WASTE DETECTION AND CLASSIFICATION USING DETECTRON2

A MINI PROJECT REPORT SUBMITTED TO THE BHARATHIAR UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

MASTER OF SCIENCE IN DATA ANALYTICS

Submitted by

SENTHIL KUMAR N

(Reg. no: 21CSEG31)

Under the guidance of

Mr. S. PALANISAMY, MCA., M.Phil.,

Assistant Professor

Dr. K. ARUMUGAM, MCA., MBA., MPhil., PhD.,

Guest Faculty

DEPARTMENT OF COMPUTER APPLICATIONS

BHARATHIAR UNIVERSITY

COIMBATORE-641046.

DECEMBER - 2022

CERTIFICATE

This is to certify that, this mini-project work entitled "**WASTE DETECTION AND CLASSIFICATION USING DETECTRON2**" was submitted to the Department of Computer Applications, Bharathiar University in partial fulfillment of the requirements for the award of the degree of **Master of Science in Data Analytics**, is a record of original work done by **SENTHIL KUMAR N (21CSEG31),** during his period of study in the Department of Computer Applications, Bharathiar University, Coimbatore, under my supervision and guidance, and this project work has not formed the basis for the award of any Degree/ Diploma /Associateship/ Fellowship or similar title to any candidate of any University.

Place:

Date:

Project Guide **Head of the Department**

Submitted for the University Viva-Voce Examination held on ___________

Internal Examiner **External Examiner**

DECLARATION

I hereby declare that this mini project work titled, "**WASTE DETECTION AND CLASSIFICATION USING DETECTRON2**" submitted to Department of Computer Applications, Bharathiar University, is a record of original work done by **SENTHIL KUMAR N (21CSEG31)**, under the supervision and guidance of **Mr. S. PALANISAMY, MCA., M.Phil.,** Assistant Professor and **Dr. K. ARUMUGAM.**, **MCA., MBA., MPhil., PhD.,** Guest Faculty Department of Computer Applications, Bharathiar University, and that this project work has not formed the basis for the award of any Degree/ Diploma/ Associateship/ Fellowship or similar title to any candidate of any University.

Place: Signature of the candidate

Date: (SENTHIL KUMAR N)

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I express my sincere gratitude to our Professor & Head of the Department **Dr. T. DEVI, M.C.A., M.Phil., Ph.D. (UK),** Department of Computer Application, Bharathiar University, Coimbatore, thanks to department faculty members and guest faculties supported to acquire knowledge from various inputs. I am forever indebted to this mini project guides **Mr. S. PALANISAMY, MCA., M.Phil.,** and **Dr. K. ARUMUGAM, MCA., MBA., MPhil., PhD,** Department of Computer Applications, Bharathiar University for providing valuable suggestions and always been enthusiastically guiding us throughout the project.

I express my thanks to technical staff members lab administrator **Mr. A. SIVARAJ** and **Mr. P. ELANCHEZLIAN** and office staff members **Mrs. B. PRIYADHARSHINI** and **Mr. A. KALIDASS**.

Finally, I also extend my special thanks to my family, friends, who have kindly provided the necessary support for the successful completion of the project and their moral support.

ABSTRACT

The process of segregating waste prompts the generation of energy out of waste, diminishing landfills, recycling, and reduction of waste. Erroneous disposal of waste leads to recycling contamination. Contamination is a tremendous issue to the recycling industry that can be alleviated with automatic computerized waste sorting. The presence of models or strategies which help people to sort trash has become extremely important in the right discard of that garbage. Even though there are various sorts of recycling categories, many people remain confused or cannot appropriately recognize how to decide the right trash bin to dispose of every trash. Waste management and systematic sorting of them are considered to be a significant role in ecological development around the world. Society needs to reduce the amount of waste by recycling and reusing discarded materials that result in reducing environmental problems. This project aims to create an automated waste detection system using a library called Detectron2 by Facebook that will gather the waste images taken from a camera with instance segmentation, detection & prediction, and categorize the waste materials into 60 categories so that the waste can be properly dumped in the recyclable and non-recyclable bin.

1. INTRODUCTION

Modern waste management is well known to all, but unfortunately neglected by many who practice waste segregation to address the problems caused by improper waste disposal. Illegal dumping is an ongoing problem in many urban communities around the world. Odours and pollutants caused by abandoned household items, dumped garbage and construction debris are ruining cities and threatening the well-being of citizens. To curb illegal dumping, some urban areas are planning network-based voluntary reporting systems and security camera-based monitoring systems. However, these methods require manual observation and detection, are vulnerable to false alarms, and are costly. Garbage is a global problem that affects all living things. Research shows that 74% of the plastic that enters the ocean from the Philippines comes from trash. In our daily life, we may neglect to sort household garbage accurately, and industrially, the organization responsible for this part must expend a great deal of effort and effort. Separation of garbage is to divide garbage into several parts. This is done regularly by physical hand-picking, which can be dangerous and a health hazard if not done properly. Waste classification and identification is known to solve this problem. It helps everyone, especially government agencies and officials, to effectively separate waste, especially recyclable waste.

