**Currency Detection For Visually Impaired**

**ABSTRACT**

Indian currency notes, issued by Indian government, are involved in monetary transactions across India. The paper aims to provide an efficient method for detection of Indian currency notes, which is an application of object detection using deep learning. The detection of currency notes aids visually impaired people who can’t recognize the monetary value in their possession. Hence, an effective model is designed in which currency notes are detected and human speech of detected currency note is provided. The notes are first trained using YOLOV5 model and then, the results are validated against validation data. The evaluation metrics are observed and model losses are minimized. Then, a new test data is considered as inference for recognition of classes. The web app is designed using YOLOV5, Flask and web technologies. The web app detected Indian currency notes with bounding box probability greater than 0.80 for all the classes. The labels of each note are detected with bounding box probability greater than 0.90 and labels are spoken out clearly using this designed model.

**INDEX**

**ABSTRACT**

**1. INTRODUCTION**

**2. SYSTEM ANALYSIS**

2.1 Existing System

2.2 Proposed System

2.3 Feasibility Study

**3. SYSTEM REQUIREMENTS SPECIFICATION**

3.1 Functional Requirements

3.2 Non Functional Requirements

3.3 Input & Output Design

3.4 Systems Requirement and Specification

**4. SYSTEM DESIGN**

4.1 DFD’s

4.2 UML Diagrams

4.3 Data Dictionaries and ER Diagram

**5. IMPLEMENTATION**

5.1 Module & Description

5.2 Technology Description

5.3. Coding

**6. SYSTEM TESTING**

**CONCLUSION**

**REFERENCES**

**CHAPTER-1**

**INTRODUCTION**

I**NTRODUCTION**

Object Detection is technique used in detection of real-time objects, in images and videos using either machine learning or deep learning and OpenCV. Object detection is applied in surveillance, image retrieval, security, Advanced Driver Assistance Systems (ADAS). Object detection includes image classification and object localization [1]. Based on WHO statistics, at least 2.2 billion people have a near or distance vision impairment. The leading causes for visual impairment include uncorrected refractive errors, cataract, diabetic retinopathy, glaucoma and age-related vision impairment [2] In 2017, National Center for Biotechnology Information (NCBI) reported that 253 million people globally were visually impaired among which 217 million had visualimpairment and remaining were identified as blind [3]. Visually impaired lack the ability to interpret bank-notes is very limited and usually depend on others. The aim is to overcome this difficulty and to provide easier currency detection for visually impaired. The currency notes are detected based on feature extraction, considering geometric size and characteristic texture [4]. In India, Indian Currency Recognition System currently uses CNN and OpenCV based recognition system to detect monetary notes [5]. This model has lower detection accuracy than the proposed system which uses YOLOv5 for object detection. The proposed system involves YOLOv5 algorithm to detect test images. The model is trained using transfer learning and is later inferenced against test data. A web-based application is designed using YOLOv5 and Flask. This app provides labels on the screen as well as audio file of detected currency note, when a currency note image is uploaded. The audio file can be listened to using play button provided in the web app. This will not only aid visually impaired, but also, provide easier way to recognize Indian currency notes to tourists and foreigners visiting India. The challenge is to design a web application based on YOLOv5 model to detect Indian currency notes accurately and provide optimized and efficient results, both on laptop and mobile.

**Literature review**

**S. Luo and W. Zheng, “You-Only-Look-Once-v5 Based Table Detection for Academic Papers,” 2021 Int. Conf. Digit. Soc. Intell. Syst. DSInS 2021, pp. 53–56, 2021, doi: 10.1109/DSInS54396.2021.9670600.**

Shenmei Luo and Wenbin Zheng (2021) proposed a table detection method for academic paper based on YOLOv5 (You Only Look Once v5) where detection accuracy and speed ofproposed approach are observed to be superior compared to other competitive approaches [6].

**S. D. Achar, C. Shankar Singh, C. S. Sumanth Rao, K. Pavana Narayana, and A. Dasare, “Indian Currency Recognition System Using CNN and Comparison with YOLOv5,” 2021 IEEE Int. Conf. Mob. Networks Wirel. Commun. ICMNWC 2021, pp. 1–6, 2021, doi: 10.1109/ICMNWC52512.2021.9688513.**

Sagar et. al (2021) designed a system for detection of Indian currencies that aids visually impaired people to recognize and read out the possible Indian paper currencies with 79.83% accuracy [4]. The proposed model is based on YOLOv5, where it compares the efficiency of YOLOv5 with CNN algorithm for currency detection.

**K. K. S. N. Reddy, C. Yashwanth, S. H. Kvs, P. A. T. V. Sai, and S. Khetarpaul, “Object and Currency Detection with Audio Feedback for Visually Impaired,” 2020 IEEE Reg. 10 Symp. TENSYMP 2020,no. June, pp. 1152–1155, 2020, doi: 10.1109/TENSYMP50017.2020.9230687.**

Sai Nadh et. al (2020) has developed a module for blind people and design mobile application for visually impaired Indian citizens to recognize currency value when held near android phone to predict the currency value so that currency can be recognized correctly [7].

**P. Shreya, N. Shreyas, D. Pushya, and N. Uma Maheswar Reddy, “BLIND ASSIST : A One Stop Mobile Application for the Visually Impaired,” 2021 IEEE Pune Sect. Int. Conf. PuneCon 2021, pp. 1–4, 2021, doi: 10.1109/PuneCon52575.2021.9686476.**

Shreya et. al (2021) designed a mobile application for Visually Impaired Persons – Blind Assist [8] based on web technologies to scan currency note using device’s camera and get its denomination using a voice assistant.

**R. R. M. Et al., “Currency Detection for Visually Impaired Iraqi Banknote as a Study Case,” Turkish J. Comput. Math. Educ., vol. 12, no. 6, pp. 2940–2948, 2021, doi: 10.17762/turcomat.v12i6.6078.**

Raghad Raied Mahmood et. al proposed real-time Iraqi currency note detection system that aids visually impaired people using YOLO-v3 and gTTS to obtain audio output. custom Iraqi banknote dataset for detection and recognition of banknotes. The Iraqi notes were detected in a short time with high mAP of 97.405% [9].

**M. Mahendru and S. K. Dubey, “Real time object detection with audio feedback using Yolo vs. Yolo\_V3,” Proc. Conflu. 2021 11th Int. Conf. Cloud Comput. Data Sci. Eng., pp. 734–740, 2021, doi: 10.1109/Confluence51648.2021.9377064.**

Mansi Mahendru and Sanjay Kumar Dubey (2021) proposed system for visually impaired to detect multiple objects and prompt voice to alert person stating near and farther objects around them using gTTS and TensorFlow [10]. A comparison between YOLO and YOLO\_v3 for object detection was tested under same criteria based on measures like accuracy and model performance.

**CHAPTER-2**

**SYSTEM ANALYSIS**

**2.1 EXISTING SYSTEM**

In This Existing system methodproved tha tthe Mexican banknotes can be classified by extracting their texture features and colour. This technique uses the RGB colour model and the Local Binary Patterns for the identification process. Theaccuracy of this method is very low. Junfang etal. Used an improved LBP algorithm, called block LBP algorithm for characteristic extract. It is based on the ordinary Local Binary Pattern (LBP) method. This method is very simple and has a low speed.

**2.2 Proposed system and advantages**

The proposed system involves YOLOv5 algorithm to detect test images. The model is trained using transfer learning and is later inferenced against test data. A web-based application is designed using YOLOv5 and Flask. This app provides labels on the screen as well as audio file of detected currency note, when a currency note image is uploaded. The audio file can be listened to using play button provided in the web app. This will not only aid visually impaired, but also, provide easier way to recognize Indian currency notes to tourists and foreigners visiting India.

