

Image Segmentation using Mask R-CNN

A Major Project Report Submitted
In partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology in Information Technology

by

K. Jithender Reddy	18N31A1272
N. Bheemesh Kumar	18N31A12B2
N. Chakravarthi	18N31A12C0
N. Harishkrishna	19N35A1212

Under the esteemed guidance of
Ms.B.Pavani
Asst. Professor



Department of Information Technology

Malla Reddy College of Engineering & Technology

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA & NAAC with 'A' Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100

website: www.mrcet.ac.in

2018-2022



Malla Reddy College of Engineering & Technology

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA & NAAC with 'A' Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100

website: www.mrcet.ac.in

CERTIFICATE

This is to certify that this is the bonafide record of the project entitled **“Image segmentation using mask RCNN”**, submitted by **K Jithender Reddy (18N31A1272)**, **N Bemesh Kumar (18N31A12B2)**, **N Chakravarthi (18N31A12C0)** and **N Harishkrishna (19N35A1212)** of B.Tech in the partial fulfillment of the requirements for the degree of Bachelor of Technology in Information Technology, Department of IT during the year 2021-2022. The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Guide

Ms.B.Pavani
Asst. Professor

Head of the Department

Dr. G.Sharada
Professor

External Examiner

DECLARATION

We hereby declare that the project titled “**Image Segmentation using mask RCNN**” submitted to Malla Reddy College of Engineering and Technology (UGC Autonomous), affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Information Technology is a result of original research carried-out in this report. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

K Jithender Reddy – 18N31A1272

N Bemesh Kumar – 18N31A12B2

N Chakravarthi – 18N31A12C0

N Harishkrishna – 19N35A1212

ACKNOWLEDGEMENT

We feel honored to place our warm salutation to our college Malla Reddy College of Engineering and Technology (UGC-Autonomous) for giving us an opportunity to do this Project as part of our B.Tech Program. We are ever grateful to our Director Dr. VSK Reddy and Principal **Dr.S.Srinivas Rao** who enabled us to have experience in engineering and gain profound technical knowledge.

We express our heartiest thanks to our HOD, **Dr. G. Sharada** for encouraging us in every aspect of our course and helping us realize our full potential.

We would like to thank our Project Guide **Ms. B.Pavani** for her regular guidance, suggestions and constant encouragement. We are extremely grateful to our Project Coordinator **Mrs. B. Aruna Kumari** for her continuous monitoring and unflinching co-operation throughout project work.

We would like to thank our Class Incharge **Mrs. B. Aruna kumari** who in spite of being busy with her academic duties took time to guide and keep us on the correct path.

We would also like to thank all the faculty members and supporting staff of the Department of IT and all other departments who have been helpful directly or indirectly in making our project a success.

We are extremely grateful to our parents for their blessings and prayers for the completion of our project that gave us strength to do our project.

With regards and gratitude

K Jithender Reddy – 18N31A1272

N Bemesh Kumar – 18N31A12B2

N Chakravarthi – 18N31A12C0

N Harishkrishna – 19N35A1212

Abstract

Image segmentation is a critical process in computer vision. It involves dividing a visual input into segments to simplify image analysis. Segments represent objects or parts of objects, and comprise sets of pixels, or “super pixels. Recently due to the success of deep learning models in a wide range of vision applications, there has been a substantial amount of work aimed at developing image segmentation approaches using deep learning models. Image segmentation with CNN involves feeding segments of an image as input to a convolutional neural network which labels the pixels. Fully convolutional networks, FCNs use convolutional layers to process varying input sizes and can work faster. The final output layer has a large receptive field and corresponds to the height and width of the image, while the number of channels corresponds to the number of classes. An architecture based on deep encoders and decoders, also known as semantic pixel-wise segmentation. It involves encoding the input image into low dimensions and then recovering it with orientation in-variance capabilities the decoder Most notably is the R-CNN, or Region-Based Convolutional Neural Networks, and the most recent technique called Mask R-CNN that is capable of achieving state-of-the-art results on a range of object detection tasks.

TABLE OF CONTENTS

<u>S.NO</u>	<u>TITLE</u>	<u>PG.NO</u>
1	INTRODUCTION	06
	1.1 PURPOSE AND OBJECTIVES	07
	1.2 EXISTING AND PROPOSED SYSTEM	07
	1.3 SCOPE OF PROJECT	09
2	LITERATURE SURVEY	11
3	SYSTEM ANALYSIS	13
	2.1 HARDWARE AND SOFTWARE REQUIREMENTS	13
	2.2 SOFTWARE REQUIREMENTS SPECIFICATION	13
4	SYSTEM DESIGN	24
	4.1 DESCRIPTION	24
	4.2 ARCHITECTURE	25
	4.3 UML DIAGRAMS	26
5	METHODOLOGY	31
	5.1 TECHNOLOGIES USED	31
	5.2 MODULES DESCRIPTION	37
	5.3 PROCESS/ALGORITHM	38
6	IMPLEMENTATION	46
	6.1 SAMPLE CODE	46
	6.2 OUTPUT SCREENS	52
	6.3 TEST CASES	53
7	CONCLUSION	54
8	FUTURE SCOPE	55
9	BIBLIOGRAPHY	56

1. INTRODUCTION

Image segmentation is a critical process in computer vision. It involves dividing a visual input into segments to simplify image analysis. Segments represent objects or parts of objects, and comprise sets of pixels, or "super-pixels". Image segmentation sorts pixels into larger components, eliminating the need to consider individual pixels -Classification: categorizing the entire image into classes such as animals, humans, objects.

-Object detection: detecting objects in an image and drawing a rectangle around them.

-Segmentation: Identifying the parts of the image and understanding what object it belongs to.

An image is a collection or set of different pixels. We group together the pixels that have similar attributes using image segmentation. Object detection builds a bounding box corresponding to each class in the image. But it tells us nothing about the shape of the object. We only get the set of bounding box coordinates. Image segmentation creates a pixel-wise mask for each object in the image. This technique gives us a far more granular understanding of the object(s) in the image.

In digital image processing and computer vision, image segmentation is the process of partitioning a digital image into multiple image segments, also known as image regions or image objects (sets of pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.

Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image. Each of the pixels in a region are similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different in color with respect to the same characteristic(s).

When applied to a stack of images, typical in medical imaging, the resulting contours after image segmentation can be used to create 3D reconstructions with the help of interpolation algorithms like marching cube

1.1 PURPOSES AND OBJECTIVES:

The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images.

One of the most important issues in image segmentation based on a region of interest (ROI) is how to decide upon a semantic object region according to a specific purpose. Computer experimental results show that the proposed model can successfully segment an ROI boundary in natural scenes and computer graphics.

Mask R-CNN was built using Faster R-CNN. While Faster R-CNN has 2 outputs for each candidate object, a class label and a bounding-box offset, Mask R-CNN is the addition of a third branch that outputs the object mask. The additional mask output is distinct from the class and box outputs, requiring the extraction of a much finer spatial layout of an object.

Mask R-CNN is an extension of Faster R-CNN and works by adding a branch for predicting an object mask (Region of Interest) in parallel with the existing branch for bounding box recognition.

1.2 EXISTING SYSTEM:

Image segmentation originally started from Digital Image Processing coupled with optimization algorithms.

Thresholding

Thresholding is one of the easiest methods of image segmentation where a threshold is set for dividing pixels into two classes. Pixels that have values greater than the threshold value are set to 1 while pixels with values lesser than the threshold value are set to 0.

The image is thus converted into a binary map, resulting in the process often termed binarization. Image thresholding is very useful in case the difference in pixel values between the two target classes is very high, and it is easy to choose an average value as the threshold.

Edge Segmentation

Edge segmentation, also called edge detection, is the task of detecting edges in images. From a segmentation-based viewpoint, we can say that edge detection corresponds to classifying which pixels in an image are edge pixels and singling out those edge pixels under a separate class correspondingly.

Clustering Based Segmentation

Clustering algorithms perform better than their counterparts and can provide reasonably good segments in a small amount of time. Popular algorithms like the K-means clustering algorithms are unsupervised algorithms that work by clustering pixels with common attributes together as belonging to a particular segment.

K-means clustering, in particular, takes all the pixels into consideration and clusters them into “k” classes. Differing from region-growing methods, clustering-based methods do not need a seed point to start segmenting from.

1.2.1 Disadvantages of existing System

- Threshold selection is not always straightforward.
- When using edge segmentation, if two or more objects are overlapped the edge computation becomes hard.
- In clustering if the objects are overlapped the clusters cannot be divided properly.

1.2.2 PROPOSED SYSTEM:

The target of our project is to apply the Mask R CNN algorithm for image segmentation.

There are three levels of image analysis: Classification, Object detection segmentation.

Classification: categorizing the entire image into class such as animals, humans, objects.

Object detection: detecting objects in an image and drawing a rectangle around them.

Segmentation: Identifying the parts of the image and understanding what object it belongs to.

Mask R-CNN is a state-of-the-art deep neural network architecture used for image segmentation. Using Mask R-CNN, we can automatically compute pixel-wise masks for objects in the image, allowing us to segment the foreground from the background.

Mask R-CNN, can automatically predict both the bounding box and the pixel-wise segmentation mask of each object in an input image. The downside is that masks produced by Mask R-CNN aren't always "clean" — there is typically a bit of background that "bleeds" into the foreground segmentation.

1.2.3 Advantages of Proposed System

- **Simplicity:** Mask R-CNN is simple to train.
- **Performance:** Mask R-CNN outperforms all existing, single-model entries on every task.
- **Efficiency:** The method is very efficient and adds only a small overhead to Faster R-CNN.
- **Flexibility:** Mask R-CNN is easy to generalize to other tasks.

1.3 SCOPE OF PROJECT:

The future of image processing will involve scanning the heavens for other intelligent life out in space. Also new intelligent, digital species created entirely by research scientists in various nations of the world will include advances in image processing applications. Due to advances in image processing and related technologies there will be millions and millions of robots in the world in a few decades time, transforming the way the world is managed. Advances in image processing and artificial intelligence will involve spoken commands, anticipating the information requirements of governments, translating languages, recognizing and tracking people and things, diagnosing medical conditions, performing surgery, reprogramming defects in human DNA, and automatic driving all forms of transport. With increasing power and sophistication of modern computing, the concept of computation can go beyond the present limits and in future, image processing technology will advance and the visual system of man can be replicated.

The future trend in remote sensing will be towards improved sensors that record the same scene in many spectral channels. Graphics data is becoming increasingly important in image

processing applications. The future image processing applications of satellite-based imaging ranges from planetary exploration to surveillance applications.

