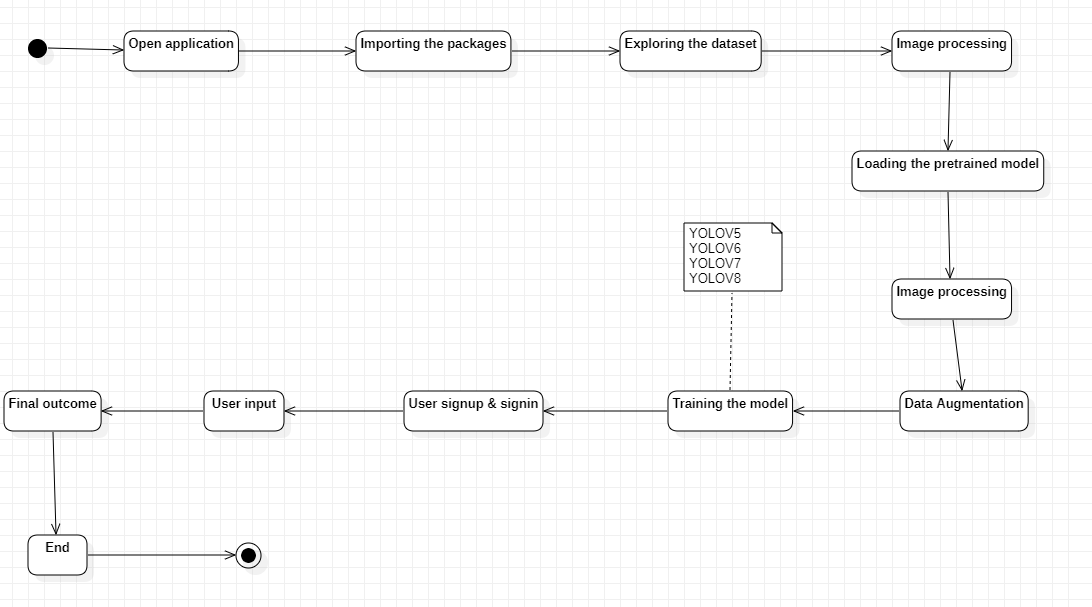
**Class diagram:**

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.



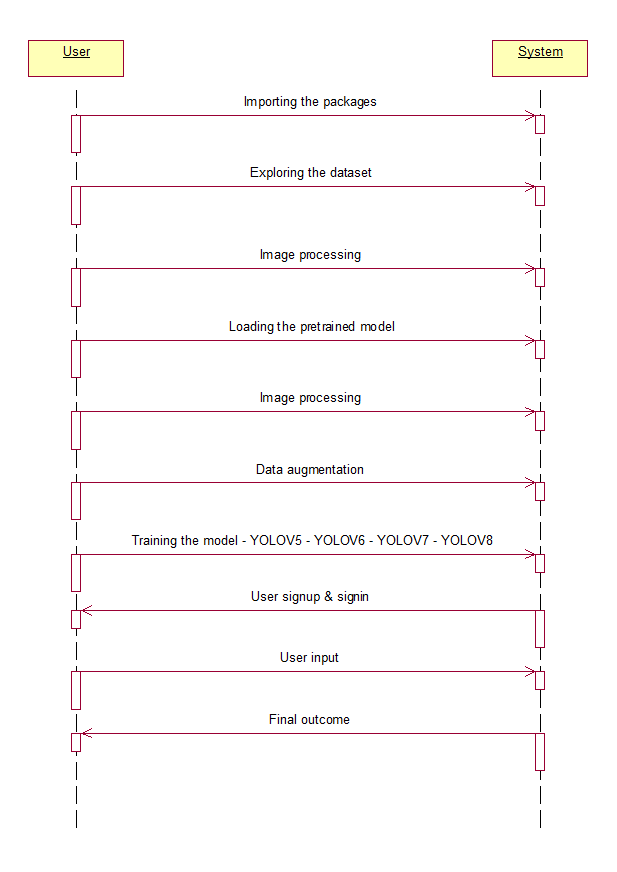
**Activity diagram:**

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.

****

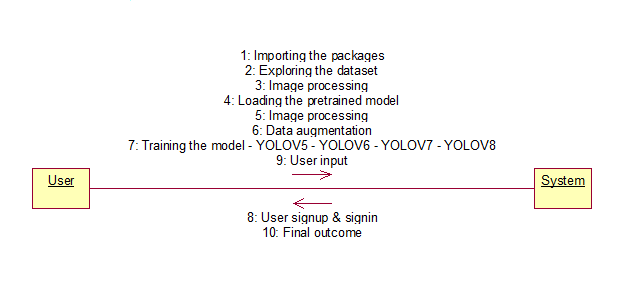
**Sequence diagram:**

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".



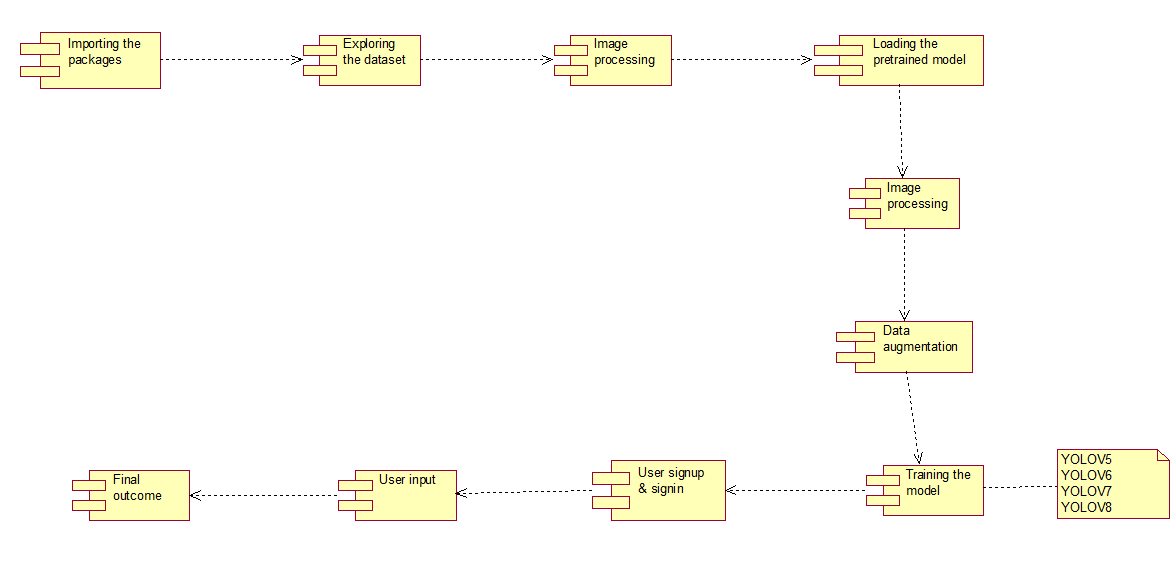
**Collaboration diagram:**

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.