**2.3 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**

**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.

**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.

**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.

**CHAPTER-3**

**SYSTEM REQUIREMENTS SPECIFICATION**

**REQUIREMENT ANALYSIS**

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

**3.1 FUNCTIONAL REQUIREMENTS**

Functional requirement should include function performed by a specific screen outline work-flows performed by the system and other business or compliance requirement the system must meet. Functional requirements specify which output file should be produced from the given file they describe the relationship between the input and output of the system, for each functional requirement a detailed description of all data inputs and their source and the range of valid inputs must be specified. The functional specification describes what the system must do, how the system does it is described in the design specification. If a user requirement specification was written, all requirements outlined in the user requirements specifications should be addressed in the functional requirements.

**3.2 NON FUNCTIONAL REQUIREMENTS**

Describe user-visible aspects of the system that are not directly related with the functional behavior of the system. Non-Functional requirements include quantitative constraints, such as response time (i.e. how fast the system reacts to user commands.) or accuracy (.e. how precise are the systems numerical answers.).

**3.3 INPUT & OUTPUT DESIGN**

**INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**OBJECTIVES**

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

**OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

1. Convey information about past activities, current status or projections of the
2. Future.
3. Signal important events, opportunities, problems, or warnings.
4. Trigger an action.
5. Confirm an action.

**3.4 SYSTEMS REQUIREMENT AND SPECIFICATION**

**SYSTEM REQUIREMENTS:**

**Hardware Requirements:**

* System : INTEL CORE I5
* Hard Disk : 120 GB.
* Ram : 4Gb.

**Software Requirements:**

* Operating system : Windows 7 Ultimate.
* Coding Language : Python.
* Data Base : csv/text

**CHAPTER- 4**

**SYSTEM DESIGN**

6.1. INTRODUCTION

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage. The purpose of the design phase is to plan a solution of the problem specified by the requirement document.

This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affection the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design.

System Design also called top-level design aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of the system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided.

During, Detailed Design, the internal logic of each of the modules specified in system design is decided. During this phase, the details of the data of a module is usually specified in a high-level design description language, which is independent of the target language in which the software will eventually be implemented.

In system design the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for each of the modules. In other works, in system design the attention is on what components are needed, while in detailed design how the components can be implemented in software is the issue.

Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified.

During the system design activities, Developers bridge the gap between the requirements specification, produced during requirements elicitation and analysis, and the system that is delivered to the user.

Design is the place where the quality is fostered in development. Software design is a process through which requirements are translated into a representation of software.

**Data Flow Diagrams:**

A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system’s structure charts. The Basic Notation used to create a DFD’s are as follows:

1. Dataflow: Data move in a specific direction from an origin to a destination.

2. Process: People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.

3. Source: External sources or destination of data, which may be People, programs, organizations or other entities.

4. Data Store: Here data are stored or referenced by a process in the System.

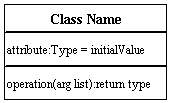
**What is a UML Class Diagram?**

Class diagrams are the backbone of almost every object-oriented method including UML. They describe the static structure of a system.

Basic Class Diagram Symbols and Notations

Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.

Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (centered, bolded, and capitalized), list the attributes in the second partition, and write operations into the third.



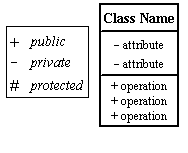
**Active Class**

Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.



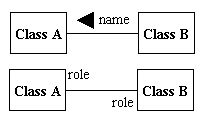
**Visibility**

Use visibility markers to signify who can access the information contained within a class. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class. [.](http://www.smartdraw.com/resources/tutorials/Text-and-Tables)



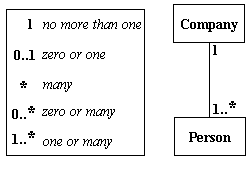
**Associations**

Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.  
Note: It's uncommon to name both the association and the class roles.



**Multiplicity (Cardinality)**

Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for one company only.

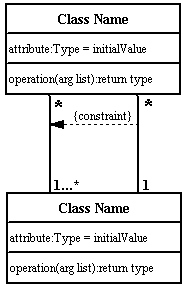


Constraint

Place constraints inside curly braces {}.

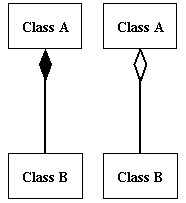
http://wc1.smartdraw.com/resources/tutorials/images/uml_constraint.gif

Simple Constraint



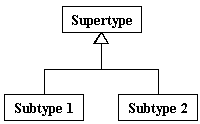
Composition and Aggregation

Composition is a special type of aggregation that denotes a strong ownership between Class A, the whole, and Class B, its part. Illustrate composition with a filled diamond. Use a hollow diamond to represent a simple aggregation relationship, in which the "whole" class plays a more important role than the "part" class, but the two classes are not dependent on each other. The diamond end in both a composition and aggregation relationship points toward the "whole" class or the aggregate



Generalization

Generalization is another name for inheritance or an "is a" relationship. It refers to a relationship between two classes where one class is a specialized version of another. For example, Honda is a type of car. So the class Honda would have a generalization relationship with the class car.



In real life coding examples, the difference between inheritance and aggregation can be confusing. If you have an aggregation relationship, the aggregate (the whole) can access only the PUBLIC functions of the part class. On the other hand, inheritance allows the inheriting class to access both the PUBLIC and PROTECTED functions of the super class.

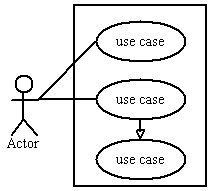
**What is a UML Use Case Diagram?**

Use case diagrams model the functionality of a system using actors and use cases. Use cases are services or functions provided by the system to its users.

Basic Use Case Diagram Symbols and Notations

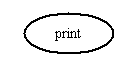
System

Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.



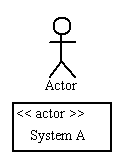
Use Case

Draw use cases using ovals. Label with ovals with verbs that represent the system's functions.



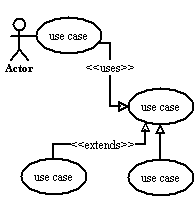
Actors

Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.



Relationships

Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labeled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another in order to perform a task. An "extends" relationship indicates alternative options under a certain use case.



**Sequence Diagram**

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.

Basic Sequence Diagram Symbols and Notations

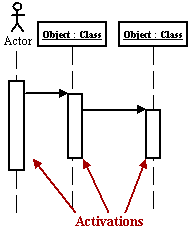
Class roles

Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Class roles

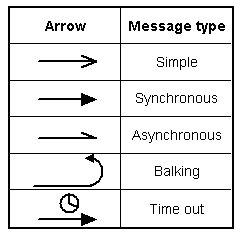
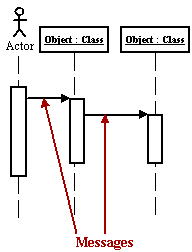
Activation

Activation boxes represent the time an object needs to complete a task.

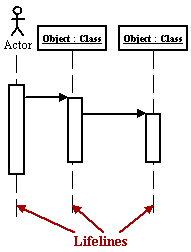


Messages

Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.

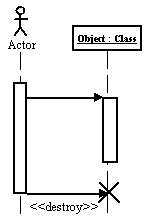
  
Various message types for Sequence and Collaboration diagrams

Lifelines

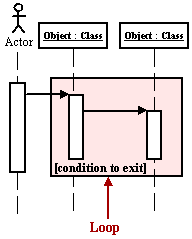
Lifelines are vertical dashed lines that indicate the object's presence over time.  


Destroying Objects

Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an X.



Loops

A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [ ].  


Collaboration Diagram

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system.

Basic Collaboration Diagram Symbols and Notations

Class roles

Class roles describe how objects behave. Use the UML object symbol to illustrate class roles, but don't list object attributes.