Using large scale homogeneous cellular arrays of simple circuits to perform image processing tasks and to demonstrate pattern-forming phenomena is an emerging topic. The cellular neural network is an implementable alternative to fully connected neural networks and has evolved into a paradigm for future imaging techniques. The usefulness of this technique has applications in the areas of silicon retina, pattern formation, etc.

2. LITERATURE SURVEY

Author: Mr V. Neethi Devan Assistant professor, Mepco Schlenk College

Paper: Image Segmentation for object Detection using Mask R-CNN in colab

In the last few years researches across the global applied deep learning concept in computer vision applications. The authors addressed the problems in using two neural networks architecture LeNet and network in Network (NN) performance of the architectures to study computational efficiency by using classification and detection problems. They used multiple databases.

The recent development in Deep Learning made more progress in research activities in Digital Image Processing. The authors analyzed the various pros and cons of each approach. The focus is to promote knowledge of classical computer vision techniques. Also exploring how other options of computer vision can be combined. Many hybrid methodologies are studied and proved to improve computer vision performance.

To process big data applications, a large amount of space is needed in industry. Also, more space is required for the video streams from CCTV cameras and social media data, sensor data, agriculture data, medical data and data evolved from space research. The authors conducted a survey which starts from object recognition, action recognition, crowd analysis and finally violence detection in a crowd environment. The various problems in the existing methods were identified and summarized.

Author: Dr. G. Chandrashekar

Papers: Image segmentation in Mask R-CNN, Global Research and development Journal Publication

The authors used an approach on how to use Deep learning for all types of well-known applications such as Speech recognition, Image processing and NLP. In Deep learning, a pre-trained neural network is used to identify and remove noise from images. The processing of images using deep learning is processed for image pre-processing and image augmentation for Various applications with better results.

The authors used the latest image classification techniques based on deep neural network architectures to improve the identification of highly boosted electroweak particles with respect to existing methods. Also, they introduced new methods to visualize and interpret the high-level features learned by deep neural networks that provide discrimination beyond physics-derived variables, adding a new capability to understand physics and to design more powerful classification methods at the LHC.

The authors proposed an analysis of tracking-by-detection approach which includes detection by YOLO and tracking by SORT algorithm. This paper has information about a custom image dataset being trained for 6 specific classes using YOLO and this model is being used in videos for tracking by SORT algorithm. Recognizing a vehicle or pedestrian in an ongoing video is helpful for traffic analysis. The goal of this paper is for analysis and knowledge of the domain.

Authors: Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, Demetri Terzopoulos

Paper: Image Segmentation Using Deep Learning

Image segmentation is a key topic in image processing and computer vision with applications such as scene understanding, medical image analysis, robotic perception, video surveillance, augmented reality, and image compression, among many others. Various algorithms for image segmentation have been developed in the literature. Recently, due to the success of deep learning models in a wide range of vision applications, there has been a substantial amount of works aimed at developing image segmentation approaches using deep learning models. In this survey, we provide a comprehensive review of the literature at the time of this writing, covering a broad spectrum of pioneering works for semantic and instance-level segmentation, including fully convolutional pixel-labeling networks, encoder-decoder architectures, multi-scale and pyramid-based approaches, recurrent networks, visual attention models, and generative models in adversarial settings. We investigate the similarity, strengths and challenges of these deep learning models, examine the most widely used datasets, report performances, and discuss promising future research directions in this area.

3. SYSTEM ANALYSIS

The development and deployment of the application require the following general and specific minimum requirements for hardware:

3.1 HARDWARE REQUIREMENTS:

- **CPU:** 1.5GHZ and above
- **RAM:** Up to 8GB
- **PROCESSOR:** I5 or above

The development and deployment of the application require the following general and specific minimum requirements for software:

SOFTWARE REQUIREMENTS:

- **OS:** Windows (7,8, XP)
- **PYTHON:** Version 3.6
- Google colab

3.2 SOFTWARE REQUIREMENTS SPECIFICATION:

Input Design:

Input design connects the data given by the user through types of data and the trained model which is trained using the classification algorithm that gives us the corresponding recruitment prediction about the various objects in the image.

Objectives:

The objective is to build a prediction detection system using convolution neural networks which recognize various posts and convert them into categories.

1. To improve upon the previous implementation.
2. Translate the categories of posts.
3. Create Real-Time or Near-Real-Time Application.
4. Achieve partial background and illumination independence.

Output Design:

The quality output is one, which meets the requirements of the end user and presents the text clearly. The output which is given will achieve the partial background and illumination

independence, so the accuracy through this model will be increased.

FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, and outputs. This approach is applied to estimate the calorie count based on the keywords that are present in the text given by the end user. The results are reported on highly challenging categories of the ingredients that have not yet been considered in previous works. We provide comprehensive analysis of the text processing method and show below lesser error rates with the proposed approach, which is sufficient for deployment in practical applications.

NON-FUNCTIONAL REQUIREMENTS:

The major non-functional Requirements of the system are as follows

Usability The system is designed with a completely automated process hence there is no or less user intervention.

Reliability The system is more reliable because of the qualities that are inherited from the chosen platform python. The code built by using python is more reliable.

Performance This system is developing in the high-level languages and using the advanced front end and back-end technologies it will give response to the end user on the client system within very less time.

Supportability The system is designed to be cross platform supportable. The system is supported on a wide range of hardware and any software platform.

Following are the features of a good SRS document:

1. **Correctness:** User review is used to provide the accuracy of requirements stated in the SRS. SRS is said to be perfect if it covers all the needs that are truly expected from the system.
2. **Completeness:** The SRS is complete if, and only if, it includes the following elements: All essential requirements, whether relating to functionality, performance, design, constraints, attributes, or external interfaces. Definition of their responses of the software to all realizable classes of input data in all available categories of situations. Full labels and references to all figures, tables, and diagrams in the SRS and definitions of all terms and units of measure.
3. **Consistency:** The SRS is consistent if, and only if, no subset of individual requirements described in its conflict. There are three types of possible conflict in the SRS: The specified characteristics of real-world objects may conflict. For example, (a) The format of an output report may be described in one requirement as tabular but in another as textual. (b) One condition may state that all lights shall be green while another states that all lights shall be blue.

4. **Unambiguousness:** SRS is unambiguous when every fixed requirement has only one interpretation. This suggests that each element is uniquely interpreted. In case there is a method used with multiple definitions, the requirements report should determine the implications in the SRS so that it is clear and simple to understand.

5. **Ranking for importance and stability:** The SRS is ranked for importance and stability if each requirement in it has an identifier to indicate either the significance or stability of that particular requirement. 20 Typically, all requirements are not equally important. Some prerequisites may be essential, especially for life-critical applications, while others may be desirable. Each element should be identified to make these differences clear and explicit. Another way to rank requirements is to distinguish classes of items as essential, conditional, and optional.

6. **Modifiability:** SRS should be made as modifiable as likely and should be capable of quickly obtain changes to the system to some extent. Modifications should be perfectly indexed and cross-referenced.

7. **Verifiability:** SRS is correct when the specified requirements can be verified with a cost effective system to check whether the final software meets those requirements. The requirements are verified with the help of reviews.

8. **Traceability:** The SRS is traceable if the origin of each of the requirements is clear and if it facilitates the referencing of each condition in future development or enhancement documentation.

There are two types of Traceability:

1. **Backward Traceability:** This depends upon each requirement explicitly referencing its source in earlier documents.

2. **Forward Traceability:** This depends upon each element in the SRS having a unique name or reference number. The forward traceability of the SRS is especially crucial when a software product enters the operation and maintenance phase. As code and design documents are

modified, it is necessary to be able to ascertain the complete set of requirements that may be concerned by those modifications.

3. Design Independence: There should be an option to select from multiple design alternatives for the final system. More specifically, the SRS should not contain any implementation details.

4. Testability: An SRS should be written in such a method that it is simple to generate test cases and test plans from the report. 21

5. Understandable by the customer: An end user may be an expert in his/her explicit domain but might not be trained in computer science. Hence, the purpose of formal notations and symbols should be avoided to as much extent as possible. The language should be kept simple and clear.

6. Properties of a good SRS document: Concise: The SRS report should be concise and at the same time, unambiguous, consistent, and complete. Verbose and irrelevant descriptions decrease readability and also increase error possibilities.

7. Structured: It should be well-structured. A well-structured document is simple to understand and modify. In practice, the SRS document undergoes several revisions to cope up with the user requirements. Often, user requirements evolve over a period of time. Therefore, to make the modifications to the SRS document are easy, it is vital to make the report well-structured.

8. Black-box view: It should only define what the system should do and refrain from stating how to do these. This means that the SRS document should define the external behavior of the system and not discuss the implementation issues. The SRS report should view the system to be developed as a black box and should define the externally visible behavior of the system. For this reason, the SRS report is also known as the black-box specification of a system.

9. Conceptual integrity: It should show conceptual integrity so that the reader can merely understand it. Response to undesired events: It should characterize acceptable responses to unwanted events. These are called system responses to exceptional conditions.

10.Verifiable: All requirements of the system, as documented in the SRS document, should be correct. This means that it should be possible to decide whether or not requirements have been met in an implementation. In this, the possible outcome of a software system which includes effects due to operation of the program is fully explained. All functional requirements which may include calculations, data processing, etc. are placed in a ranked order.

Modules:

- 1.Data Collection**
- 2.Data Pre-Processing**
- 3.Working Procedure**
- 4. Evaluation**

DATA COLLECTION:

Data used in this paper is a set of records. This step is concerned with selecting the subset of all available data that you will be working with. ML problems start with data preferably, lots of data (examples or observations) for which you already know the target answer. Data for which you already know the target answer is called labeled data.

Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes. Data collection is a research component in all study fields, including physical and social sciences, humanities, and business. While methods vary by discipline, the emphasis on ensuring accurate and honest collection remains the same. The goal for all data collection is to capture quality evidence that allows analysis to lead to the formulation of convincing and credible answers to the questions that have been posed. Data collection and validation consists of four steps when it involves taking a census and seven steps when it involves sampling.

DATA PRE-PROCESSING:

Organize your selected data by formatting, cleaning and sampling from it.

Three common data pre-processing steps are:

- Formatting
- Cleaning
- Sampling

Formatting:

The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file.

Cleaning:

Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.

Sampling:

There may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. You can take a smaller representative sample of the selected.

Data preprocessing can refer to manipulation or dropping of data before it is used in order to ensure or enhance performance and is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Data-gathering methods are often loosely controlled, resulting in out-of-range values (e.g., Income: -100), impossible data combinations.

Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running any analysis. Often, data preprocessing is the most important phase of a machine learning project, especially in computational biology.