****

**IMPLEMENTATION**

1. **IMPLEMENTATION**

MODULES:

* **Data loading:** using this module we are going to import the dataset.

**Dataset Link:** <https://universe.roboflow.com/object-detection/dota-v2.0>

**Dataset Description:** The dataset includes 1822 images

* **Data Preprocessing:** using this module we will explore the data.
* **Splitting data into train & test:** using this module data will be divided into train & test
* **Model generation:** Model building – YOLOV5 – YOLOV6 – YOLOV7 – YOLOV8. Algorithms accuracy calculated
* **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

**Algorithms:**

**YOLOV5:**

YOLOv5 is a model in the You Only Look Once (YOLO) family of computer vision models. YOLOv5 is commonly used for detecting objects. YOLOv5 comes in four main versions: small (s), medium (m), large (l), and extra large (x), each offering progressively higher accuracy rates. Each variant also takes a different amount of time to train.

**YOLOV6:**

Mt-YOLOv6, or simply YOLOv6, is a single-stage object detection algorithm. A single-stage object detection model performs object localization and image classification within the same network. Object localization involves identifying the position of a single object (or multiple objects) within an image frame.

**YOLOV7:**

YOLOv7 algorithms can be used to recognize and track objects as they move through a production line, allowing for more efficient and accurate manufacturing. Additionally, object detection is used for quality control and defect detection in products or components as they are being manufactured.

**YOLOV8:**

YOLOv8 is a new state-of-the-art computer vision model built by Ultralytics, the creators of YOLOv5. The YOLOv8 model contains out-of-the-box support for object detection, classification, and segmentation tasks, accessible through a Python package as well as a command line interface.

**6.2 SAMPLE CODE:**

import torch

from yolov5 import utils

display = utils.notebook\_init()  # checks

from PIL import Image, ImageFile

ImageFile.LOAD\_TRUNCATED\_IMAGES = True

from matplotlib import pyplot as plt

import numpy as np

import rasterio

from rasterio.windows import Window

from rasterio.plot import show

import math

import numpy as np

# Model

model = torch.hub.load('ultralytics/yolov5', 'yolov5x6', pretrained=True)  # or yolov5m, yolov5l, yolov5x, custom

# Image

img = '/kaggle/input/ortofotos-2017-rgb/331/RGB-2017/ORTO\_3314-451\_RGB\_1000\_v2.jp2'

dataset = rasterio.open(img)

square\_size = 240

math.ceil(dataset.width / square\_size), math.ceil(dataset.height / square\_size)

%matplotlib inline

for i in np.arange(math.ceil(dataset.width / square\_size)):

    for j in np.arange(math.ceil(dataset.height / square\_size)):

# for i in [2]:

#     for j in [14]:

        data = dataset.read(window=Window(square\_size \* i, square\_size \* j, square\_size, square\_size))

#         data = grayscale[square\_size \* i:square\_size \* (i+1), square\_size \* j:square\_size \* (j+1)].reshape(3, 240, 240)

        results = model(data)

#         print(i, j)

#         print(data.shape)

#         results.print()

        if len(results.xywh[0]) > 0:

            plt.imshow(np.squeeze(results.render()))

            plt.show()

**SOFTWARE ENVIRONMENT**

**7. SOFTWARE ENVIRONMENT**

**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.

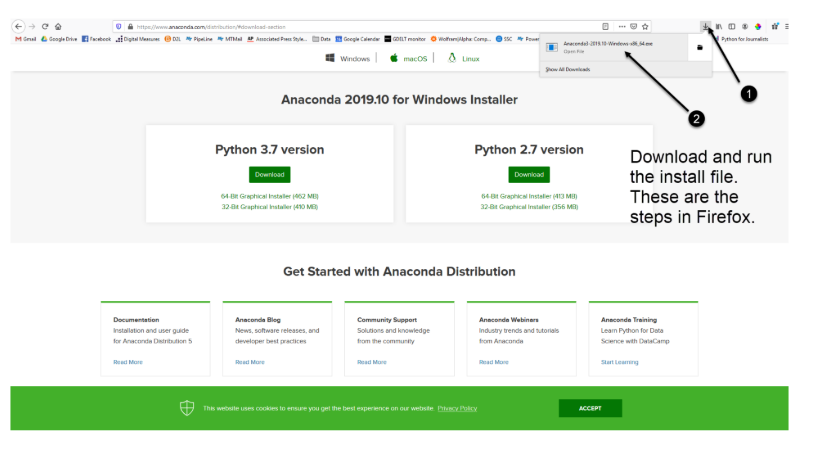
Follow the prompts and agree to the terms and conditions. When you are asked if you want to "add Anaconda to my PATH environment variable," make sure that you select "yes." This will ensure that Anaconda is added to your system's PATH, which is a list of directories that your operating system uses to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

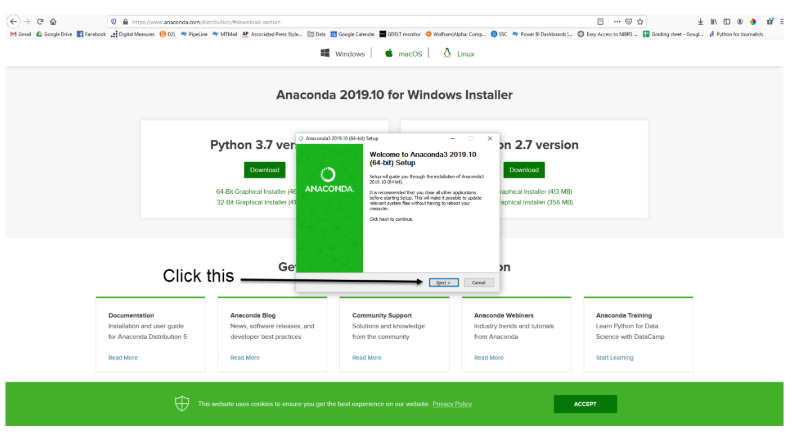
Next in the Python anaconda tutorial is its installation. The latest version of Anaconda at the time of writing is 2019.10. Follow these steps to download and install Anaconda on your machine:

1. Go to this link and download Anaconda for Windows, Mac, or Linux: – [Download anaconda](https://www.anaconda.com/distribution/)

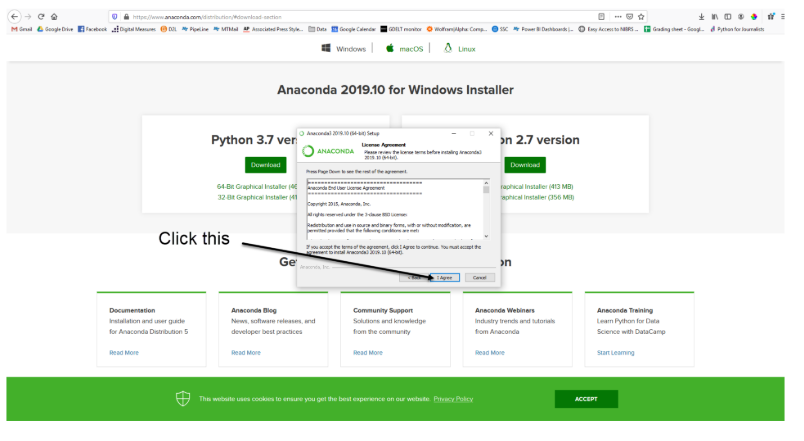


You can download the installer for Python 3.7 or for Python 2.7 (at the time of writing). And you can download it for a 32-bit or 64-bit machine.