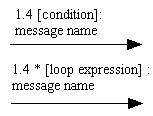
Class roles

Association roles

Association roles describe how an association will behave given a particular situation. You can draw association roles using simple lines labeled with stereotypes.  
Association roles

Messages

Unlike sequence diagrams, collaboration diagrams do not have an explicit way to denote time and instead number messages in order of execution. Sequence numbering can become nested using the Dewey decimal system. For example, nested messages under the first message are labeled 1.1, 1.2, 1.3, and so on. The a condition for a message is usually placed in square brackets immediately following the sequence number. Use a \* after the sequence number to indicate a loop.  
[Learn how to add arrows to your lines.](http://www.smartdraw.com/resources/tutorials/Lines)



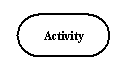
Activity Diagram

An activity diagram illustrates the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation. Because an activity diagram is a special kind of state chart diagram, it uses some of the same modeling conventions.

Basic Activity Diagram Symbols and Notations

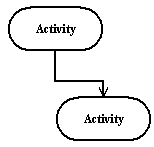
Action states

Action states represent the non interruptible actions of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.



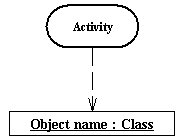
**Action Flow**

Action flow arrows illustrate the relationships among action states.



**Object Flow**

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.



Initial State

A filled circle followed by an arrow represents the initial action state.

Initial State

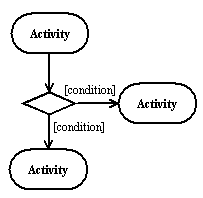
Final State

An arrow pointing to a filled circle nested inside another circle represents the final action state.

Final State

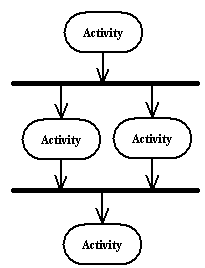
Branching

A diamond represents a decision with alternate paths. The outgoing alternates should be labeled with a condition or guard expression. You can also label one of the paths "else."



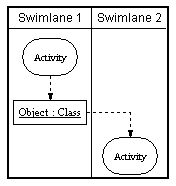
Synchronization

A synchronization bar helps illustrate parallel transitions. Synchronization is also called forking and joining.



Swimlanes

Swimlanes group related activities into one column.



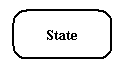
State chart Diagram

A state chart diagram shows the behavior of classes in response to external stimuli. This diagram models the dynamic flow of control from state to state within a system.

Basic State chart Diagram Symbols and Notations

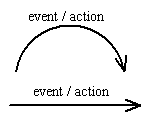
States

States represent situations during the life of an object. You can easily illustrate a state in Smart Draw by using a rectangle with rounded corners.



Transition

A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it.



Initial State

A filled circle followed by an arrow represents the object's initial state.

Initial State

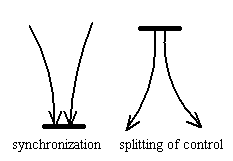
Final State

An arrow pointing to a filled circle nested inside another circle represents the object's final state.

Final State

Synchronization and Splitting of Control

A short heavy bar with two transitions entering it represents a synchronization of control. A short heavy bar with two transitions leaving it represents a splitting of control that creates multiple states.



STATE CHART DIAGRAM:

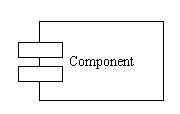
What is a UML Component Diagram?

A component diagram describes the organization of the physical components in a system.

Basic Component Diagram Symbols and Notations

Component

A component is a physical building block of the system. It is represented as a rectangle with tabs.  
[Learn how to resize grouped objects like components.](http://www.smartdraw.com/resources/tutorials/Objects)



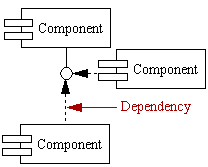
Interface

An interface describes a group of operations used or created by components.

Interface

Dependencies

Draw dependencies among components using dashed arrows.  
[Learn about line styles in SmartDraw.](http://www.smartdraw.com/resources/tutorials/Lines)



COMPONENT DIAGRAM:

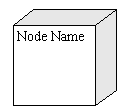
**What is a UML Deployment Diagram?**

Deployment diagrams depict the physical resources in a system including nodes, components, and connections.

Basic Deployment Diagram Symbols and Notations

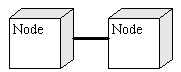
Component

A node is a physical resource that executes code components.  
[Learn how to resize grouped objects like nodes.](http://www.smartdraw.com/resources/tutorials/Objects)



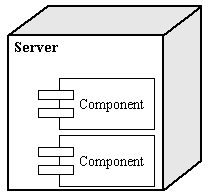
Association

Association refers to a physical connection between nodes, such as Ethernet.  
[Learn how to connect two nodes.](http://www.smartdraw.com/resources/tutorials/Lines)

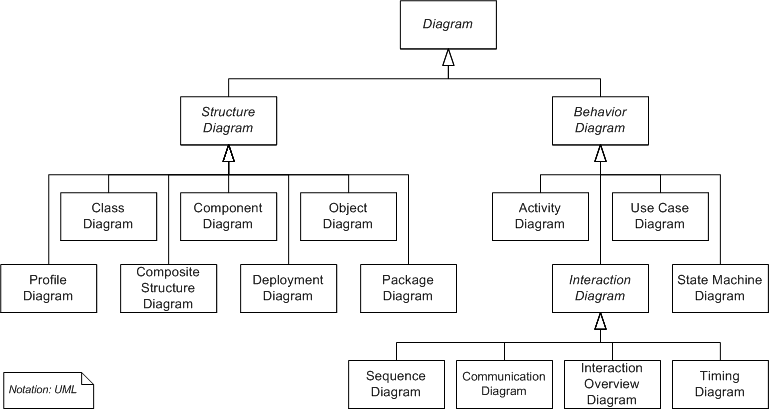


Components and Nodes

Place components inside the node that deploys them.



**UML Diagrams Overview**



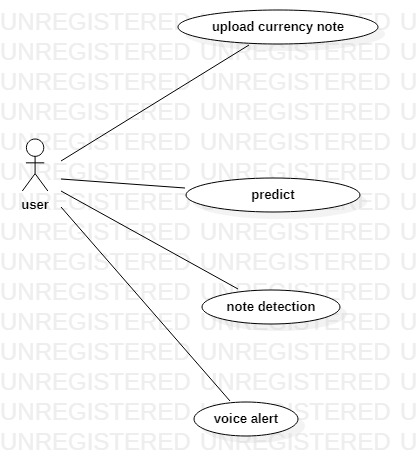
UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

**USECASE DAIGRAM:**

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases.So we can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

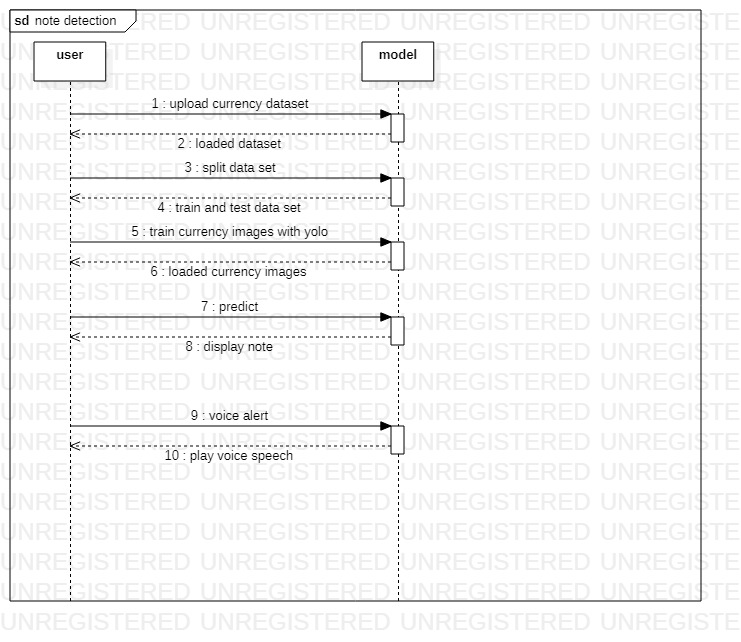
The actors can be human user, some internal applications or may be some external applications. So in a brief when we are planning to draw an use case diagram we should have the following items identified.