Software Environment:

Implementation on (Python): Script: As yet, I encompass alert on intelligent program ability of Python. This is precious capability to permit you to sort in program as well as to encompass it and execute rapidly in intellectual mode.

Scripts are reusable: Essentially, content is a book record containing the explanation to comprise a Python program. Whenever you encompass prepared the contented, you preserve effect it again as well as again lacking have to retype it each instance. Scripts are editable: Maybe, more appreciably, you preserve make assorted variant of content via altering the assertion initial through one file onto then utilizing a word processor. Then, to tip you preserve and execute each one of the individual adaptations. Thus, it is not tricky to construct assorted projects through a base compute of compose. You will need a text editor: Pretty much any word processor resolve acquires the job done for making Python script credentials.

You preserve utilize Microsoft Notepad, Microsoft WordPad, Microsoft Word, otherwise pretty much any word mainframe if you desire to. Difference between a script and a program Script: Script be meticulous as of core code of application, which is frequently printed in alternate lingo, as well as be recurrently finished otherwise possibly tainted via the end consumer. Stuffing be repeatedly deciphered as of source code otherwise byte code, whereas application they manage be habitually prearranged to local machine code.

Program: The program has executable structure to PC preserve utilize basically to affect the strategy. Comparable program in rational source code structure, as of which executable project be derived (e.g., assemble)

Classification Algorithms:

Classification algorithms is a system that can learn from example through self improvement and without being explicitly coded by programmers. The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results. Classification algorithms combine data with statistical tools to predict an output. This output is then used by corporations to make actionable insights. Classification algorithms are closely related to data mining and Bayesian predictive modeling. The machine receives data as input, and uses an algorithm to formulate answers.

A typical Classification algorithms task is to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendations. Classification algorithms are also used for a variety of tasks

like fraud detection, predictive maintenance, portfolio optimization, automatizing tasks and so on.

Classification algorithms vs. Traditional Programming

Traditional programming differs significantly from Classification algorithms. In traditional programming, a programmer codes all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.

Classification algorithms working:

Classification algorithms is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if it feeds previously unseen examples, the machine has difficulties to predict.

The core objective of Classification algorithms is the learning and inference. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the data. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a feature vector. You can think of a feature vector as a subset of data that is used to tackle a problem. The machine uses some fancy algorithms to simplify the reality and transform this discovery into a model. Therefore, the learning stage is used to describe the data and summarize it into a model.

For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model.

Inferring:

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of Classification algorithms. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.

The life of Classification algorithms programs is straightforward and can be summarized

in the following points:

1. Define a question
2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm
8. Loop 4-7 until the results are satisfying
9. Use the model to make a prediction

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.

PYTHON OVERVIEW

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

Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language: Python is a great language for the beginner level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

History of Python:

Python was shaped via Guido van Rossum in last fraction of eighties as well as middle nineties at National explore institution pro math as well as Computer Science in Netherland. Python has numerous dissimilar dialects, counting ABC, Modula-3, C, C++, Algol68, SmallTalk, as well as Unix shell plus other pre-arranged dialects. Python is secluded. Like Perl, Python source code is presently accessible beneath the GNU. General Public License (GPL). Python is currently reserved via a hub upgrading cluster at the institution, despite the fact that Guido van Rossum truly holds an indispensable job in coordinating it.

Python's standard library:

- Pandas
- Numpy
- Sklearn
- seaborn
- matplotlib
- Importing Datasets

PANDAS:

Pandas is quite a game changer when it comes to analyzing data with Python and it is one of the most preferred and widely used tools in data munging/wrangling, if not the most used one. Pandas is open source. What's cool about Pandas is that it takes data (like a CSV or TSV file, or a SQL database) and creates a Python object with rows and columns called data frame that looks very similar to tables in statistical software (think Excel or SPSS for example. People who are familiar with R would see similarities to R too). This is so much easier to work with in comparison to working with lists and/or dictionaries through for loops or list comprehension.

Installation and Getting Started:

In order to “get” Pandas you would need to install it. You would also need to have Python 2.7 and above as a pre-requirement for installation. It is also dependent on other libraries (like NumPy) and has optional dependencies (like Matplotlib for plotting). Therefore,

I think that the easiest way to get Pandas set up is to install it through a package like the Anaconda distribution , “a cross platform distribution for data analysis and scientific computing.” In order to use Pandas in your Python IDE (Integrated Development Environment) like Jupyter Notebook or Spyder (both of them come with Anaconda by default), you need to import the Pandas library first. Importing a library means loading it into the memory and then it’s there for you to work with. In order to import Pandas all you have to do is run the following code:

- **import pandas as pd**
- **import numpy as np**

Usually you would add the second part (‘as pd’) so you can access Pandas with ‘pd.command’ instead of needing to write ‘pandas.command’ every time you need to use it. Also, you would import numpy as well, because it is a very useful library for scientific computing with Python. Now Pandas is ready for use! Remember, you would need to do it every time you start a new Jupyter Notebook, Spyder file etc.

Working with Pandas:

Loading and Saving Data with Pandas

When you want to use Pandas for data analysis, you’ll usually use it in one of three different ways:

- Convert a Python’s list, dictionary or Numpy array to a Pandas data frame
- Open a local file using Pandas, usually a CSV file, but could also be a delimited text file (like TSV), Excel, etc
- Open a remote file or database like a CSV or a JSON on a website through a URL or read from a SQL table/database.

4. SYSTEM DESIGN

Design is concerned with identifying software components specifying relationships among components. Specifying structure and providing a blueprint 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 and clearly specified.

Design will explain software components in detail. This will help the implementation of the system. Moreover, this will guide the further changes in the system to satisfy the future requirements. 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.

The design of a system is perhaps the most critical factor affecting the quality of the software and has a major impact on the later phases, particularly testing and maintenance. The output of this phase is the design document. This document is similar to a blueprint or plan for the solution, and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phase system design and detail design. System design, which is sometimes 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.

During detailed design the internal logic of the modules specified in system design is decided. During this phase further details of the data structures and algorithmic design of each of the modules is specified. The logic 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 the 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 words, 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.

4.1 SYSTEM ARCHITECTURE:

Architecture diagram is a diagram of a system, in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. The block diagram is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation.

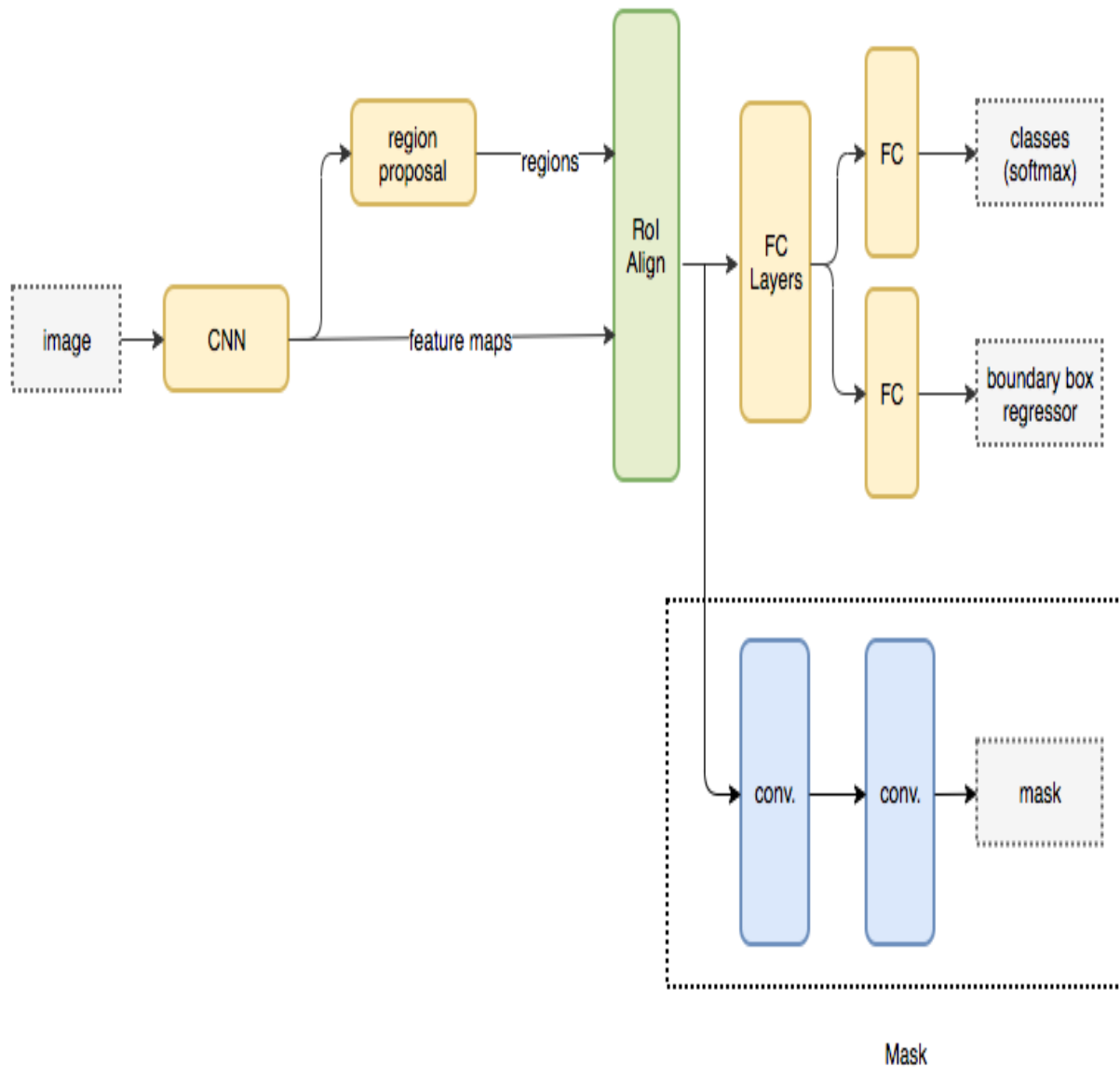


Fig:4.1.1 System Architecture

4.2 UML DIAGRAMS:

The unified modeling is a standard language for specifying, visualizing, constructing and documenting the system and its components is a graphical language which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure and control information about the systems.

Depending on the development culture, some of these artifacts are treated more or less formally than others. Such artifacts are not only the deliverables of a project; they are also critical in controlling, measuring, and communicating about a system during its development and after its deployment.

The UML addresses the documentation of a system's architecture and all of its details. The UML also provides a language for expressing requirements and for tests. Finally, the UML provides a language for modeling the activities of project planning and release management.