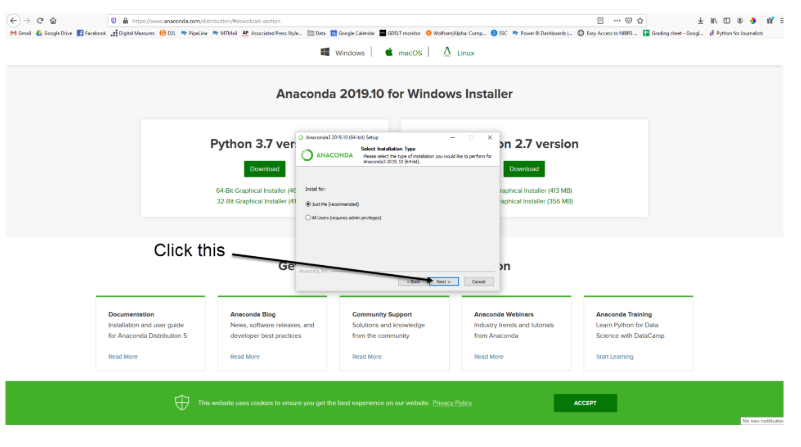
2. Click on the downloaded .exe to open it. This is the Anaconda setup. Click next.



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



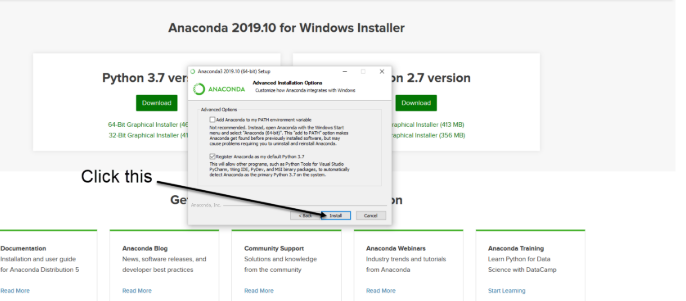
4. You can install it for all users or just for yourself. If you want to install it for all users, you need administrator privileges.



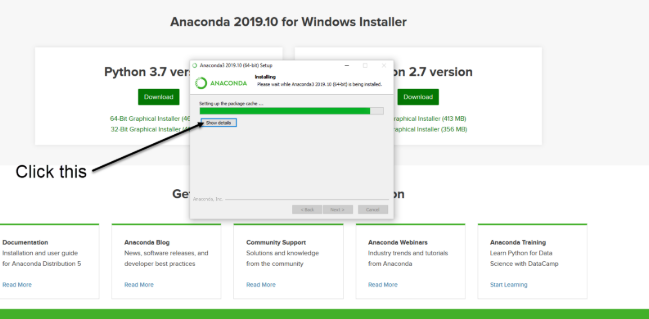
5. Choose where you want to install it. Here, you can see the available space and how much you need.



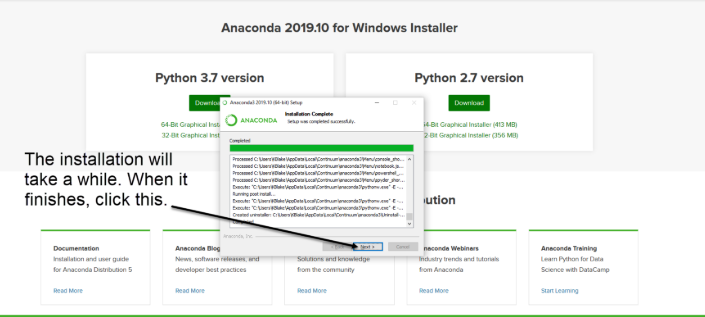
6. Now, you’ll get some advanced options. You can add Anaconda to your system’s PATH environment variable, and register it as the primary system Python 3.7. If you add it to PATH, it will be found before any other installation. Click on ‘Install’.



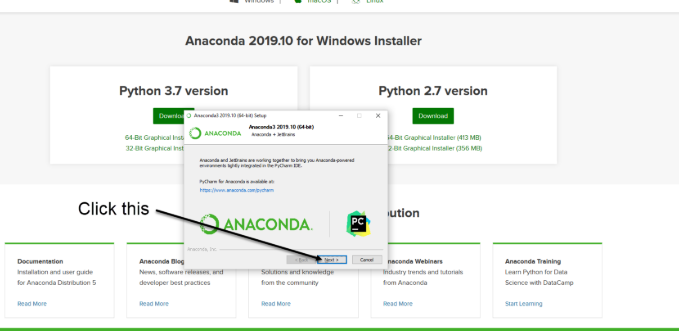
7. It will unpack some packages and extract some files on your machine. This will take a few minutes.



8. The installation is complete. Click Next.



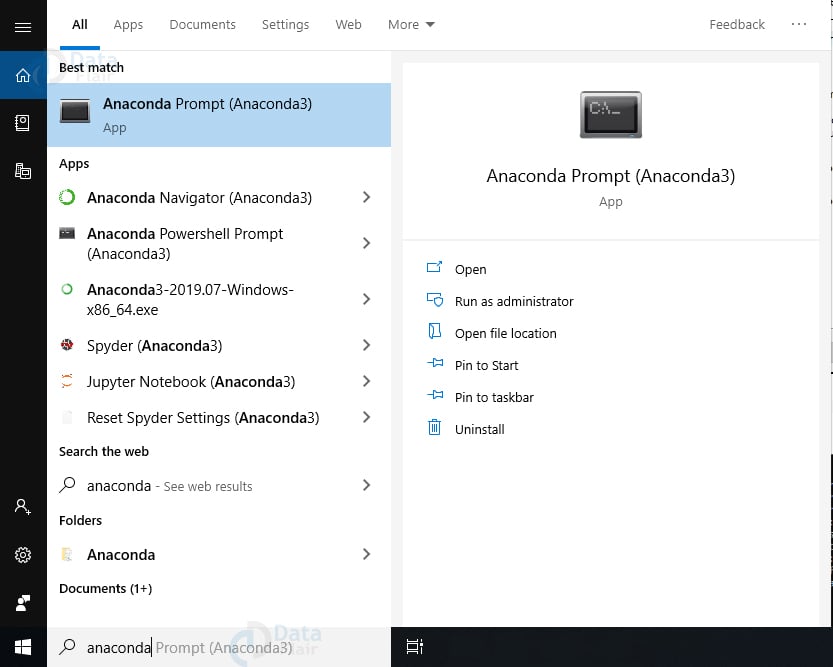
9. This screen will inform you about PyCharm. Click Next.