* Functionalities to be represented as an use case
* Actors
* Relationships among the use cases and actors.



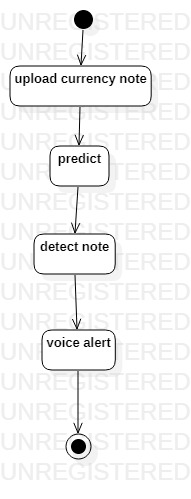
**SEQUENCE DAIGRAM**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

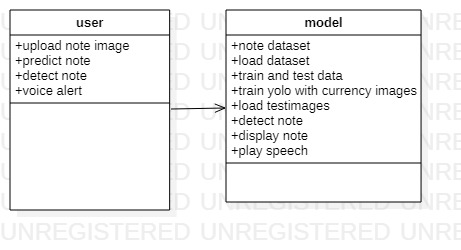


**ACTIVITY DAIGRAM:**

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



**Class diagram**



**CHAPTER-5**

**IMPLEMENTATION**

**5.1 MODULE & DESCRIPTION**

**Modules**

* Dataset of indian currency notes
* Dataset Augmentation
* Dataset Annotation
* Train the model
* Inference the model against test data
* Evaluation

**Modules discription**

**Dataset and Experimental Environment:**

Currency notes used in the experiment are collected from sources like banks and shops. The dataset comprises of old notes of ₹10, ₹20 and new notes of ₹50, ₹100, ₹200, ₹500 and ₹2000 images (in .jpg format), in equal proportions. The images data comprise 200 images per class, which are captured on different backgrounds and different visual settings.

**Dataset Augmentation:**

These 1400 images are then augmented using rotation, blur and shift transformation techniques to generate dataset with 5600 images.

**Dataset Annotation:**

These 5600 images are then manually annotated using MakeSense annotator tool. The annotations are then exported in YOLO format and stored in labels directory Split the entire dataset along with their labels in 70-30 split. The training dataset comprises 3920 images and their labels. Remaining 1680 images along with their annotations comprise the validation dataset.

**Train the model:**

The image size is set to 416. The model epochs are set to 30 and batch size is set to 32. Set the directory path for train dataset and configuration files. Then, run the python command to train the entire dataset using YOLOv5 model. The evaluation metrics are – Precision, Recall and mean Average Precision (mAP). The losses calculated for box, object and class for each epoch.

**Inference the model against test data:**

The model is inferenced using best.pt as model weight and setting the directory path of test data. The test data contains 100 images of all classes and video file containing currency note image frames.

**Evaluation**

The evaluation metrics used to analyse the results include – Precision, recall and mAP. Higher values of recall, precision and mAP (mean Average Precision) are preferred to detect the currency notes accurately. The other metrics include box loss, obj loss and cls loss. In a confusion matrix with True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN), the evaluation metrics Precision, recall and F1- score can be calculated.

**5.2 TECHNOLOGY DESCRIPTION**

**Introduction of Python**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language), Python has a design philosophy that emphasizes code [readability](https://en.wikipedia.org/wiki/Readability) (notably using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B)or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation). Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library)

## What is Python

## Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

**It is used for:**

* web development (server-side),
* software development,
* mathematics,
* system scripting.

### What can Python do

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

### Why Python

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

### Good to know

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

### Python Syntax compared to other programming languages

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

**Python Install**

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

C:\Users\Your Name>python --version

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

python --version

If you find that you do not have python installed on your computer, then you can download it for free from the following website: https://www.python.org/

**Python Quickstart**

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

**The way to run a python file is like this on the command line:**

C:\Users\Your Name>python helloworld.py

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

helloworld.py

print("Hello, World!")

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

C:\Users\Your Name>python helloworld.py

The output should read:

Hello, World!

Congratulations, you have written and executed your first Python program.

**The Python Command Line**

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

C:\Users\Your Name>python

Or, if the "python" command did not work, you can try "py":

C:\Users\Your Name>py

**From there you can write any python, including our hello world example from earlier in the tutorial:**

C:\Users\Your Name>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>print("Hello, World!")

Which will write "Hello, World!" in the command line:

C:\Users\Your Name>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>>print("Hello, World!")

Hello, World!

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

exit()

**Virtual Environments and Packages**

Introduction

Python applications will often use packages and modules that don’t come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library’s interface.

This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

The solution for this problem is to create a virtual environment, a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0. If application B requires a library be upgraded to version 3.0, this will not affect application A’s environment.

**Creating Virtual Environments**

The module used to create and manage virtual environments is called venv. venv will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running python3 or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the venv module as a script with the directory path:

python3 -m venv tutorial-env

This will create the tutorial-env directory if it doesn’t exist, and also create directories inside it containing a copy of the Python interpreter, the standard library, and various supporting files.

A common directory location for a virtual environment is .venv. This name keeps the directory typically hidden in your shell and thus out of the way while giving it a name that explains why the directory exists. It also prevents clashing with .env environment variable definition files that some tooling supports.

Once you’ve created a virtual environment, you may activate it.

On Windows, run:

tutorial-env\Scripts\activate.bat

On Unix or MacOS, run:

source tutorial-env/bin/activate

(This script is written for the bash shell. If you use the csh or fish shells, there are alternate activate.csh and activate.fish scripts you should use instead.)

Activating the virtual environment will change your shell’s prompt to show what virtual environment you’re using, and modify the environment so that running python will get you that particular version and installation of Python. For example:

$ source ~/envs/tutorial-env/bin/activate

(tutorial-env) $ python

Python 3.5.1 (default, May 6 2016, 10:59:36)

...

>>> import sys

>>>sys.path

['', '/usr/local/lib/python35.zip', ...,

'~/envs/tutorial-env/lib/python3.5/site-packages']

>>>

Managing Packages with pip

You can install, upgrade, and remove packages using a program called pip. By default pip will install packages from the Python Package Index, <https://pypi.org>. You can browse the Python Package Index by going to it in your web browser, or you can use pip’s limited search feature:

(tutorial-env) $ pip search astronomy

skyfield - Elegant astronomy for Python

gary - Galactic astronomy and gravitational dynamics.

novas - The United States Naval Observatory NOVAS astronomy library

astroobs - Provides astronomy ephemeris to plan telescope observations

PyAstronomy - A collection of astronomy related tools for Python.

...

pip has a number of subcommands: “search”, “install”, “uninstall”, “freeze”, etc. (Consult the Installing Python Modules guide for complete documentation for pip.)

You can install the latest version of a package by specifying a package’s name:

(tutorial-env) $ pip install novas

Collecting novas

Downloading novas-3.1.1.3.tar.gz (136kB)

Installing collected packages: novas

Running setup.py install for novas

Successfully installed novas-3.1.1.3

You can also install a specific version of a package by giving the package name followed by == and the version number:

(tutorial-env) $ pip install requests==2.6.0

Collecting requests==2.6.0

Using cached requests-2.6.0-py2.py3-none-any.whl

Installing collected packages: requests

Successfully installed requests-2.6.0

If you re-run this command, pip will notice that the requested version is already installed and do nothing. You can supply a different version number to get that version, or you can run pip install --upgrade to upgrade the package to the latest version:

(tutorial-env) $ pip install --upgrade requests

Collecting requests

Installing collected packages: requests

Found existing installation: requests 2.6.0

Uninstalling requests-2.6.0:

Successfully uninstalled requests-2.6.0

Successfully installed requests-2.7.0

pip uninstall followed by one or more package names will remove the packages from the virtual environment.

**pip show will display information about a particular package:**

(tutorial-env) $ pip show requests

---

Metadata-Version: 2.0

Name: requests

Version: 2.7.0

Summary: Python HTTP for Humans.