Building blocks of UML:

- Things
- Relationships
- Diagrams

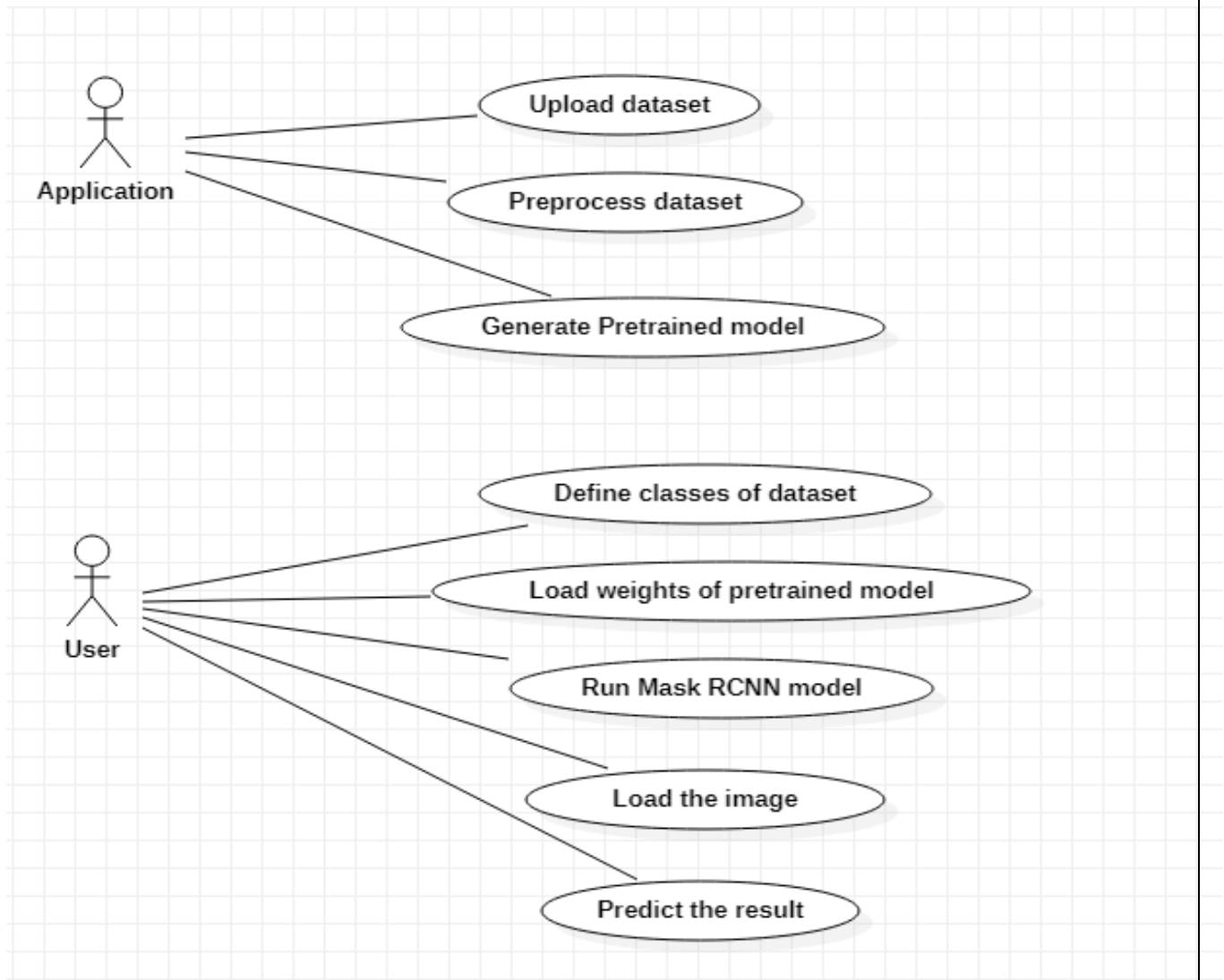
Things in UML:

There are four kinds of things in the UML:

- Structural things
- Behavioral things
- Grouping things
- Annotational things

Use case diagram:

By these we want to show the things which are associated with our model and the relationship between those things. It shows how elements are associated with each other and this association describes the functionality of our application.

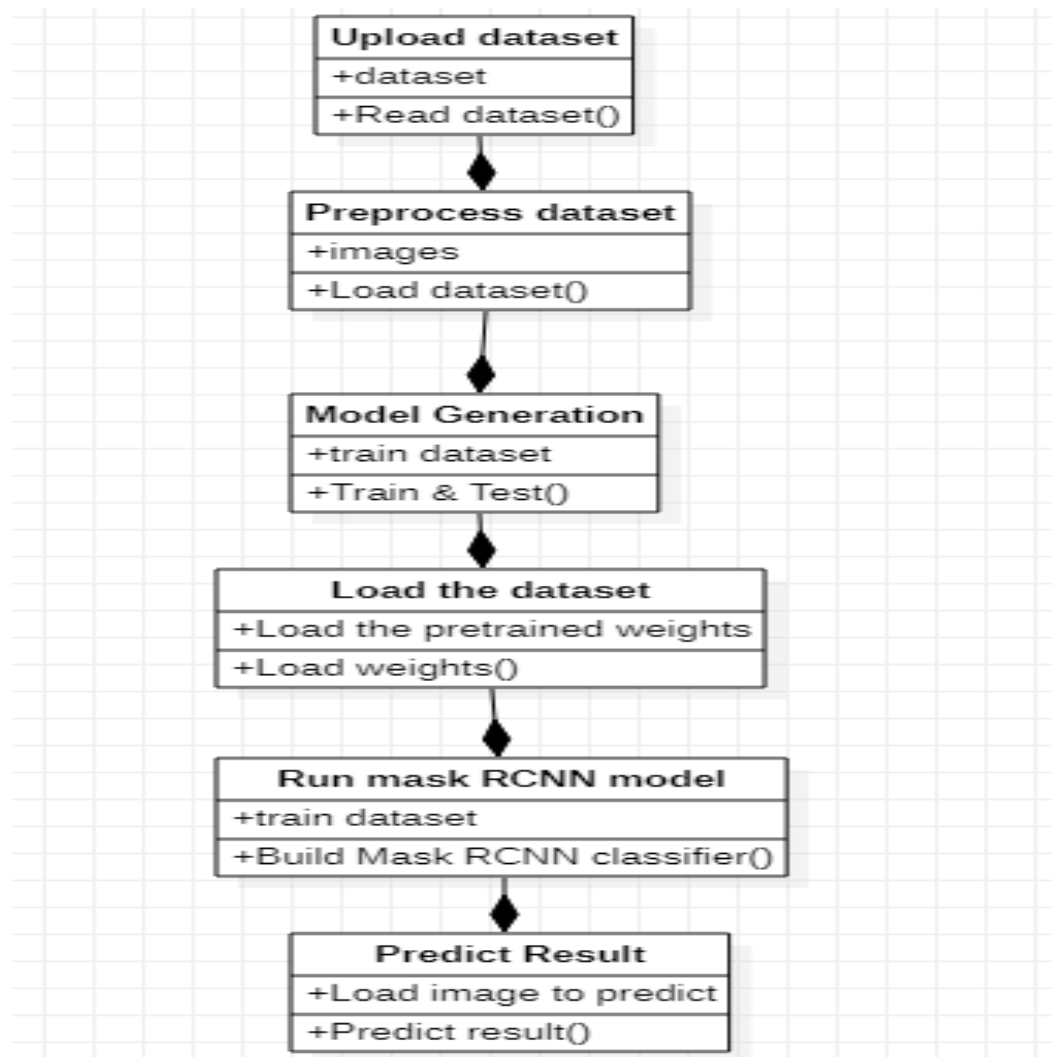


Class diagram:

A class is a set of objects that share a common structure and common behavior (the same attributes, operations, relationships and semantics). A class is an abstraction of real-world items.

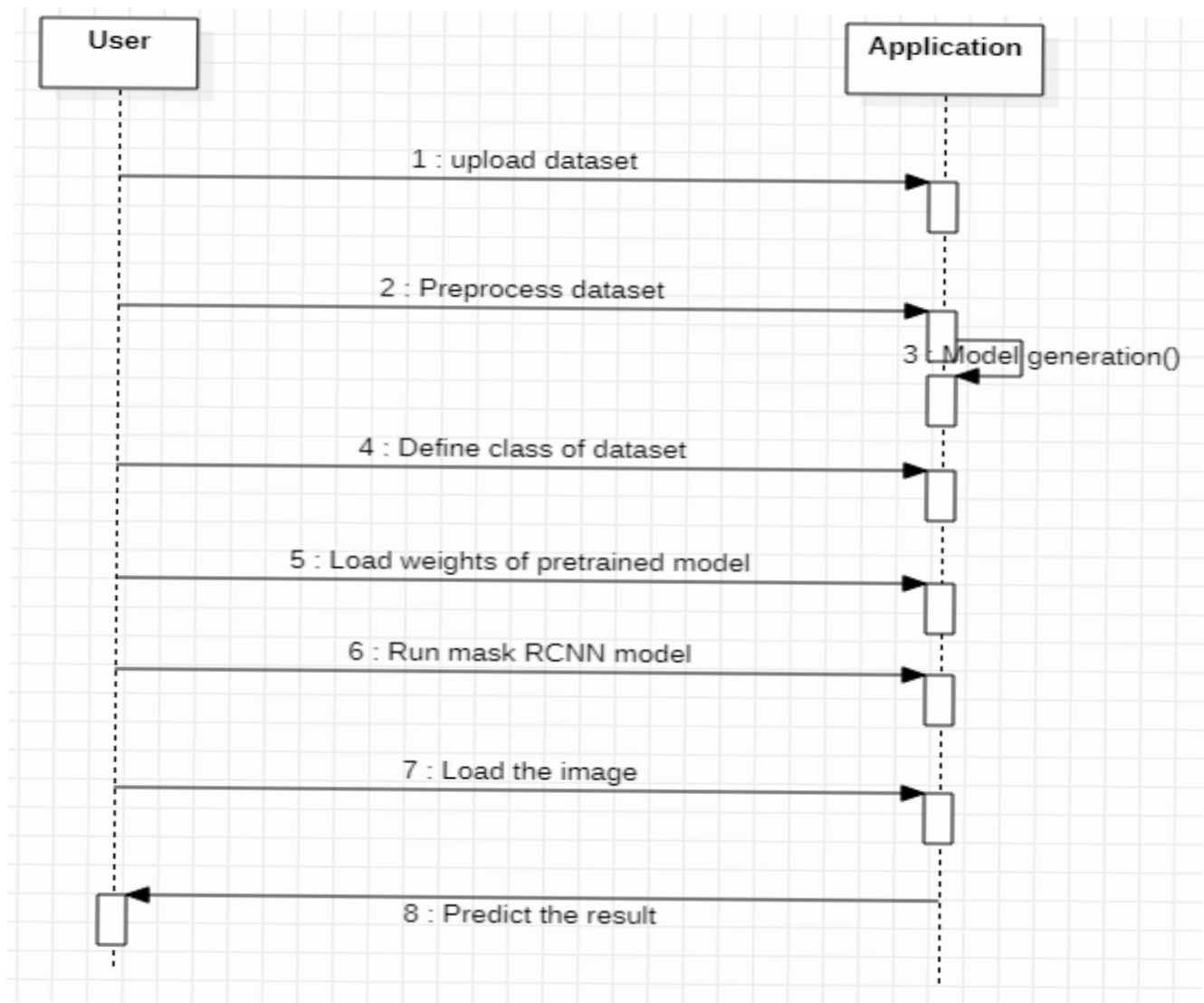
There are 4 approaches for identifying classes:

- Noun phrase approach:
- Common class pattern approach.
- Use case Driven Sequence or Collaboration approach.
- Classes, Responsibilities and collaborators Approach



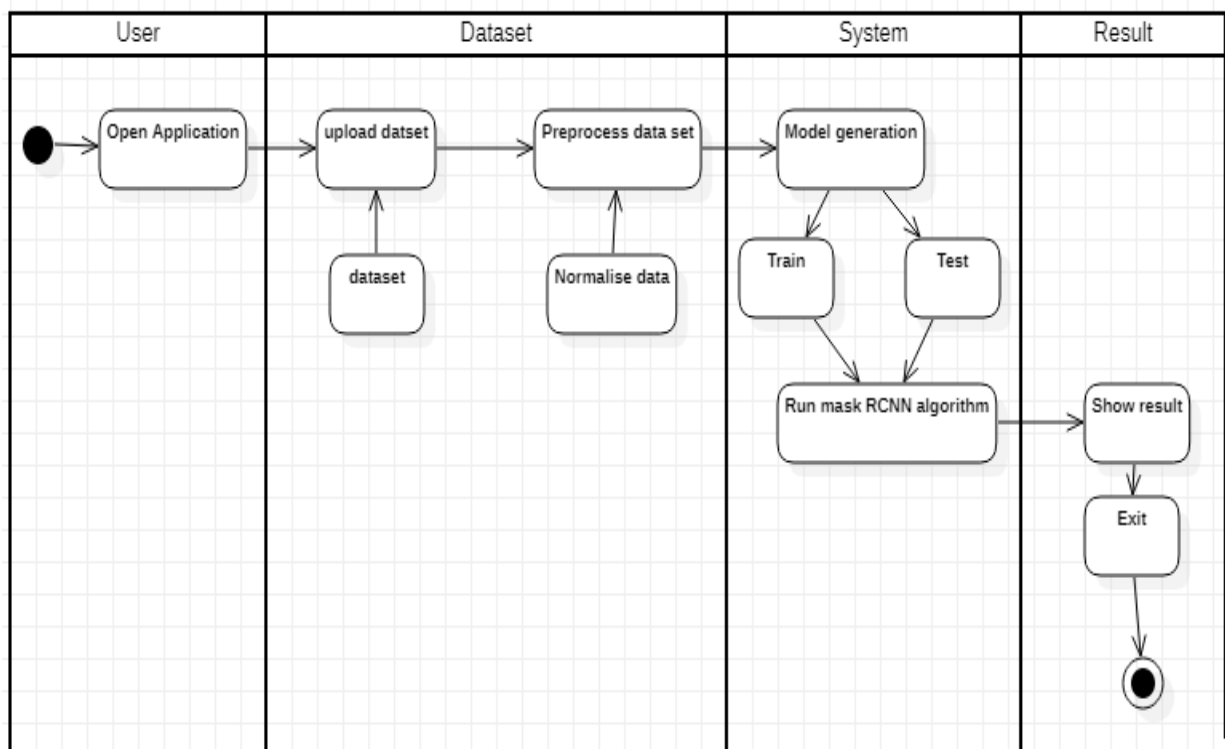
Sequence diagram:

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.



Activity diagram:

An activity diagram is essentially a flowchart, showing flow of control from activity to activity. We use activity diagrams to model the dynamic aspects of a system. For the most part, this involves modeling the sequential (and possibly concurrent) steps in a computational process. With an activity diagram. We can also model the flow of an object as it moves from state to state at different points in the flow of control.



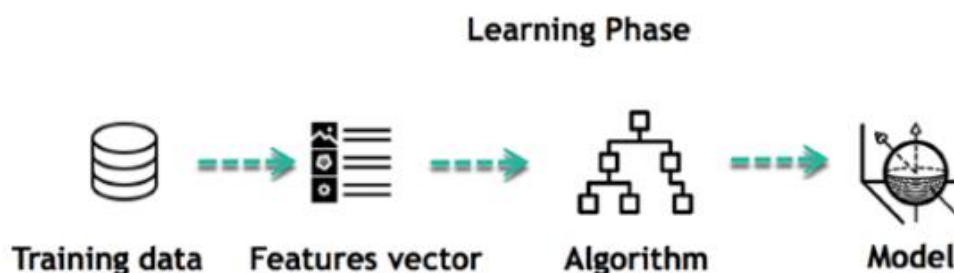
5 METHODOLOGY

5.1 TECHNOLOGIES USED:

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if it's feed a previously unseen example, the machine has difficulties to predict.

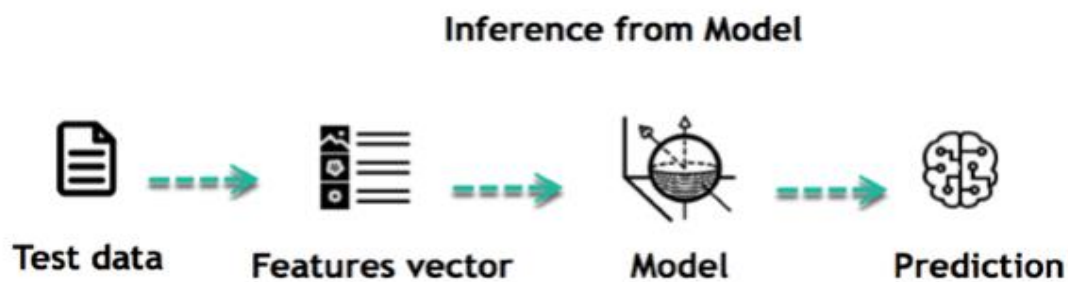
The core objective of machine learning is the learning and inference. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the data. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a feature vector. You can think of a feature vector as a subset of data that is used to tackle a problem.

The machine uses some fancy algorithms to simplify the reality and transform this discovery into a model. Therefore, the learning stage is used to describe the data and summarize it into a model.



For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model

Inferring When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.



The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question
2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm
8. Loop 4-7 until the results are satisfying
9. Use the model to make a prediction

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.

Google Colab:

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

Colab, or "Colaboratory", allows you to write and execute Python in your browser, with

- Zero configuration required
- Access to GPUs free of charge
- Easy sharing

Whether you're a student, a data scientist or an AI researcher, Colab can make your work easier.

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

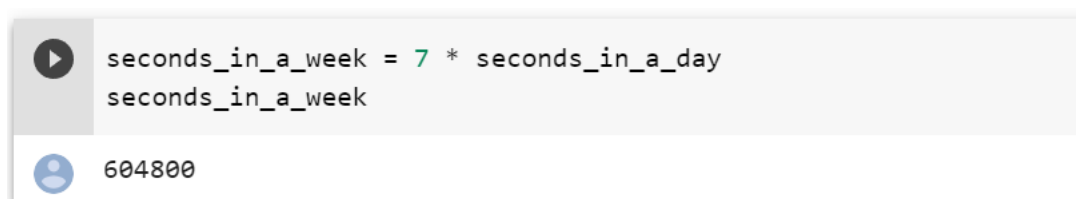
For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:



A screenshot of a Colab code cell. It features a play button icon on the left. The code is: `seconds_in_a_day = 24 * 60 * 60` followed by `seconds_in_a_day` on a new line. Below the code, there is a user icon and the output value `86400`.

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/CTRL Enter". To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:



A screenshot of a Colab code cell. It features a play button icon on the left. The code is: `seconds_in_a_week = 7 * seconds_in_a_day` followed by `seconds_in_a_week` on a new line. Below the code, there is a user icon and the output value `604800`.

Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them. To create a new Colab notebook you can use the File menu above, or use the following link: [create a new Colab notebook] (<http://colab.research.google.com#create=true>).

Colab notebooks are Jupyter notebooks that are hosted by Colab. To learn more about the Jupyter project, see [jupyter.org] (<https://www.jupyter.org>).

Jupyter

Jupyter Notebook (formerly IPython Notebooks) is a web-based_interactive computational environment for creating notebook documents.

A Jupyter Notebook document is a browser-based REPL containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media. Underneath the interface, a notebook is a JSON document, following a versioned schema, usually ending with the ".ipynb" extension.

Jupyter notebooks are built upon a number of popular open-source libraries:

- IPython
- ZeroMQ
- Tornado
- jQuery
- Bootstrap (front-end framework)
- MathJax

Jupyter Notebook can connect to many kernels to allow programming in different languages. A Jupyter kernel is a program responsible for handling various types of requests (code execution, code completions, inspection), and providing a reply. Kernels talk to the other components of Jupyter using ZeroMQ, and thus can be on the same or remote machines. Unlike many other Notebook-like interfaces, in Jupyter, kernels are not aware that they are attached to a specific document, and can be connected to many clients at once. Usually, kernels allow execution of only a single language, but there are a couple of exceptions. By default Jupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), there are 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.

A Jupyter Notebook can be converted to a number of open standard output formats(HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library^[15] or "jupyter

nbconvert" command line interface in a shell. To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library^[16] is provided as a service through NbViewer^[17] which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Numpy:

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

SciPy:

SciPy in Python is an open-source library used for solving mathematical, scientific, engineering, and technical problems. It allows users to manipulate the data and visualize the data using a wide range of high-level Python commands. SciPy is built on the Python NumPy extension. SciPy is also pronounced as "Sigh Pi."