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 interpreted, 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](https://www.geeksforgeeks.org/python-programming-language/)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](https://www.python.org/downloads/) 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](https://www.geeksforgeeks.org/difference-between-high-level-and-low-level-languages/). 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](https://www.geeksforgeeks.org/python-oops-concepts/). 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](https://www.geeksforgeeks.org/pyqt5-qaction/), PyQt4, wxPython, or [Tk in python](https://www.geeksforgeeks.org/python-gui-tkinter/). 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](https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/)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](https://www.geeksforgeeks.org/introduction-to-linux-operating-system/), 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++](http://www.geeksforgeeks.org/c-plus-plus/), 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](https://www.geeksforgeeks.org/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 **bytecode**.

### **12. Large Standard Library**

Python has a large [standard library](https://www.geeksforgeeks.org/libraries-in-python/) 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 expression](https://www.geeksforgeeks.org/regular-expression-python-examples-set-1/)s, [unit-testing](https://www.geeksforgeeks.org/unit-testing-software-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**

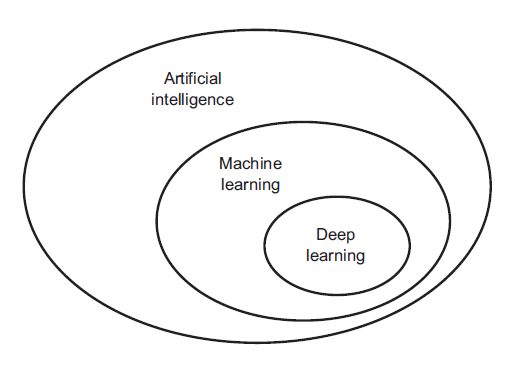
With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in Python like javascript. Backend is the strong forte of Python it’s extensively used for this work cause of its frameworks like [Django](https://www.geeksforgeeks.org/django-tutorial/)and [Flask](https://www.geeksforgeeks.org/flask-creating-first-simple-application/).

### **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.

**What is deep learning?**

Deep learning is a subset of [machine learning](https://www.ibm.com/topics/machine-learning), which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

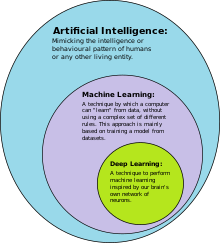


Deep learning drives many [artificial intelligence (AI)](https://www.ibm.com/topics/artificial-intelligence) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

**How deep learning works**

Deep learning neural networks, or artificial neural networks, attempts to mimic the human brain through a combination of data inputs, weights, and bias. These elements work together to accurately recognize, classify, and describe objects within the data.

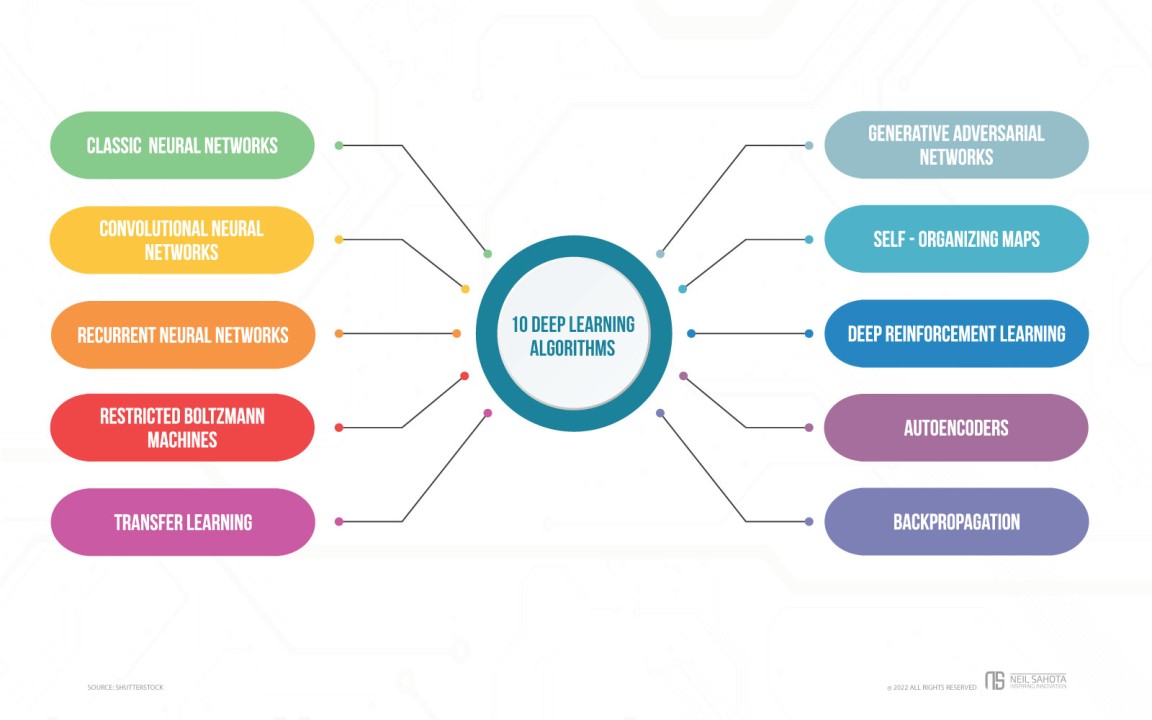
Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called *visible*layers. The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made.



Another process called backpropagationuses algorithms, like gradient descent, to calculate errors in predictions and then adjusts the weights and biases of the function by moving backwards through the layers in an effort to train the model. Together, forward propagation and backpropagation allow a neural network to make predictions and correct for any errors accordingly. Over time, the algorithm becomes gradually more accurate.

The above describes the simplest type of deep neural network in the simplest terms. However, deep learning algorithms are incredibly complex, and there are different types of neural networks to address specific problems or datasets. For example,

* [*Convolutional neural networks (CNNs),*](https://www.ibm.com/topics/convolutional-neural-networks)used primarily in computer vision and image classification applications, can detect features and patterns within an image, enabling tasks, like object detection or recognition. In 2015, a CNN bested a human in an object recognition challenge for the first time.
* [*Recurrent neural network (RNNs)*](https://www.ibm.com/topics/recurrent-neural-networks)are typically used in natural language and speech recognition applications as it leverages sequential or times series data.