To automate the recycling process, it is important to propose a smart framework that enables effective waste sorting. Due to the large number of objects detected in a limited amount of time, the use of object detection software in waste sorting is a worthwhile practice as opposed to traditional recycling strategies. Traditional approaches rely on human labour and tend to fail to separate waste for recycling. The Deep Learning techniques have been effectively applied in a variety of areas, including, medical imaging, autonomous driving, and many industrial environments. It gives amazing results in object identification problems. Applying these technologies to waste sorting can increase the amount of recycled materials, make everyday life more convenient for ordinary people and make industries more efficient.

Deep Learning is the class of Machine Learning Algorithms which is a subset of Artificial Intelligence that uses multiple layers of data representation and feature extraction. The different applications where deep learning plays an important role are in speech and visual recognition, speech to text conversion, detection, and recognition of the face, image recognition, drug detection, weather prediction, etc. Deep learning permits the preparation of numerous layers through the computational models to learn the representations of data with the abstraction of numerous layers. Convolutional Neural Network (CNN) is the most suitable image classification technique in the most recent years, wherefrom the segmented objects no handcrafted features are extracted. A previously trained CNN will always perform better on fresh images for classification as it has already adapted or learned the visual features and can transfer that information through transfer learning.

Thus, this project is aimed at planning and developing up a framework with a deep learning approach that can be effectively used for waste segregation. The image will be recognized by utilizing the concept of a convolutional neural network and with the help of the state-of-the-art object detection algorithms that identifies wastes from their shape, colour, dimension, and size. This technique automatically will help the system to learn the pertinent features from the sample images of the trash and consequently recognize those features in new images. By using the strategy of convolutional neural networks, garbage will be classified into different classes. The strategy utilized for this characterization is with the assistance of PyTorch and Mask R-CNN technique. Through this technique, bounding boxes segmentation masks are made on the recyclable waste demonstrating which of the 60 different classes, the waste falls into. The main objective of this study is to develop software to detect types of recyclable materials in trash bins and check for possible contamination (non-recyclable materials), which would ultimately reduce human effort in waste segregation and expedite the entire process.

2. RELATED WORK

Throughout the previous years, various works have been executed with the point of limiting the effect of the incorrect disposal of waste. Many neural networks and other deep neural network architecture like EfficientNet-B2, ResNet, DenseNet, EfficientNetv2 and other classification algorithms such as Support Vector Machine based image classification projects are being done previously. The majority of the proposed approaches are based on deep learning algorithms utilized in the computer vision field.

A comparison study was performed by Ying Liu et al. 2018, to detect and recognize garbage through an improved YOLOv2 model with narrow-band Internet of Things and compared with Fast R-CNN algorithm. The result shows that the both YOLOv2 and Fast R-CNN algorithms predicts relatively equal. The accuracy level was 89%.

And other study performed by Sylwia et al. 2021, experimented different bench mark datasets and annotated the target labels with 7 categories of wastes. The authors built two stage detectors for litter detection and classification. EfficientDet-D2 is used to localize the litter, and EfficientNet-B2 to classify the detected the waste into 7 categories. The proposed approach achieves up to 70% of average precision in waste detection and around 75% of classification accuracy on the test dataset.

Basically, in the most recent years, computer vision has been considered as an apparatus to help waste classification, and deep learning strategies have reached sensible outcomes in controlled situations.

3. BACKGROUND

3.1. NEURAL NETWORKS

The core component of Machine Learning (including Deep Learning) is artificial neurons. The learning computations are performed with the help of neurons. In artificial intelligence models, many neurons work together to perform complex numerical/mathematical calculations. The formation of networks formed by these neurons is known as artificial neural networks. The idea of artificial neurons comes from neurons that exist biologically in the human sensory or nervous system. Like neural networks in the human body, artificial neural networks are divided into layers. Dendrites are nothing more than information terminals of neurons in artificial neurons. Input from an axon is processed through the synapses and dendrites of another neuron, and its output is transferred to other neurons. An input signal traveling around a line of input is multiplied by the weight of the line in the computational model. A weighted input signal is processed by a mathematical function. This special function is called the activation function '*f* '. Already processed signals are passed to the next layer of neurons for further processing. In this model, the learning factors are known to be the weights of associations between neurons. The values of this model are changed during training with the ultimate goal of merging the error to zero. In the human body, signals carried by dendrites are summed at the soma, and when the sum exceeds a threshold, signals from axons are initiated at that point. A similar methodology is used for mathematical or numerical models. The threshold is determined by the activation function *f*. The sigmoid function is known to be the default determination of the activation function. The sigmoid function takes the sum value as input and transforms it somewhere between 0 and 1.