Pillow:

The Pillow module provides the open() and show() function to read and display the image respectively. For displaying the image Pillow first converts the image to a .png format (on Windows OS) and stores it in a temporary buffer and then displays it. Therefore, due to the conversion of the image format to .png some properties of the original image file format might be lost (like animation). Therefore, it is advised to use this method only for test purposes.

cython:

The Cython language is a superset of the Python language that additionally supports calling C functions and declaring C types on variables and class attributes. This allows the compiler to generate very efficient C code from Cython code. The C code is generated once and then compiles with all major C/C++ compilers in CPython 2.6, 2.7 (2.4+ with Cython 0.20.x) as well as 3.3 and all later versions. We regularly run integration tests against all supported

CPython versions and their latest in-development branches to make sure that the generated code stays widely compatible and well adapted to each version. PyPy support is work in progress (on both sides) and is considered mostly usable since Cython 0.17. The latest PyPy version is always recommended here.

matplotlib:

Matplotlib is a low-level graph plotting library in python that serves as a visualization utility. Matplotlib was created by John D. Hunter. Matplotlib is open source and we can use it freely. Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

scikit-image:

scikit-image is an image processing Python package that works with NumPy arrays which is a collection of algorithms for image processing. Let's discuss how to deal with images into set of information and it's some application in the real world. Simple and efficient tools for image processing and computer vision techniques. Accessible to everybody and reusable in various contexts. Built on the top of NumPy, SciPy, and matplotlib.

TensorFlow>=1.3.0:

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

Keras>=2.0.8:

Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination.

Opencv-python:

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

h5py:

The h5py package provides both a high- and low-level interface to the HDF5 library from Python. The low-level interface is intended to be a complete wrapping of the HDF5 API, while the high-level component supports access to HDF5 files, datasets and groups using established Python and NumPy concepts. A strong emphasis on automatic conversion between Python (NumPy) datatypes and data structures and their HDF5 equivalents vastly simplifies the process of reading and writing data from Python.

imgaug:

A library for image augmentation in machine learning experiments, particularly convolutional neural networks. Supports the augmentation of images, keypoints/landmarks, bounding boxes, heatmaps and segmentation maps in a variety of different ways.

IPython:

IPython provides a rich toolkit to help you make the most out of using Python interactively. A powerful interactive Python shell. A Jupyter kernel to work with Python code in Jupyter notebooks and other interactive frontend.

5.2 MODULE DESCRIPTION

Step 1: Install the dependencies

Install all the required dependencies.

Step 2: Download the pre-trained weights (trained on MS COCO)

Next, we need to download the pretrained weights. These weights are obtained from a model that was trained on the MS COCO dataset. Once you have downloaded the weights, paste this file in the samples folder of the Mask_RCNN repository that we cloned in step 1.

Step 3: Load the weights

Load the pre-trained weights downloaded on step 2.

Step 4: Predicting for our image

Finally, we will use the Mask R-CNN architecture and the pretrained weights to generate predictions for our own images.

Once you're done with these four steps, it's time to jump into your Jupyter Notebook! We will implement all these things in Python and then generate the masks along with the classes and bounding boxes for objects in our images.

5.3 ALGORITHM

Mask R-CNN is a Convolutional Neural Network (CNN) and state-of-the-art in terms of image segmentation. This variant of a Deep Neural Network detects objects in an image and generates a high-quality segmentation mask for each instance.

In this article, I will provide a simple and high-level overview of Mask R-CNN. Then, we will discuss the basic concepts required to understand what Mask R-CNN is and how it works:

1. Convolutional Neural Networks (CNN)
2. Region-Based Convolutional Neural Networks (R-CNN)
3. Faster R-CNN with Region Proposal Networks (RPN)
4. Mask R-CNN and how it works
5. Example projects and applications

Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a type of artificial neural network used in image recognition and processing that is optimized to process pixel data. Therefore, Convolutional Neural Networks are the fundamental and basic building blocks for the computer vision task of image segmentation (CNN segmentation).

The Convolutional Neural Network Architecture consists of three main layers:

1. Convolutional layer: This layer helps to abstract the input image as a feature map via the use of filters and kernels.
2. Pooling layer: This layer helps to down sample feature maps by summarizing the presence of features in patches of the feature map.
3. Fully connected layer: Fully connected layers connect every neuron in one layer to every neuron in another layer.

Combining the layers of a CNN enables the designed neural network to learn how to identify and recognize the object of interest in an image. Simple Convolutional Neural Networks are built for image classification and object detection with a single object in the image.

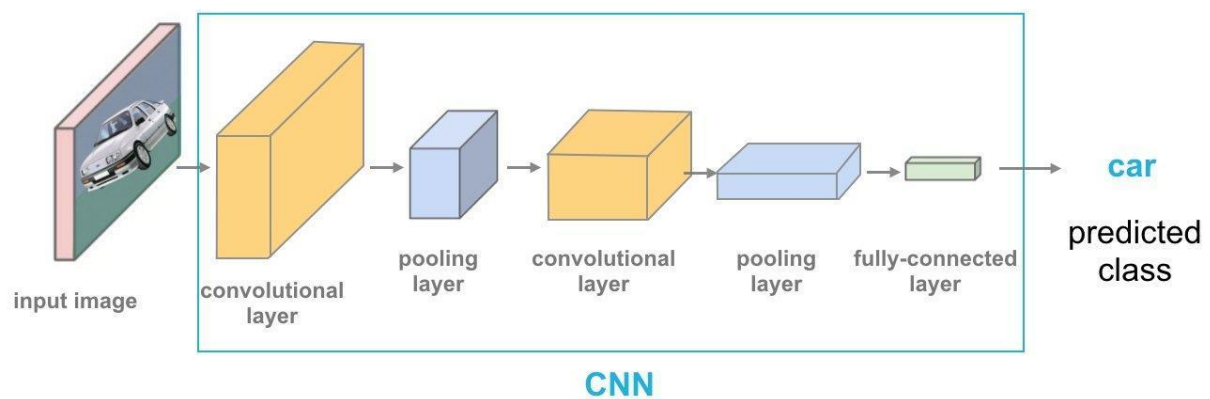


Fig: 5.3.1 Convolutional Neural Network

Concept of the CNN architecture: How a convolutional neural network works.

Convolutional layers:

In a CNN, the input is a tensor with a shape: (number of inputs) x (input height) x (input width) x (input channels). After passing through a convolutional layer, the image becomes abstracted

to a feature map, also called an activation map, with shape: (number of inputs) x (feature map height) x (feature map width) x (feature map channels).

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus.^[14] Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features and classify data, this architecture is generally impractical for larger inputs such as high-resolution images. It would require a very high number of neurons, even in a shallow architecture, due to the large input size of images, where each pixel is a relevant input feature. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for each neuron in the second layer. Instead, convolution reduces the number of free parameters, allowing the network to be deeper.^[15] For example, regardless of image size, using a 5 x 5 tiling region, each with the same shared weights, requires only 25 learnable parameters. Using regularized weights over fewer parameters avoids the vanishing gradients and exploding gradients problems seen during backpropagation in traditional neural networks. Furthermore, convolutional neural networks are ideal for data with a grid-like topology (such as images) as spatial relations between separate features are taken into account during convolution and/or pooling.

Pooling layers:

Convolutional networks may include local and/or global pooling layers along with traditional convolutional layers. Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, tiling sizes such as 2 x 2 are commonly used. Global pooling acts on all the neurons of the feature map.^{[18][19]} There are two common types of pooling in popular use: max and average. *Max pooling* uses the maximum value of each local cluster of neurons in the feature map,^{[20][21]} while *average pooling* takes the average value.

Fully connected layers:

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multilayer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

Weights:

Each neuron in a neural network computes an output value by applying a specific function to the input values received from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning consists of iteratively adjusting these biases and weights.

The vector of weights and the bias are called *filters* and represent particular features of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces the memory footprint because a single bias and a single vector of weights are used across all receptive fields that share that filter, as opposed to each receptive field having its own bias and vector weighting.

In a more complex situation with multiple objects in an image, a simple CNN architecture isn't optimal. For those situations, Mask R-CNN is a state-of-the-art architecture that is based on R-CNN (also referred to as RCNN).

R CNN

R-CNN or RCNN, stands for Region-Based Convolutional Neural Network, it is a type of machine learning model that is used for computer vision tasks, specifically for object detection.

The RCNN architecture was designed to solve image detection tasks. Also, R-CNN architecture forms the basis of Mask R-CNN and it was improved into what we know as Faster R-CNN.

Faster R CNN

Fast R-CNN is an improved version of R-CNN architectures with two stages:

1. Region Proposal Network (RPN). RPN is simply a Neural Network that proposes multiple objects that are available within a particular image.
2. Fast R-CNN. This extracts features using RoI Pool (Region of Interest Pooling) from each candidate box and performs classification and bounding-box regression. RoI Pool is an operation for extracting a small feature map from each RoI in detection.

Faster R-CNN advances this stream by learning the attention mechanism with a Region Proposal Network and Fast R-CNN architecture. The reason why "Fast R-CNN" is faster than

R-CNN is because you don't have to feed 2'000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image, and a feature map is generated from it.

Furthermore, Faster R-CNN is an optimized form of R-CNN because it is built to enhance computation speed (run R-CNN much faster).

The main difference between Fast and Faster RCNN is that Fast R-CNN uses selective search for generating Regions of Interest, while Faster R-CNN uses a "Region Proposal Network" (RPN).

Mask R CNN

Mask R-CNN, or Mask RCNN, is a Convolutional Neural Network (CNN) and state-of-the-art in terms of image segmentation and instance segmentation. Mask R-CNN was developed on top of Faster R-CNN, a Region-Based Convolutional Neural Network.

The first step to understanding how Mask R-CNN works requires an understanding of the concept of Image Segmentation.

The computer vision task Image Segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as image objects). This segmentation is used to locate objects and boundaries (lines, curves, etc.).

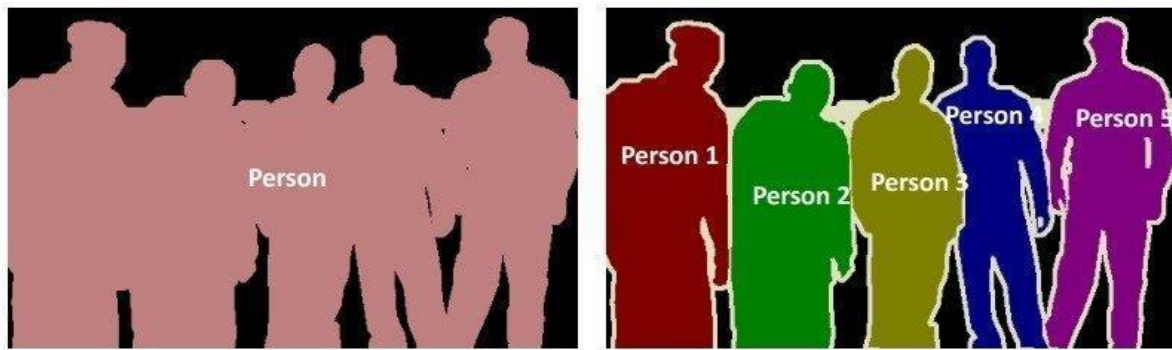
There are 2 main types of image segmentation that fall under Mask R-CNN:

1. Semantic Segmentation
2. Instance Segmentation

Semantic Segmentation

Semantic segmentation classifies each pixel into a fixed set of categories without differentiating object instances. In other words, semantic segmentation deals with the identification/classification of similar objects as a single class from the pixel level.