## Applications for deep learning

Applications that utilize deep learning are already integrated into our daily lives and have uses in many different industries. [Generative AI](https://www.redhat.com/en/topics/cloud-computing/what-is-generative-ai), which now powers many AI tools, is made possible through deep learning.

The use cases for deep learning are forever evolving, but 3 of the most popular technologies being utilized today are computer vision, speech recognition, and natural language processing (NLP).



* **Computer vision:**Computers can use deep learning techniques to comprehend images the same way humans do. This means automated content moderation, facial recognition, and image classification.
* **Speech recognition:**Pitch, tone, language, and accent can all be analyzed and by way of deep learning models. Not only can this be used to improve customer experience, but it is also helpful from an accessibility standpoint in cases that require real-time transcription.
* **Natural language processing (NLP):**Computers use deep learning algorithms to analyze and gather insights from text data and documents. This can aid in the function of summarizing long documents, indexing key phrases that indicate sentiment (such as positive or negative comments), and generating insight for automated virtual assistants and chatbots.

**Deep learning Applications examples**

Real-world deep learning applications are a part of our daily lives, but in most cases, they are so well-integrated into products and services that users are unaware of the complex data processing that is taking place in the background. Some of these examples include the following:

**Law enforcement**

Deep learning algorithms can analyze and learn from transactional data to identify dangerous patterns that indicate possible fraudulent or criminal activity. Speech recognition, computer vision, and other deep learning applications can improve the efficiency and effectiveness of investigative analysis by extracting patterns and evidence from sound and video recordings, images, and documents, which helps law enforcement analyze large amounts of data more quickly and accurately.

**Financial services**

Financial institutions regularly use predictive analytics to drive algorithmic trading of stocks, assess business risks for loan approvals, detect fraud, and help manage credit and investment portfolios for clients.

**Customer service**

Many organizations incorporate deep learning technology into their customer service processes. [Chatbots](https://www.ibm.com/topics/chatbots)—used in a variety of applications, services, and customer service portals—are a straightforward form of AI. Traditional chatbots use natural language and even visual recognition, commonly found in call center-like menus. However, more [sophisticated chatbot solutions](https://www.ibm.com/products/watson-assistant) attempt to determine, through learning, if there are multiple responses to ambiguous questions. Based on the responses it receives, the chatbot then tries to answer these questions directly or route the conversation to a human user.

Virtual assistants like Apple's Siri, Amazon Alexa, or Google Assistant extends the idea of a chatbot by enabling speech recognition functionality. This creates a new method to engage users in a personalized way.

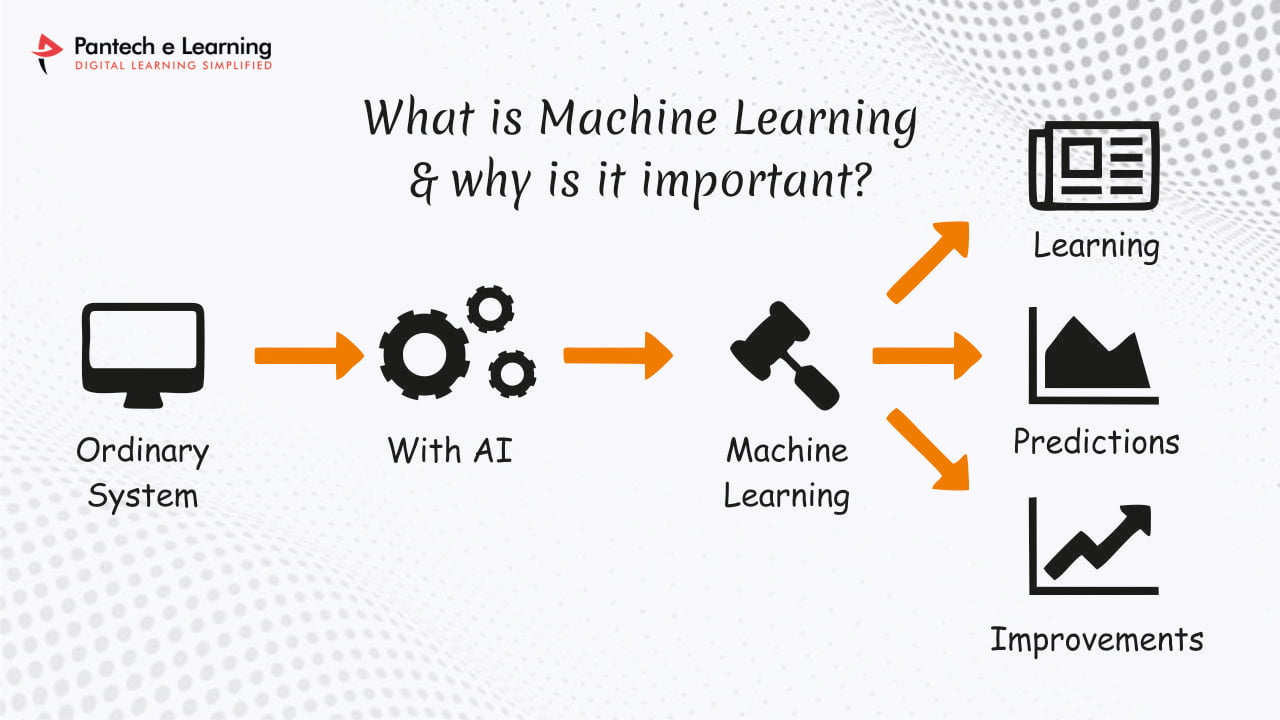
**Healthcare**

The healthcare industry has benefited greatly from deep learning capabilities ever since the digitization of hospital records and images. Image recognition applications can support medical imaging specialists and radiologists, helping them analyze and assess more images in less time.

## Importance of Deep Learning

Deep learning algorithms play a crucial role in determining the features and can handle the large number of processes for the data that might be structured or unstructured. Although, deep learning algorithms can overkill some tasks that might involve complex problems because they need access to huge amounts of data so that they can function effectively. For example, there's a popular deep learning tool that recognizes images namely **Imagenet** that has access to **14 million** images in its dataset-driven algorithms. It is a highly comprehensive tool that has defined a next-level benchmark for deep learning tools that aim images as their dataset.

Deep learning algorithms are highly progressive algorithms that learn about the image that we discussed previously by passing it through each neural network layer. The layers are highly sensitive to detect low-level features of the image like **edges** and **pixels** and henceforth the combined layers take this information and form holistic representations by comparing it with previous data. For example, the middle layer might be programmed to detect some special parts of the object in the photograph which other deep trained layers are programmed to detect special objects like **dogs, trees, utensils,** etc.