Home-page: http://python-requests.org

Author: Kenneth Reitz

Author-email: me@kennethreitz.com

License: Apache 2.0

Location: /Users/akuchling/envs/tutorial-env/lib/python3.4/site-packages

Requires:

pip list will display all of the packages installed in the virtual environment:

(tutorial-env) $ pip list

novas (3.1.1.3)

numpy (1.9.2)

pip (7.0.3)

requests (2.7.0)

setuptools (16.0)

pip freeze will produce a similar list of the installed packages, but the output uses the format that pip install expects. A common convention is to put this list in a requirements.txt file:

(tutorial-env) $ pip freeze > requirements.txt

(tutorial-env) $ cat requirements.txt

novas==3.1.1.3

numpy==1.9.2

requests==2.7.0

The requirements.txt can then be committed to version control and shipped as part of an application. Users can then install all the necessary packages with install -r:

(tutorial-env) $ pip install -r requirements.txt

Collecting novas==3.1.1.3 (from -r requirements.txt (line 1))

...

Collecting numpy==1.9.2 (from -r requirements.txt (line 2))

...

Collecting requests==2.7.0 (from -r requirements.txt (line 3))

...

Installing collected packages: novas, numpy, requests

Running setup.py install for novas

Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0

pip has many more options. Consult the Installing Python Modules guide for complete documentation for pip. When you’ve written a package and want to make it available on the Python Package Index, consult the Distributing Python Modules guide.

**Cross Platform**

Platform. Architecture (executable=sys.executable, bits='', linkage='')

Queries the given executable (defaults to the Python interpreter binary) for various architecture information.

Returns a tuple (bits, linkage) which contain information about the bit architecture and the linkage format used for the executable. Both values are returned as strings.

Values that cannot be determined are returned as given by the parameter presets. If bits is given as '', the sizeof(pointer) (or sizeof(long) on Python version < 1.5.2) is used as indicator for the supported pointer size.

The function relies on the system’s file command to do the actual work. This is available on most if not all Unix platforms and some non-Unix platforms and then only if the executable points to the Python interpreter. Reasonable defaults are used when the above needs are not met.

Note On Mac OS X (and perhaps other platforms), executable files may be universal files containing multiple architectures.

To get at the “64-bitness” of the current interpreter, it is more reliable to query the sys.maxsize attribute:

is\_64bits = sys.maxsize> 2\*\*32

platform.machine ()

Returns the machine type, e.g. 'i386'. An empty string is returned if the value cannot be determined.

platform.node ()

Returns the computer’s network name (may not be fully qualified!). An empty string is returned if the value cannot be determined.

platform. Platform(aliased=0, terse=0)

Returns a single string identifying the underlying platform with as much useful information as possible.

The output is intended to be human readable rather than machine parseable. It may look different on different platforms and this is intended.

If aliased is true, the function will use aliases for various platforms that report system names which differ from their common names, for example SunOS will be reported as Solaris. The system\_alias() function is used to implement this.

Setting terse to true causes the function to return only the absolute minimum information needed to identify the platform.

platform.processor()

Returns the (real) processor name, e.g. 'amdk6'.

An empty string is returned if the value cannot be determined. Note that many platforms do not provide this information or simply return the same value as for machine(). NetBSD does this.

platform.python\_build()

Returns a tuple (buildno, builddate) stating the Python build number and date as strings.

platform.python\_compiler()

Returns a string identifying the compiler used for compiling Python.

platform.python\_branch()

Returns a string identifying the Python implementation SCM branch.

New in version 2.6.

platform.python\_implementation()

Returns a string identifying the Python implementation. Possible return values are: ‘CPython’, ‘IronPython’, ‘Jython’, ‘PyPy’.

New in version 2.6.

platform.python\_revision()

Returns a string identifying the Python implementation SCM revision.

New in version 2.6.

platform.python\_version()

Returns the Python version as string 'major.minor.patchlevel'.

Note that unlike the Python sys.version, the returned value will always include the patchlevel (it defaults to 0).

platform.python\_version\_tuple()

Returns the Python version as tuple (major, minor, patchlevel) of strings.

Note that unlike the Python sys.version, the returned value will always include the patchlevel (it defaults to '0').

platform.release()

Returns the system’s release, e.g. '2.2.0' or 'NT' An empty string is returned if the value cannot be determined.

platform.system()

Returns the system/OS name, e.g. 'Linux', 'Windows', or 'Java'. An empty string is returned if the value cannot be determined.

platform.system\_alias(system, release, version)

Returns (system, release, version) aliased to common marketing names used for some systems. It also does some reordering of the information in some cases where it would otherwise cause confusion.

platform.version()

Returns the system’s release version, e.g. '#3 on degas'. An empty string is returned if the value cannot be determined.

platform.uname()

Fairly portable uname interface. Returns a tuple of strings (system, node, release, version, machine, processor) identifying the underlying platform.

Note that unlike the os.uname() function this also returns possible processor information as additional tuple entry.

Entries which cannot be determined are set to ''.

Java Platform

platform.java\_ver(release='', vendor='', vminfo=('', '', ''), osinfo=('', '', ''))

Version interface for Jython.

Returns a tuple (release, vendor, vminfo, osinfo) with vminfo being a tuple (vm\_name, vm\_release, vm\_vendor) and osinfo being a tuple (os\_name, os\_version, os\_arch). Values which cannot be determined are set to the defaults given as parameters (which all default to '').

Windows Platform

platform.win32\_ver(release='', version='', csd='', ptype='')

Get additional version information from the Windows Registry and return a tuple (release, version, csd, ptype) referring to OS release, version number, CSD level (service pack) and OS type (multi/single processor).

As a hint: ptype is 'Uniprocessor Free' on single processor NT machines and 'Multiprocessor Free' on multi processor machines. The ‘Free’ refers to the OS version being free of debugging code. It could also state ‘Checked’ which means the OS version uses debugging code, i.e. code that checks arguments, ranges, etc.

Note This function works best with Mark Hammond’s win32all package installed, but also on Python 2.3 and later (support for this was added in Python 2.6). It obviously only runs on Win32 compatible platforms.

Win95/98 specific

platform.popen(cmd, mode='r', bufsize=None)

Portable popen() interface. Find a working popen implementation preferring win32pipe.popen(). On Windows NT, win32pipe.popen() should work; on Windows 9x it hangs due to bugs in the MS C library.

Mac OS Platform

platform.mac\_ver(release='', versioninfo=('', '', ''), machine='')

Get Mac OS version information and return it as tuple (release, versioninfo, machine) with versioninfo being a tuple (version, dev\_stage, non\_release\_version).

Entries which cannot be determined are set to ''. All tuple entries are strings.

Unix Platforms

platform.dist(distname='', version='', id='', supported\_dists=('SuSE', 'debian', 'redhat', 'mandrake', ...))

This is an old version of the functionality now provided by linux\_distribution(). For new code, please use the linux\_distribution().

The only difference between the two is that dist() always returns the short name of the distribution taken from the supported\_dists parameter.

Deprecated since version 2.6.

platform.linux\_distribution(distname='', version='', id='', supported\_dists=('SuSE', 'debian', 'redhat', 'mandrake', ...), full\_distribution\_name=1)

Tries to determine the name of the Linux OS distribution name.

supported\_dists may be given to define the set of Linux distributions to look for. It defaults to a list of currently supported Linux distributions identified by their release file name.

If full\_distribution\_name is true (default), the full distribution read from the OS is returned. Otherwise the short name taken from supported\_dists is used.

Returns a tuple (distname,version,id) which defaults to the args given as parameters. id is the item in parentheses after the version number. It is usually the version codename.

Note This function is deprecated since Python 3.5 and removed in Python 3.8. See alternative like the distro package.