As shown in the image above, all objects were classified as a single entity (person). Semantic segmentation is otherwise known as background segmentation because it separates the subjects of the image from the background.



Semantic Segmentation

Instance Segmentation

Fig: 5.3.2 Semantic Segmentation and Instance Segmentation

Instance Segmentation

Instance Segmentation, or Instance Recognition, deals with the correct detection of all objects in an image while also precisely segmenting each instance. It is, therefore, the combination of object detection, object localization, and object classification. In other words, this type of segmentation goes further to give a clear distinction between each object classified as similar instances.

As shown in the example image above, for Instance Segmentation, all objects are persons, but this segmentation process separates each person as a single entity. Semantic segmentation is otherwise known as foreground segmentation because it accentuates the subjects of the image instead of the background.

Mask R-CNN was built using Faster R-CNN. While Faster R-CNN has 2 outputs for each candidate object, a class label and a bounding-box offset, Mask R-CNN is the addition of a third branch that outputs the object mask. The additional mask output is distinct from the class and box outputs, requiring the extraction of a much finer spatial layout of an object.

Mask R-CNN is an extension of Faster R-CNN and works by adding a branch for predicting an object mask (Region of Interest) in parallel with the existing branch for bounding box recognition.

The key element of Mask R-CNN is the pixel-to-pixel alignment, which is the main missing piece of Fast/Faster R-CNN. Mask R-CNN adopts the same two-stage procedure with an identical first stage (which is RPN). In the second stage, in parallel to predicting the class and

box offset, Mask R-CNN also outputs a binary mask for each RoI. This is in contrast to most recent systems, where classification depends on mask predictions.

Furthermore, Mask R-CNN is simple to implement and train given the Faster R-CNN framework, which facilitates a wide range of flexible architecture designs. Additionally, the mask branch only adds a small computational overhead, enabling a fast system and rapid experimentation.

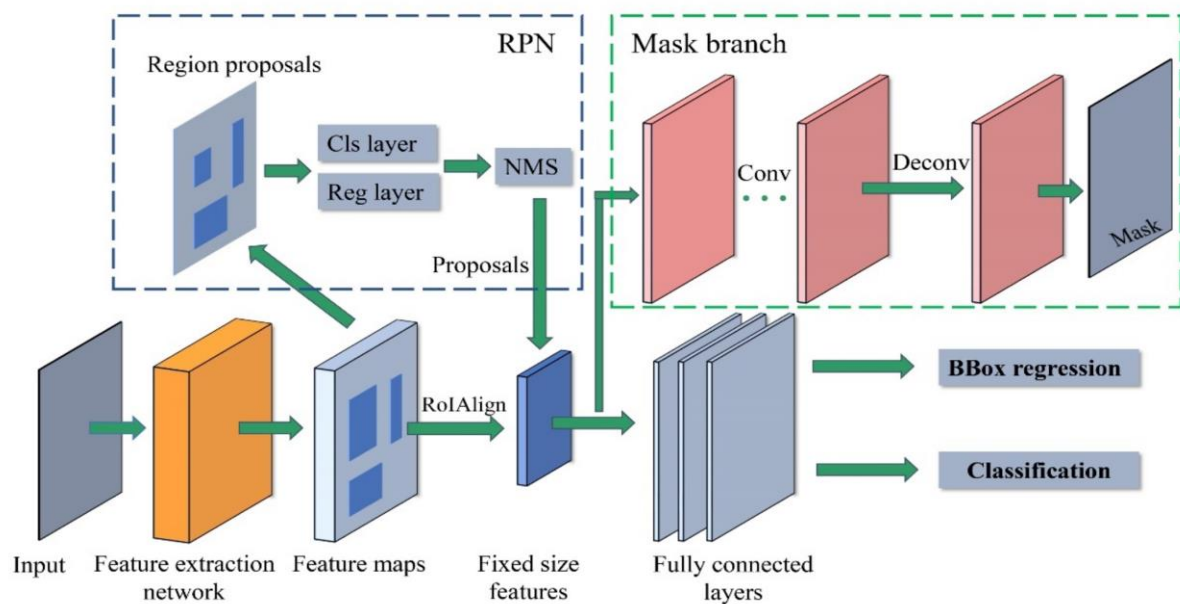


Fig: 5.3.3 Mask RCNN layers

ROI Align:

This concept is useful in stage 2 where the RoIPool extracts features from bounding-boxes. For each RoI as input, there will be a mask and a feature map as output. The mask is obtained using the FCN(Fully Convolutional Network) and the feature map is obtained using the RoIPool. The mask helps with spatial layout, which is crucial to the pixel-to-pixel correspondence. The two things we desire along the procedure are: pixel-to-pixel correspondence; no quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points. Pixel-to-pixel correspondence makes sure that the input and output match in size. If there is a size difference, there will be information loss, and coordinates cannot be matched.

RoIPool is standard for extracting a small feature map from each RoI. However, it performs quantization before subdividing into spatial bins which are further quantized. Quantization produces misalignments when it comes to predicting pixel accurate masks. Therefore, instead of quantization, the coordinates are computed using bilinear interpolation. They use bilinear interpolation to get the exact values of the inputs features at the 4 RoI bins and aggregate the result (using max or average). These results are robust to the sampling location and number of points and to guarantee spatial correspondence. Some implementation details should be mentioned: first, an RoI is considered positive if it has IoU with a ground-truth box of at least 0.5 and negative otherwise. It is important because the mask loss L_{mask} is defined only on positive RoIs. Second, image-centric training is used to rescale images so that pixel correspondence is achieved. An example complete structure is, the proposal number is 1000 for FPN, and then run the box prediction branch on these proposals. The mask branch is then applied to the highest scoring 100 detection boxes. The mask branch can predict K masks per RoI, but only the k th mask will be used, where k is the predicted class by the classification branch. The m -by- m floating-number mask output is then resized to the RoI size and binarized at a threshold of 0.5.

6 IMPLEMENTATION

To implement the code we used google colaboratory as the execution platform.

In this project we implemented 3 ways for image segmentation

- i: Captured picture segmentation implementation
- ii: Real time image capture segmentation implementation
- iii: Real time video capture segmentation implementation

6.1 SAMPLE CODE:

```
%tensorflow_version 1.x
import os
import sys
import random
import math
import numpy as np
import skimage.io
import matplotlib
import matplotlib.pyplot as plt
# imported all the required dependencies

ROOT_DIR = os.path.abspath("")
sys.path.append(ROOT_DIR)

!pip install mrcnn
from mrcnn import utils
import mrcnn.model as modellib
from mrcnn import visualize
from mrcnn.config import Config
sys.path.append(os.path.join(ROOT_DIR, "samples/coco/"))
import demo as coco
%matplotlib inline

# Directory to save logs and trained model
MODEL_DIR = os.path.join(ROOT_DIR, "logs")
```

```

# Local path to trained weights file
COCO_MODEL_PATH = os.path.join(ROOT_DIR, "mask_rcnn_coco.h5")

# Download COCO trained weights from Releases if needed
if not os.path.exists(COCO_MODEL_PATH):
    utils.download_trained_weights(COCO_MODEL_PATH)

# Directory of images to run detection on
IMAGE_DIR = os.path.join(ROOT_DIR, "images")
class InferenceConfig(coco.CocoConfig):
    # Set batch size to 1 since we'll be running inference on
    # one image at a time. Batch size = GPU_COUNT * IMAGES_PER_GPU
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1

config = InferenceConfig()
config.display()
# Create model object in inference mode.
model = modellib.MaskRCNN(mode="inference", model_dir=MODEL_DIR, config=config)

# Load weights trained on MS-COCO
model.load_weights(COCO_MODEL_PATH, by_name=True)
os.mkdir("video")
# COCO Class names
# Index of the class in the list is its ID. For example, to get ID of
# the teddy bear class, use: class_names.index('teddy bear')

class_names = ['BG', 'person', 'bicycle', 'car', 'motorcycle', 'airplane',
               'bus', 'train', 'truck', 'boat', 'traffic light',
               'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird',
               'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear',
               'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie',
               'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball',
               'kite', 'baseball bat', 'baseball glove', 'skateboard',
               'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup',
               'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',
               'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza',

```



```

        'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed',
        'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote',
        'keyboard', 'cell phone', 'microwave', 'oven', 'toaster',
        'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors',
        'teddy bear', 'hair drier', 'toothbrush']

# Load a random image from the images folder
file_names = next(os.walk(IMAGE_DIR))[2]
image = skimage.io.imread(os.path.join(IMAGE_DIR, random.choice(file_names)))

# Run detection
results = model.detect([image], verbose=1)

# Visualize results
r = results[0]
visualize.display_instances(image, r['rois'], r['masks'], r['class_ids'],
                           class_names, r['scores'])

#For pre captured image
import cv2
import time
from PIL import Image
capture = cv2.VideoCapture("video/test-video-zoo.mp4")
width = int(capture.get(cv2.CAP_PROP_FRAME_WIDTH) + 0.5)
height = int(capture.get(cv2.CAP_PROP_FRAME_HEIGHT) + 0.5)
size = (height, width)
fourcc = cv2.VideoWriter_fourcc(*'DIVX')
# output_movie = cv2.VideoWriter('Zoo-video-masked.avi', fourcc, 30.0, (1920,1080))
video = cv2.VideoWriter("Zoo_video_masked.avi", cv2.VideoWriter_fourcc(*'DIVX'), fps,
(1152, 1152))
# Constructing code of the codec to be used in the function VideoWriter
# writer = cv2.VideoWriter('Zoo-video-masked.mp4', fourcc, 20, (1920, 1080))
count=0
while True:
    # Capture frame-by-frame
    ret, frame = capture.read()
    if not ret:
        break

```

```

results = model.detect([frame], verbose=1)
r = results[0]

boxes = r['rois']
masks = r['masks']
class_ids = r['class_ids']
scores = r['scores']

# Run detection
start = time.time()
masked_image = vc.display_instances(frame, boxes, masks, class_ids, class_names, scores)
end = time.time()
print("Inference time: {:.2f}s".format(end - start))
# Display the resulting frame
# cv2.imshow("", masked_image)
#save image
# writer = cv2.VideoWriter('Zoo-video-masked.mp4', fourcc, 30, (frame.shape[1],
frame.shape[0]), True)
#we are saving the image as "Masked-image.jpg under visualizeCustom code"
#this image file we are writing in our videowriter object
print("here,", type(masked_image))
img=cv2.imread("Masked-image.jpg")
print(img.shape)
video.write(img)
# output_movie.write(masked_image)
count+=1
print("count:", count)
# if cv2.waitKey(1) & 0xFF == ord('q'):
#     break

# When everything done, release the capture
print("WARNING")
cv2.destroyAllWindows()
capture.release()
video.release()
# cv2.destroyAllWindows()

```

```

# Load a random image from the images folder
file_names = next(os.walk(IMAGE_DIR))[2]
image = skimage.io.imread(os.path.join(IMAGE_DIR, random.choice(file_names)))

# Run detection
results = model.detect([image], verbose=1)

# Visualize results
r = results[0]
visualize.display_instances(image, r['rois'], r['masks'], r['class_ids'],
                           class_names, r['scores'])
print(type(image))
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode

#to capture Real time image

def take_photo(filename='photo.jpg', quality=0.8):
    js = Javascript("""
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);
      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({ video: true });
      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.