However, if we talk out the simple task that involves less complexity and a data-driven resource, deep learning algorithms fail to generalize simple data. This is one of the main reasons deep learning is not considered effective as **linear** or **boosted tree models.** Simple models aim to churn out custom data, track fraudulent transactions and deal with less complex datasets with fewer features. Also, there are various cases like **multiclass classification** where deep learning can be effective because it involves smaller but more structured datasets but is not preferred usually.

## **Challenges in Deep Learning**

Deep learning has made significant advancements in various fields, but there are still some challenges that need to be addressed. Here are some of the main challenges in deep learning:

1. Data availability: It requires large amounts of data to learn from. For using deep learning it’s a big concern to gather as much data for training.
2. Computational Resources: For training the deep learning model, it is computationally expensive because it requires specialized hardware like GPUs and TPUs.
3. Time-consuming: While working on sequential data depending on the computational resource it can take very large even in days or months.
4. Interpretability: Deep learning models are complex, it works like a black box. it is very difficult to interpret the result.
5. Overfitting: when the model is trained again and again, it becomes too specialized for the training data, leading to overfitting and poor performance on new data.

### Advantages of Deep Learning:

1. High accuracy: Deep Learning algorithms can achieve state-of-the-art performance in various tasks, such as image recognition and natural language processing.
2. Automated feature engineering: Deep Learning algorithms can automatically discover and learn relevant features from data without the need for manual feature engineering.
3. Scalability: Deep Learning models can scale to handle large and complex datasets, and can learn from massive amounts of data.
4. Flexibility: Deep Learning models can be applied to a wide range of tasks and can handle various types of data, such as images, text, and speech.
5. Continual improvement: Deep Learning models can continually improve their performance as more data becomes available.

### Disadvantages of Deep Learning:

1. High computational requirements: Deep Learning models require large amounts of data and computational resources to train and optimize.
2. Requires large amounts of labeled data: Deep Learning models often require a large amount of labeled data for training, which can be expensive and time- consuming to acquire.
3. Interpretability: Deep Learning models can be challenging to interpret, making it difficult to understand how they make decisions.  
   Overfitting: Deep Learning models can sometimes overfit to the training data, resulting in poor performance on new and unseen data.
4. Black-box nature: Deep Learning models are often treated as black boxes, making it difficult to understand how they work and how they arrived at their predictions.  
   In summary, while Deep Learning offers many advantages, including high accuracy and scalability, it also has some disadvantages, such as high computational requirements, the need for large amounts of labeled data, and interpretability challenges. These limitations need to be carefully considered when deciding whether to use Deep Learning for a specific task.

**LIBRARIES/PACKGES :-**

**Tensorflow**

TensorFlow is a [free](https://en.wikipedia.org/wiki/Free_software) and [open-source](https://en.wikipedia.org/wiki/Open-source_software) [software library for dataflow and differentiable programming](https://en.wikipedia.org/wiki/Library_(computing)) across a range of tasks. It is a symbolic math library, and is also used for [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications such as [neural networks](https://en.wikipedia.org/wiki/Neural_networks). It is used for both research and production at [Google](https://en.wikipedia.org/wiki/Google).‍

TensorFlow was developed by the [Google Brain](https://en.wikipedia.org/wiki/Google_Brain) team for internal Google use. It was released under the [Apache 2.0](https://en.wikipedia.org/wiki/Apache_License) [open-source license](https://en.wikipedia.org/wiki/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](http://ipython.org/) shells, the [Jupyter](http://jupyter.org/) 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](https://matplotlib.org/tutorials/introductory/sample_plots.html) and [thumbnail gallery](https://matplotlib.org/gallery/index.html).

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

**Scikit – learn**

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

**SYSTEM TESTING**

**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.

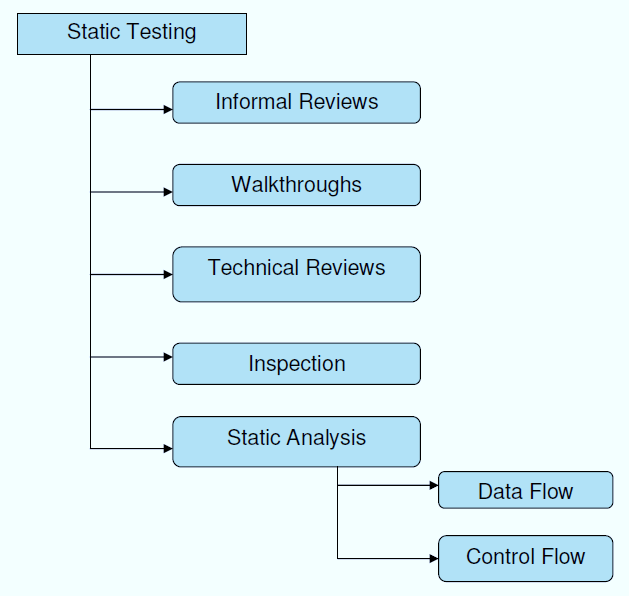
If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

**8.1Software 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.



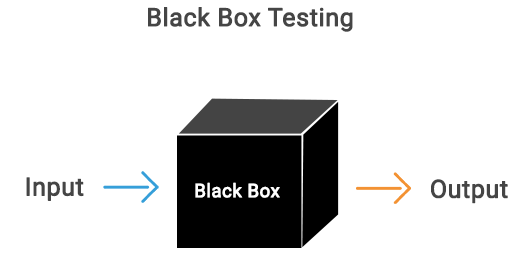
**Structural Testing:**

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.