New in version 2.6.

platform.libc\_ver(executable=sys.executable, lib='', version='', chunksize=2048)

Tries to determine the libc version against which the file executable (defaults to the Python interpreter) is linked. Returns a tuple of strings (lib, version) which default to the given parameters in case the lookup fails.

Note that this function has intimate knowledge of how different libc versions add symbols to the executable is probably only usable for executables compiled using gcc. The file is read and scanned in chunks of chunksize bytes.

**Using the Python Interpreter**

Invoking the Interpreter

The Python interpreter is usually installed as /usr/local/bin/python3.8 on those machines where it is available; putting /usr/local/bin in your Unix shell’s search path makes it possible to start it by typing the command:

python3.8

to the shell. 1 Since the choice of the directory where the interpreter lives is an installation option, other places are possible; check with your local Python guru or system administrator. (E.g., /usr/local/python is a popular alternative location.)

On Windows machines where you have installed Python from the Microsoft Store, the python3.8 command will be available. If you have the py.exe launcher installed, you can use the py command. See Excursus: Setting environment variables for other ways to launch Python.

Typing an end-of-file character (Control-D on Unix, Control-Z on Windows) at the primary prompt causes the interpreter to exit with a zero exit status. If that doesn’t work, you can exit the interpreter by typing the following command: quit().

The interpreter’s line-editing features include interactive editing, history substitution and code completion on systems that support the GNU Readline library. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys. If nothing appears to happen, or if ^P is echoed, command line editing isn’t available; you’ll only be able to use backspace to remove characters from the current line.

The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file.

A second way of starting the interpreter is python -c command [arg] ..., which executes the statement(s) in command, analogous to the shell’s -c option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote command in its entirety with single quotes.

Some Python modules are also useful as scripts. These can be invoked using python -m module [arg] ..., which executes the source file for module as if you had spelled out its full name on the command line.

When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing -i before the script.

**All command line options are described in Command line and environment.**

Argument Passing

When known to the interpreter, the script name and additional arguments thereafter are turned into a list of strings and assigned to the argv variable in the sys module. You can access this list by executing import sys. The length of the list is at least one; when no script and no arguments are given, sys.argv[0] is an empty string. When the script name is given as '-' (meaning standard input), sys.argv[0] is set to '-'. When -c command is used, sys.argv[0] is set to '-c'. When -m module is used, sys.argv[0] is set to the full name of the located module. Options found after -c command or -m module are not consumed by the Python interpreter’s option processing but left in sys.argv for the command or module to handle.

Interactive Mode

When commands are read from a tty, the interpreter is said to be in interactive mode. In this mode it prompts for the next command with the primary prompt, usually three greater-than signs (>>>); for continuation lines it prompts with the secondary prompt, by default three dots (...). The interpreter prints a welcome message stating its version number and a copyright notice before printing the first prompt:

$ python3.8

Python 3.8 (default, Sep 16 2015, 09:25:04)

[GCC 4.8.2] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>>

Continuation lines are needed when entering a multi-line construct. As an example, take a look at this if statement:

>>>

>>>the\_world\_is\_flat = True

>>>ifthe\_world\_is\_flat:

... print("Be careful not to fall off!")

...

Be careful not to fall off!

For more on interactive mode, see Interactive Mode.

**The Interpreter and Its Environment**

Source Code Encoding

By default, Python source files are treated as encoded in UTF-8. In that encoding, characters of most languages in the world can be used simultaneously in string literals, identifiers and comments — although the standard library only uses ASCII characters for identifiers, a convention that any portable code should follow. To display all these characters properly, your editor must recognize that the file is UTF-8, and it must use a font that supports all the characters in the file.

To declare an encoding other than the default one, a special comment line should be added as the first line of the file. The syntax is as follows:

# -\*- coding: encoding -\*-

where encoding is one of the valid codecs supported by Python.

For example, to declare that Windows-1252 encoding is to be used, the first line of your source code file should be:

# -\*- coding: cp1252 -\*-

One exception to the first line rule is when the source code starts with a UNIX “shebang” line. In this case, the encoding declaration should be added as the second line of the file. For example:

#!/usr/bin/env python3

# -\*- coding: cp1252 -\*-

# ****Introduction to Artificial Intelligence****

“The science and engineering of making intelligent machines, especially intelligent computer programs”. -John McCarthy-

Artificial Intelligence is an approach to make a computer, a robot, or a product to think how smart human think. AI is a study of how human brain think, learn, decide and work, when it tries to solve problems. And finally this study outputs intelligent software systems.The aim of AI is to improve computer functions which are related to human knowledge, for example, reasoning, learning, and problem-solving.

The intelligence is intangible. It is composed of

* Reasoning
* Learning
* Problem Solving
* Perception
* Linguistic Intelligence

The objectives of AI research are reasoning, knowledge representation, planning, learning, natural language processing, realization, and ability to move and manipulate objects. There are long-term goals in the general intelligence sector.

Approaches include statistical methods, computational intelligence, and traditional coding AI. During the AI research related to search and mathematical optimization, artificial neural networks and methods based on statistics, probability, and economics, we use many tools. Computer science attracts AI in the field of science, mathematics, psychology, linguistics, philosophy and so on.

# Trending AI Articles:

[1. Cheat Sheets for AI, Neural Networks, Machine Learning, Deep Learning & Big Data](https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463)

[2. Data Science Simplified Part 1: Principles and Process](https://becominghuman.ai/data-science-simplified-principles-and-process-b06304d63308)

[3. Getting Started with Building Realtime API Infrastructure](https://becominghuman.ai/getting-started-with-building-realtime-api-infrastructure-a19601fc794e)

[4. AI & NLP Workshop](https://becominghuman.ai/ai-nlp-workshop-7bc121986d61)

**Applications of AI**

· Gaming − AI plays important role for machine to think of large number of possible positions based on deep knowledge in strategic games. for example, chess,river crossing, N-queens problems and etc.

Natural Language Processing − Interact with the computer that understands natural language spoken by humans.

· Expert Systems − Machine or software provide explanation and advice to the users.

· Vision Systems − Systems understand, explain, and describe visual input on the computer.

· Speech Recognition − There are some AI based speech recognition systems have ability to hear and express as sentences and understand their meanings while a person talks to it. For example Siri and Google assistant.

· Handwriting Recognition − The handwriting recognition software reads the text written on paper and recognize the shapes of the letters and convert it into editable text.

· Intelligent Robots − Robots are able to perform the instructions given by a human.

**Major Goals**

* Knowledge reasoning
* Planning
* Machine Learning
* Natural Language Processing
* Computer Vision
* Robotics

**IBM Watson**



“Watson” is an IBM supercomputer that combines Artificial Intelligence (AI) and complex inquisitive programming for ideal execution as a “question answering” machine. The supercomputer is named for IBM’s founder, Thomas J. Watson.

IBM Watson is at the forefront of the new era of computing. At the point when IBM Watson made, IBM communicated that “more than 100 particular techniques are used to inspect perceive sources, find and make theories, find and score affirm, and combination and rank speculations.” recently, the Watson limits have been expanded and the way by which Watson works has been changed to abuse new sending models (Watson on IBM Cloud) and propelled machine learning capacities and upgraded hardware open to architects and authorities. It isn’t any longer completely a request answering figuring system arranged from Q&A joins yet can now ‘see’, ‘hear’, ‘read’, ‘talk’, ‘taste’, ‘translate’, ‘learn’ and ‘endorse’.

# Machine Learning

### Introduction

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers.

Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

In this tutorial, we’ll look into the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbor algorithm, decision tree learning, and deep learning. We’ll explore which programming languages are most used in machine learning, providing you with some of the positive and negative attributes of each. Additionally, we’ll discuss biases that are perpetuated by machine learning algorithms, and consider what can be kept in mind to prevent these biases when building algorithms.

## Machine Learning Methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.

Two of the most widely adopted machine learning methods are **supervised learning** which trains algorithms based on example input and output data that is labeled by humans, and **unsupervised learning** which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let’s explore these methods in more detail.