```

```

    await new Promise((resolve) => capture.onclick = resolve);

    const canvas = document.createElement('canvas');
    canvas.width = video.videoWidth;
    canvas.height = video.videoHeight;
    canvas.getContext('2d').drawImage(video, 0, 0);
    stream.getVideoTracks()[0].stop();
    div.remove();
    return canvas.toDataURL('image/jpeg', quality);
}

display(js)
data = eval_js('takePhoto({}).format(quality)')
binary = b64decode(data.split(',')[1])
with open(filename, 'wb') as f:
    f.write(binary)
return filename
from IPython.display import Image
try:
    filename = take_photo()
    print('Saved to {}'.format(filename))

    # Show the image which was just taken.
    #display(Image(filename))
    #image=Image(filename)
    image = skimage.io.imread(filename)
    results = model.detect([image], verbose=1)
    # Visualize results
    r = results[0]
    visualize.display_instances(image, r['rois'], r['masks'], r['class_ids'],
                               class_names, r['scores'])
    #print(type(image))
except Exception as err:
    # Errors will be thrown if the user does not have a webcam or if they do not
    # grant the page permission to access it.
    print(str(err))

```

6.2 OUTPUT SCREENS



Fig:6.2.1 Output of the given image



Fig: 6.2.2 Output of captured image

6.3 TEST CASES

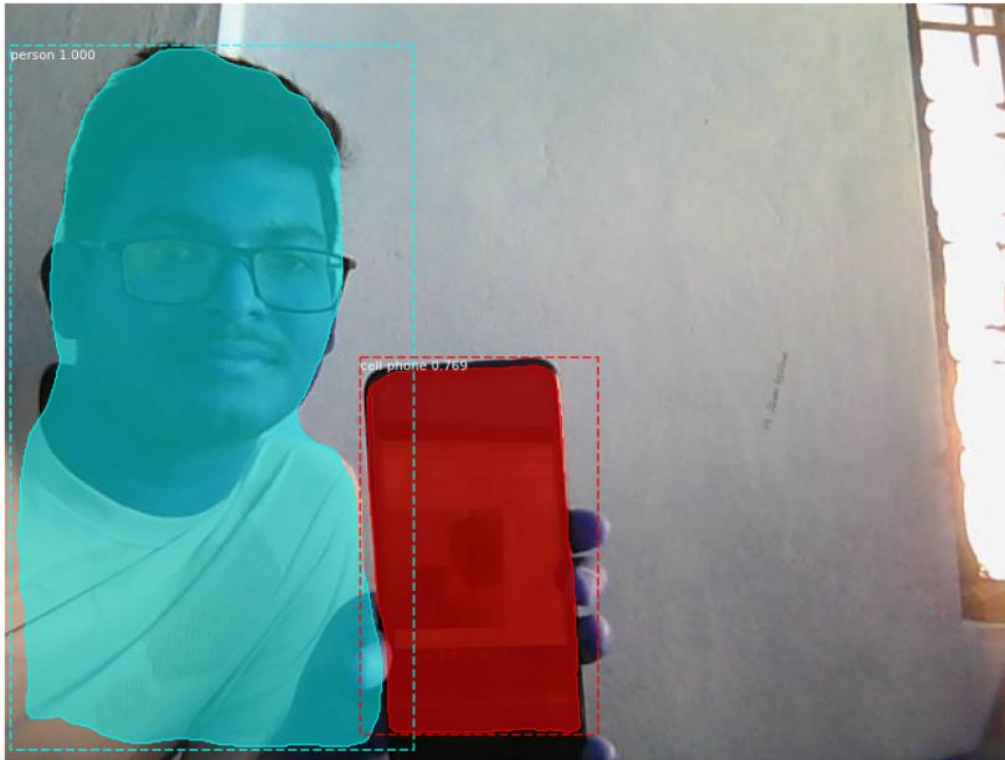


Fig: 6.3.1 Using Real Time Image capturing

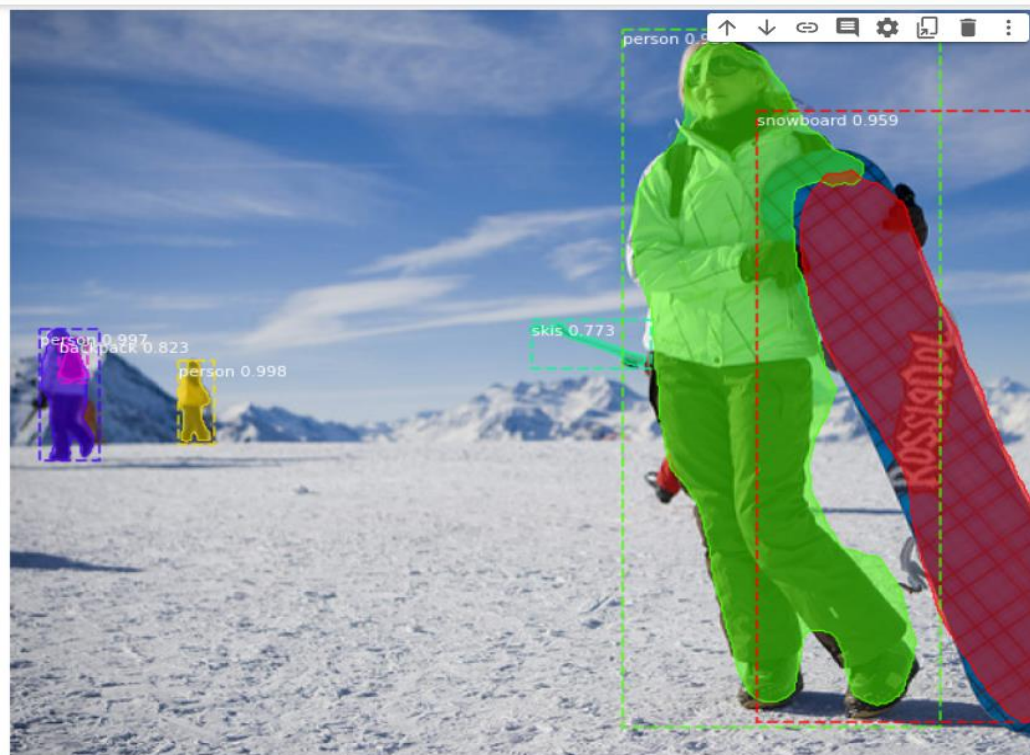


Fig: 6.3.2 Using pre-captured images

7 CONCLUSION

Mask RCNN is a deep neural network aimed to solve the instance segmentation problems in machine learning or computer vision. Mask R-CNN is a conceptually simple, flexible, and general framework for object instance segmentation. It can efficiently detect objects in an image while simultaneously generating a high-quality segmentation mask for each instance. It does object detection and instance segmentation, and can also be extended to human pose estimation. It extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps.

In this project we used both real time image and pre captured images to implement image segmentation using the Mask RCNN. The desired output is obtained.

8 FUTURE SCOPE

In forthcoming future, this project can be modified and implemented for auto pilot cars, cancer cell shape detection and object detection from satellite images. As taking basis of this project, all the above projects can be implemented. Due to advances in image processing and related technologies there will be millions and millions of robots in the world in a few decades time, transforming the way the world is managed. Advances in image processing and artificial intelligence will involve spoken commands, anticipating the information requirements of governments, translating languages, recognizing and tracking people and things, diagnosing medical conditions, performing surgery, reprogramming defects in human DNA, and automatic driving all forms of transport. With increasing power and sophistication of modern computing, the concept of computation can go beyond the present limits and in future, image processing technology will advance and the visual system of man can be replicated.

9 BIBLIOGRAPHY

- [1] J. Dai, K. He, and J. Sun, “Instance-aware semantic segmentation via multi-task network cascades,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3150–3158.
- [2] Z. Hayder, X. He, and M. Salzmann, “Boundary-aware instance segmentation,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), no. EPFLCONF-227439, 2017.
- [3] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik, “Simultaneous detection and segmentation,” in European Conference on Computer Vision. Springer, 2014, pp. 297–312.
- [4] J. Dai, K. He, and J. Sun, “Convolutional feature masking for joint object and stuff segmentation,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3992–4000.
- [5] S. Zagoruyko, A. Lerer, T.-Y. Lin, P. O. Pinheiro, S. Gross, S. Chintala, and P. Dollar, “A multipath network for object detection,” in British Machine Vision Conference, 2016.
- [6] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, “Fully convolutional instance-aware semantic segmentation,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
- [7] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” arXiv preprint arXiv:1703.06870, 2017.
- [8] P. O. Pinheiro, R. Collobert, and P. Dollar, “Learning to segment object candidates,” in Advances in Neural Information Processing Systems, 2015, pp. 1990–1998.
- [9] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollar, “Learning to refine object segments,” in European Conference on Computer Vision. Springer, 2016, pp. 75–91.

[10] H. Hu, S. Lan, Y. Jiang, Z. Cao, and F. Sha, “FastMask: Segment object multi-scale candidates in one shot,” arXiv preprint arXiv:1612.08843, 2016, accepted for CVPR 2017.

[11] Jia Ding, Aoxue Li, Zhiqiang Hu, and Liwei Wang. Accurate pulmonary nodule detection in computed tomography images using deep convolutional neural networks. CoRR, abs/1706.04303, 2017.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross B. Girshick. Mask R-CNN. CoRR, abs/1703.06870, 2017.