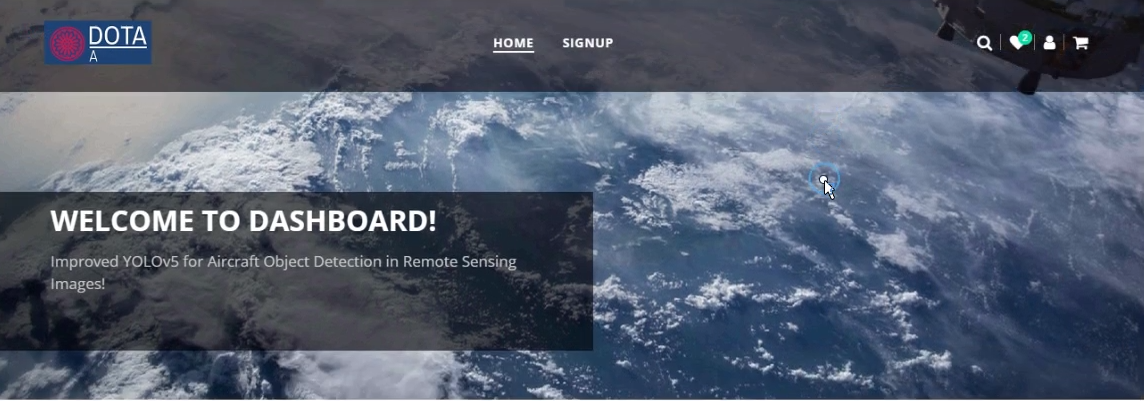
**8.2 TEST CASES:**

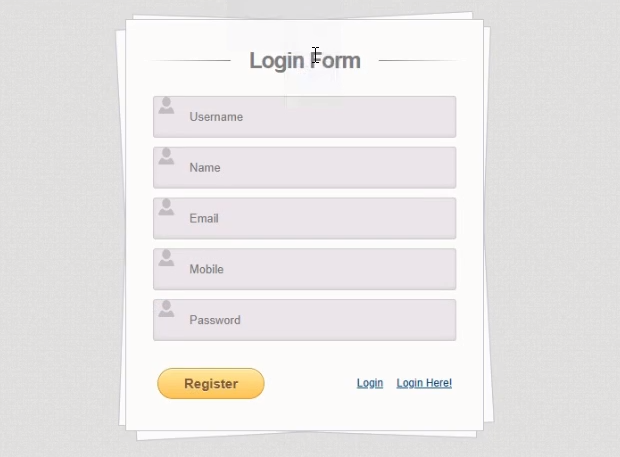
|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **INPUT** | **If available** | **If not available** |
| 1 | User signup | User get registered into the application | There is no process |
| 2 | User signin | User get login into the application | There is no process |
| 3 | Enter input for prediction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

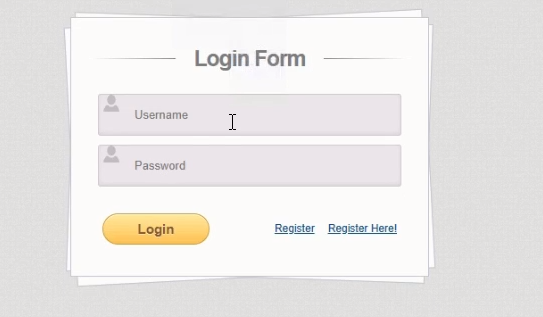
**SCREENS**

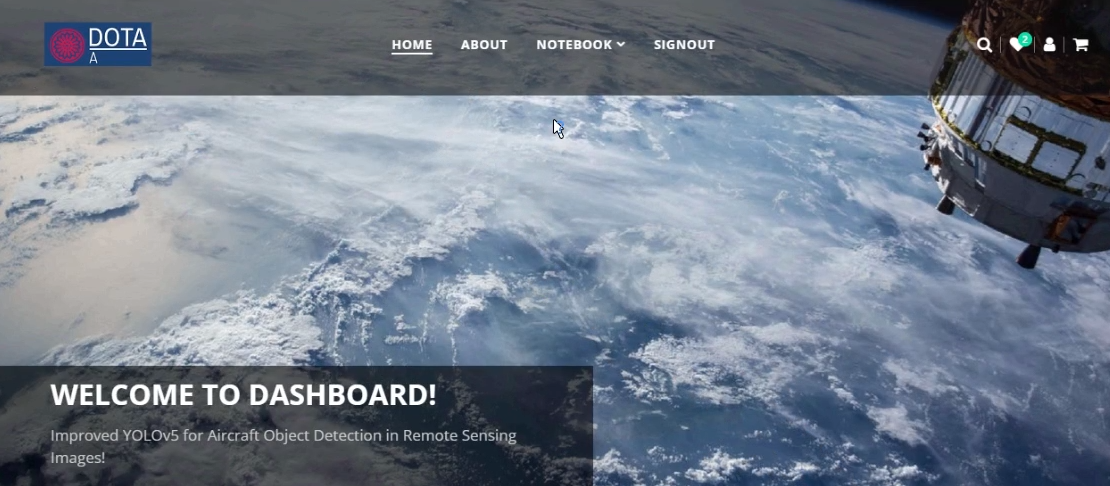
1. **SCREENSHOTS**

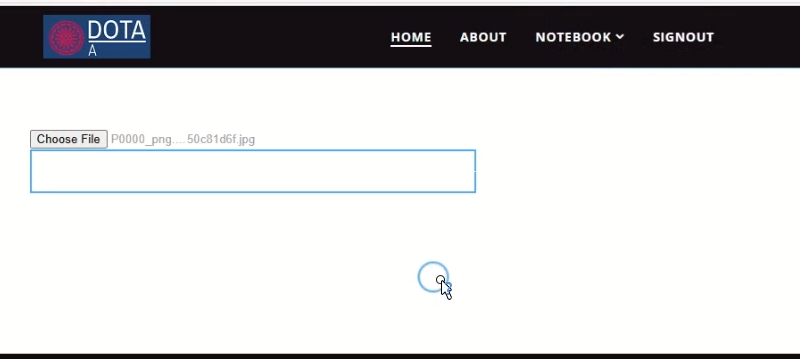
SCREENS:

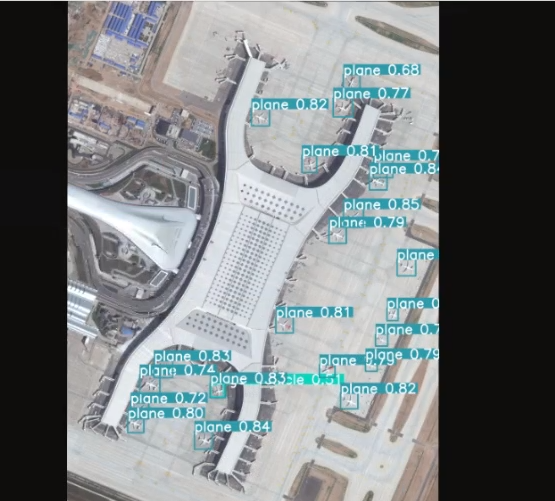


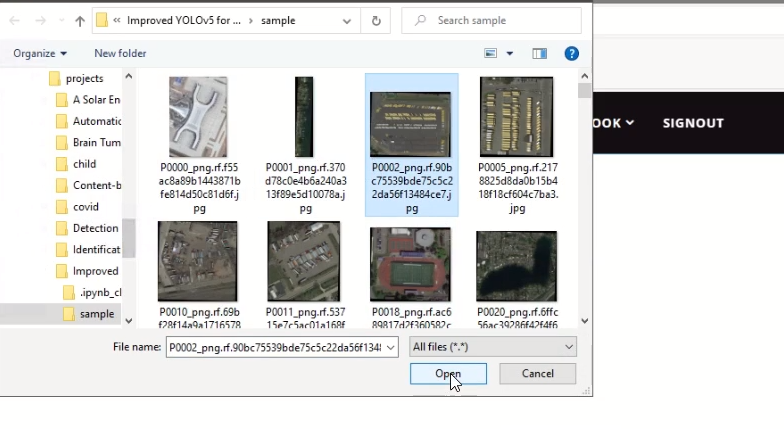


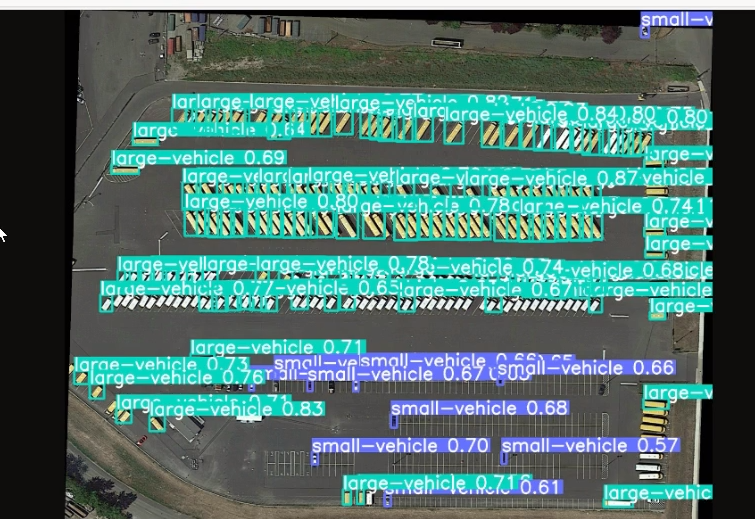












**CONCLUSION**

**10.CONCLUSION**

In this work, We developed the YOLO-extract algorithm based on the YOLOv5 algorithm,which overcome the some of the shortcomings in an YOLOv5 model such as detection accuracy and speed are low, and it is easily affected by the background of the images. Based on the characteristics of aircraft targets on optical remote sensing images, We optimizes the structure of the yolov5 model, introduces dilated convolution to improve the feature extraction capability of the model for aircraft targets, and finally optimizes the convergence speed of the loss function plus block model, as well as improves the detection accuracy and detection speed of the model. Experiments show that the method in this work can greatly improve the ability to overcome the interference of factors such as aircraft attitude and complex background. However, since remote sensing satellite images are easily affected by weather factors such as skylight conditions and clouds and fog, it is difficult to extract different types of aircraft target features in remote sensing images, and there are few data sets for aircraft types, so the detection of aircraft types cannot be completed. In subsequent experiments, relatively clear remote sensing images can be selected to learn and detect different types of aircraft features.

**BIBILOGRAPHY**

**11. REFERENCES**

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, ‘‘Rich feature hierarchies for accurate object detection and semantic segmentation,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2014, pp. 580–587.

[2] R. Girshick, ‘‘Fast R-CNN,’’ in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1440–1448.

[3] S. Ren, K. He, R. Girshick, and J. Sun, ‘‘Faster R-CNN: Towards real-time object detection with region proposal networks,’’ in Proc. Adv. Neural Inf. Process. Syst., vol. 28, 2015, pp. 1–15.

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, ‘‘SSD: Single shot multibox detector,’’ in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2016, pp. 21–37.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, ‘‘You only look once: Unified, real-time object detection,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 779–788.

[6] J. Redmon and A. Farhadi, ‘‘YOLO9000: Better, faster, stronger,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 7263–7271.

[7] J. Redmon and A. Farhadi, ‘‘YOLOv3: An incremental improvement,’’ 2018, arXiv:1804.02767.

[8] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, ‘‘YOLOv4: Optimal speed and accuracy of object detection,’’ 2020, arXiv:2004.10934.

[9] D. Xu and Y. Wu, ‘‘Improved YOLO-V3 with DenseNet for multiscale remote sensing target detection,’’ Sensors, vol. 20, no. 15, p. 4276. 2020.

[10] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, ‘‘Densely connected convolutional networks,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 4700–4708.

[11] D. Xu and Y. Wu, ‘‘FE-YOLO: A feature enhancement network for remote sensing target detection,’’ Remote Sens., vol. 13, no. 7, p. 1311, Mar. 2021.

[12] S. H. Gao, M. M. Cheng, and K. Zhao, ‘‘Res2Net: A new multi-scale backbone architecture,’’ IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 2, pp. 652–662, Feb. 2021.

[13] C. Cao, J. Wu, X. Zeng, Z. Feng, T. Wang, X. Yan, Z. Wu, Q. Wu, and Z. Huang, ‘‘Research on airplane and ship detection of aerial remote sensing images based on convolutional neural network,’’ Sensors, vol. 20, no. 17, p. 4696, Aug. 2020.

[14] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, ‘‘Pyramid scene parsing network,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 2881–2890.

[15] S. Luo, J. Yu, Y. Xi, and X. Liao, ‘‘Aircraft target detection in remote sensing images based on improved YOLOv5,’’ IEEE Access, vol. 10, pp. 5184–5192, 2022.

[16] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, ‘‘Feature pyramid networks for object detection,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 2117–2125.

[17] H. Li, P. Xiong, J. An, and L. Wang, ‘‘Pyramid attention network for semantic segmentation,’’ 2018, arXiv:1805.10180.

[18] Q. Hou, D. Zhou, and J. Feng, ‘‘Coordinate attention for efficient mobile network design,’’ in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2021, pp. 13713–13722.

[19] F. Yu and V. Koltun, ‘‘Multi-scale context aggregation by dilated convolutions,’’ 2015, arXiv:1511.07122.

[20] K. He, X. Zhang, S. Ren, and J. Sun, ‘‘Deep residual learning for image recognition,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[21] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, ‘‘Distance-IoU loss: Faster and better learning for bounding box regression,’’ in Proc. AAAI Conf. Artif. Intell., 2020, vol. 34, no. 7, pp. 12993–13000.

[22] Y.-F. Zhang, W. Ren, Z. Zhang, Z. Jia, L. Wang, and T. Tan, ‘‘Focal and efficient IOU loss for accurate bounding box regression,’’ Neurocomputing, vol. 506, pp. 146–157, Sep. 2022.

[23] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, ‘‘Focal loss for dense object detection,’’ in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 2980–2988.

[24] J. He, S. Erfani, X. Ma, J. Bailey, Y. Chi, and X.-S. Hua, ‘‘alphaIoU: A family of power intersection over union losses for bounding box regression,’’ in Proc. Adv. Neural Inf. Process. Syst., vol. 34, 2021, pp. 20230–20242.

[25] C.-Y. Wang, A. Bochkovskiy, and H.-Y. Mark Liao, ‘‘YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,’’ 2022, arXiv:2207.02696.