### Supervised Learning

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able to “learn” by comparing its actual output with the “taught” outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabeled data.

For example, with supervised learning, an algorithm may be fed data with images of sharks labeled as fish and images of oceans labeled as water. By being trained on this data, the supervised learning algorithm should be able to later identify unlabeled shark images as fish and unlabeled ocean images as water.

A common use case of supervised learning is to use historical data to predict statistically likely future events. It may use historical stock market information to anticipate upcoming fluctuations, or be employed to filter out spam emails. In supervised learning, tagged photos of dogs can be used as input data to classify untagged photos of dogs.

### Unsupervised Learning

In unsupervised learning, data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable.

The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.

Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases.

Without being told a “correct” answer, unsupervised learning methods can look at complex data that is more expansive and seemingly unrelated in order to organize it in potentially meaningful ways. Unsupervised learning is often used for anomaly detection including for fraudulent credit card purchases, and recommender systems that recommend what products to buy next. In unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together.

## Approaches

As a field, machine learning is closely related to computational statistics, so having a background knowledge in statistics is useful for understanding and leveraging machine learning algorithms.

For those who may not have studied statistics, it can be helpful to first define correlation and regression, as they are commonly used techniques for investigating the relationship among quantitative variables. **Correlation** is a measure of association between two variables that are not designated as either dependent or independent. **Regression** at a basic level is used to examine the relationship between one dependent and one independent variable. Because regression statistics can be used to anticipate the dependent variable when the independent variable is known, regression enables prediction capabilities.

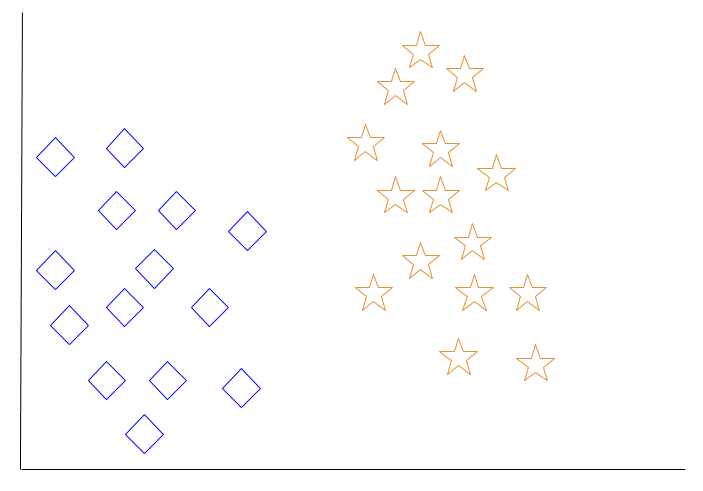
Approaches to machine learning are continuously being developed. For our purposes, we’ll go through a few of the popular approaches that are being used in machine learning at the time of writing.

### k-nearest neighbor

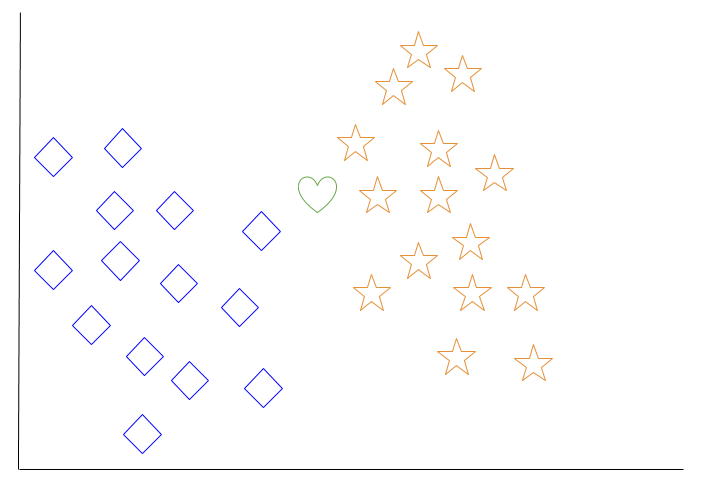
The k-nearest neighbor algorithm is a pattern recognition model that can be used for classification as well as regression. Often abbreviated as k-NN, the **k** in k-nearest neighbor is a positive integer, which is typically small. In either classification or regression, the input will consist of the k closest training examples within a space.

We will focus on k-NN classification. In this method, the output is class membership. This will assign a new object to the class most common among its k nearest neighbors. In the case of k = 1, the object is assigned to the class of the single nearest neighbor.

Let’s look at an example of k-nearest neighbor. In the diagram below, there are blue diamond objects and orange star objects. These belong to two separate classes: the diamond class and the star class.

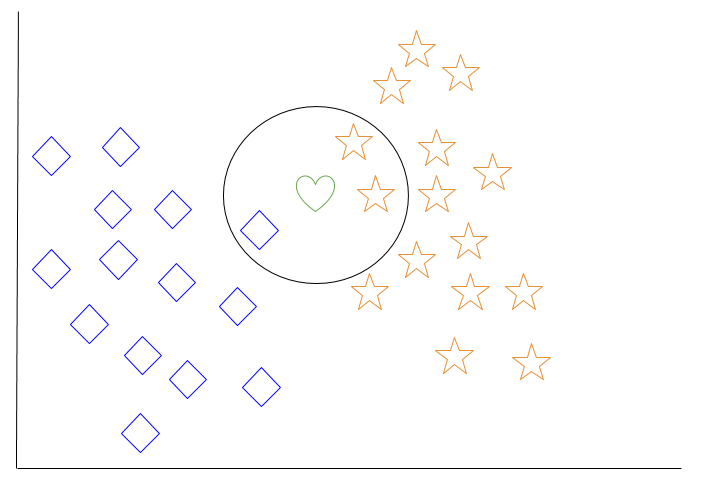


When a new object is added to the space — in this case a green heart — we will want the machine learning algorithm to classify the heart to a certain class.



When we choose k = 3, the algorithm will find the three nearest neighbors of the green heart in order to classify it to either the diamond class or the star class.

In our diagram, the three nearest neighbors of the green heart are one diamond and two stars. Therefore, the algorithm will classify the heart with the star class.



Among the most basic of machine learning algorithms, k-nearest neighbor is considered to be a type of “lazy learning” as generalization beyond the training data does not occur until a query is made to the system.

### Decision Tree Learning

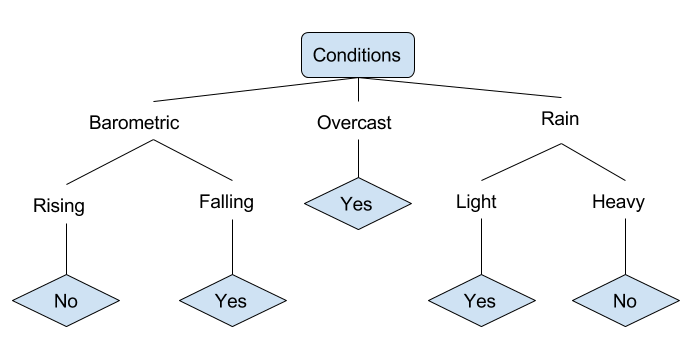
For general use, decision trees are employed to visually represent decisions and show or inform decision making. When working with machine learning and data mining, decision trees are used as a predictive model. These models map observations about data to conclusions about the data’s target value.

The goal of decision tree learning is to create a model that will predict the value of a target based on input variables.

In the predictive model, the data’s attributes that are determined through observation are represented by the branches, while the conclusions about the data’s target value are represented in the leaves.

When “learning” a tree, the source data is divided into subsets based on an attribute value test, which is repeated on each of the derived subsets recursively. Once the subset at a node has the equivalent value as its target value has, the recursion process will be complete.

Let’s look at an example of various conditions that can determine whether or not someone should go fishing. This includes weather conditions as well as barometric pressure conditions.



In the simplified decision tree above, an example is classified by sorting it through the tree to the appropriate leaf node. This then returns the classification associated with the particular leaf, which in this case is either a Yes or a No. The tree classifies a day’s conditions based on whether or not it is suitable for going fishing.

A true classification tree data set would have a lot more features than what is outlined above, but relationships should be straightforward to determine. When working with decision tree learning, several determinations need to be made, including what features to choose, what conditions to use for splitting, and understanding when the decision tree has reached a clear ending.

# Introduction to Deep Learning

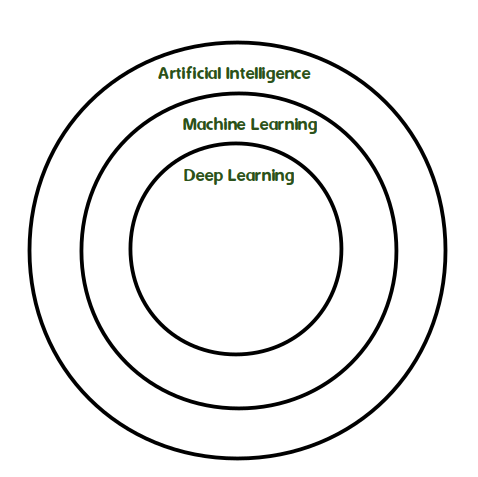
What is deep learning

Deep learning is a branch of [machine learning](https://www.geeksforgeeks.org/introduction-machine-learning/) which is completely based on [artificial neural networks](https://www.geeksforgeeks.org/tag/neural-network/), as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don’t need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It’s on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture.

**A formal definition of deep learning is- neurons**

Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousand of their neighbours.  
The question here is how we recreate these neurons in a computer. So, we create an artificial structure called an artificial neural net where we have nodes or neurons. We have some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.



**PYTHON**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language), Python has a design philosophy that emphasizes code [readability](https://en.wikipedia.org/wiki/Readability) (notably using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B)or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation). Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library)

**DJANGO**

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes [reusability](https://en.wikipedia.org/wiki/Reusability)and "pluggability" of components, rapid development, and the principle of [don't repeat yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself). Python is used throughout, even for settings files and data models.



Django also provides an optional administrative [create, read, update and delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) interface that is generated dynamically through [introspection](https://en.wikipedia.org/wiki/Introspection_(computer_science)) and configured via admin models



**5.3. CODING**

**5.4 OUTPUT SCREENS**

**CHAPTER-6**

**SYSTEM TESTING**

**SYSTEM TEST**

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

### TYPES OF TESTS

**Unit testing**

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

**Integration testing**

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

**Functional test**

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

Functional testing is centered on the following items:

* Valid Input : identified classes of valid input must be accepted.
* Invalid Input : identified classes of invalid input must be rejected.
* Functions : identified functions must be exercised.
* Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

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

**System Test**

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

**White Box Testing**

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

**Black Box Testing**

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

**Unit Testing**

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

**Test strategy and approach**

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

**Test objectives**

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

**Features to be tested**

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

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

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

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

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

**CONCLUSION**

The proposed YOLOv5 model detects currency notes with high precision, recall and mAP. The model detected 85 images accurately, out of 100 test data images. A native YOLOv5 based web app was designed that detects currency notes withhigh bounding box probability. The web app provides speech out files of detected currency note labels in English as well as Hindi. The model can be extended to detect foreign currency notes. Optical Character Recognition (OCR) can be introduced in the system to enhance the model performance.

**REFERENCES**

[1] J. G. Shanahan and L. Dai, “Realtime object detection via deep learning-based pipelines,” Int. Conf. Inf. Knowl. Manag. Proc., pp. 2977–2978, 2019, doi: 10.1145/3357384.3360320.

[2] World Health Organization;, “Blindness and vision impairment: WHO Factsheet ,” World Heal. Organ., vol. 11, no. 2018, pp. 1–5, 2021, [Online]. Available: https://www.who.int/news-room/factsheets/detail/blindness-and-visual-impairment.

[3] P. Ackland, S. Resnikoff, and R. Bourne, “World blindness and visual impairment: Despite many successes, the problem is growing,” Community Eye Heal. J., vol. 30, no. 100, pp. 71–73, 2018.

[4] S. D. Achar, C. Shankar Singh, C. S. Sumanth Rao, K. Pavana Narayana, and A. Dasare, “Indian Currency Recognition System Using CNN and Comparison with YOLOv5,” 2021 IEEE Int. Conf. Mob. Networks Wirel. Commun. ICMNWC 2021, pp. 1–6, 2021, doi: 10.1109/ICMNWC52512.2021.9688513.

[5] H. Hassanpour, A. Yaseri, and G. Ardeshiri, “Feature extraction for paper currency recognition,” 2007 9th Int. Symp. Signal Process. its Appl. ISSPA 2007, Proc., pp. 7–10, 2007, doi: 10.1109/ISSPA.2007.4555366.

[6] S. Luo and W. Zheng, “You-Only-Look-Once-v5 Based Table Detection for Academic Papers,” 2021 Int. Conf. Digit. Soc. Intell. Syst. DSInS 2021, pp. 53–56, 2021, doi: 10.1109/DSInS54396.2021.9670600.

[7] K. K. S. N. Reddy, C. Yashwanth, S. H. Kvs, P. A. T. V. Sai, and S. Khetarpaul, “Object and Currency Detection with Audio Feedback for Visually Impaired,” 2020 IEEE Reg. 10 Symp. TENSYMP 2020,no. June, pp. 1152–1155, 2020, doi: 10.1109/TENSYMP50017.2020.9230687. [8] P. Shreya, N. Shreyas, D. Pushya, and N. Uma Maheswar Reddy, “BLIND ASSIST : A One Stop Mobile Application for the Visually Impaired,” 2021 IEEE Pune Sect. Int. Conf. PuneCon 2021, pp. 1–4, 2021, doi: 10.1109/PuneCon52575.2021.9686476.

[9] R. R. M. Et al., “Currency Detection for Visually Impaired Iraqi Banknote as a Study Case,” Turkish J. Comput. Math. Educ., vol. 12, no. 6, pp. 2940–2948, 2021, doi: 10.17762/turcomat.v12i6.6078.

[10] M. Mahendru and S. K. Dubey, “Real time object detection with audio feedback using Yolo vs. Yolo\_V3,” Proc. Conflu. 2021 11th Int. Conf. Cloud Comput. Data Sci. Eng., pp. 734–740, 2021, doi: 10.1109/Confluence51648.2021.9377064.

[11] R. Padilla, S. L. Netto, and E. A. B. Da Silva, “A Survey on Performance Metrics for Object-Detection Algorithms,” Int. Conf. Syst. Signals, Image Process., vol. 2020-July, pp. 237–242, 2020, doi: 10.1109/IWSSIP48289.2020.9145130.

[12] M. Kasper-Eulaers, N. Hahn, P. E. Kummervold, S. Berger, T. Sebulonsen, and Ø. Myrland, “Short communication: Detecting heavy goods vehicles in rest areas in winter conditions using YOLOv5,” Algorithms, vol. 14, no. 4, 2021, doi: 10.3390/a14040114.

[13] R. Swami, S. Khurana, S. Singh, S. Thakur, and P. K. R. Sajjala, “Indian Currency Classification Using Deep Learning Techniques,” 2022 12th Int. Conf. Cloud Comput. Data Sci. Eng., pp. 567–572, 2022, doi: 10.1109/confluence52989.2022.9734149.

[14] R. C. Joshi, S. Yadav, and M. K. Dutta, “YOLO-v3 Based Currency Detection and Recognition System for Visually Impaired Persons,” 2020 Int. Conf. Contemp. Comput. Appl. IC3A 2020, pp. 280–285, 2020, doi: 10.1109/IC3A48958.2020.233314.