3.2. CONVOLUTIONAL NEURAL NETWORKS (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:

a) **Convolutional Layer**: This layer helps to abstract the input image as a feature map via the use of filters and kernels.
b) **Pooling Layer:** This layer helps to downsample feature maps by summarizing the presence of features in patches of the feature map.
c) **Fully connected Layer**: Fully connected layers connect every neuron in one layer to every neuron in another layer.

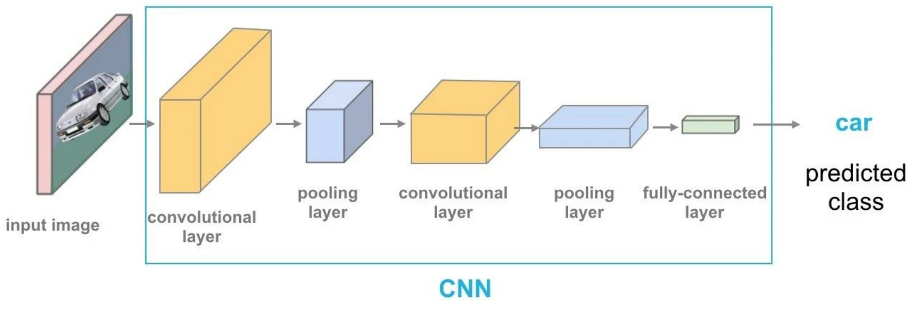


Fig. 1 Basic CNN Architecture

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.

The main function of Convolutional Neural Networks is to essentially group or classify the pictures, cluster the images by similarity, and perform object detection with the assistance of artificial neural networks. The convolutional neural network takes the information of the image and processes the picture as a tensor, which is the matrices of numbers with extra dimensions and carries out a sort of search. The 3D objects are a portion of the instances that recognize the images as volumes. It is being uploaded in numerous applications, including recognition of the face and object identification. It's one of the best non-trivial assignments. In constituent to neural networks, the convolutional layer, subsampling layers, and fully connected layer are the three distinctive layer types that are viewed as a part of CNN. CNN is mainly used for image recognition as this method has got more advantages in comparison with other strategies.

3.3. FASTER R-CNN

Faster R-CNN, is composed of two modules. The first module is a deep fully convolutional network that proposes regions, and the second module is the Fast R-CNN detector that uses the proposed regions. The entire system is a single, unified network for object detection (Fig. 2). Using the recently popular terminology of neural networks with 'attention' mechanisms, the RPN module tells the Fast R-CNN module where to look.

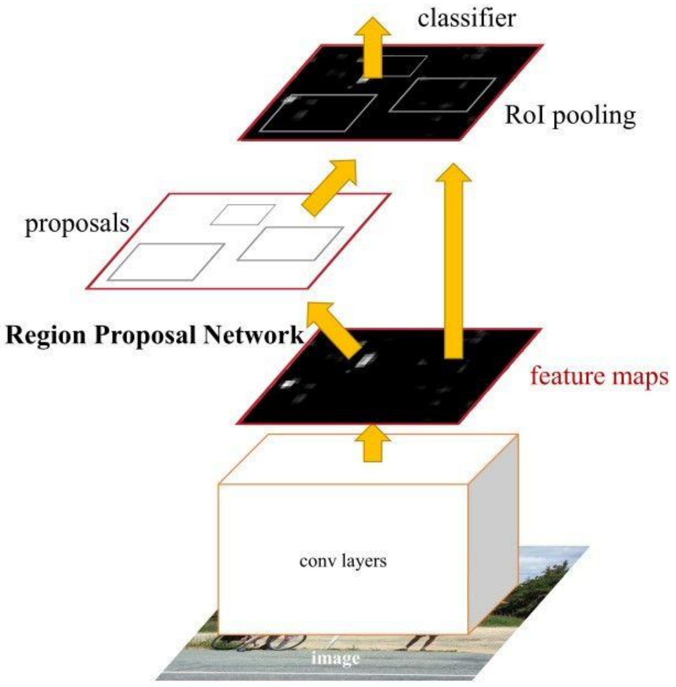


Fig. 2 Faster R-CNN, a single network

3.3.1 REGION PROPOSAL NETWORKS

A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score.

3.3.2 FEATURE MAPS

The basic idea of neural networks is that neurons learn features from the input. In CNNs, the feature map is the output of one filter applied to the previous layer. It is called a feature map because it is a mapping of where a certain kind of feature is found in the image. Convolutional Neural Networks look for "features" such as straight lines, edges, or even objects. Whenever they spot these features they report them to the feature map. Each feature map is looking for something else. One feature map could be looking for straight lines, the other for curves.

3.4. 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 work 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:

- Semantic Segmentation
- Instance Segmentation

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 nmask. The additional mask output is distinct from the class and box outputs, requiring the extraction of a much finer spatial layout of an object.

It 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.

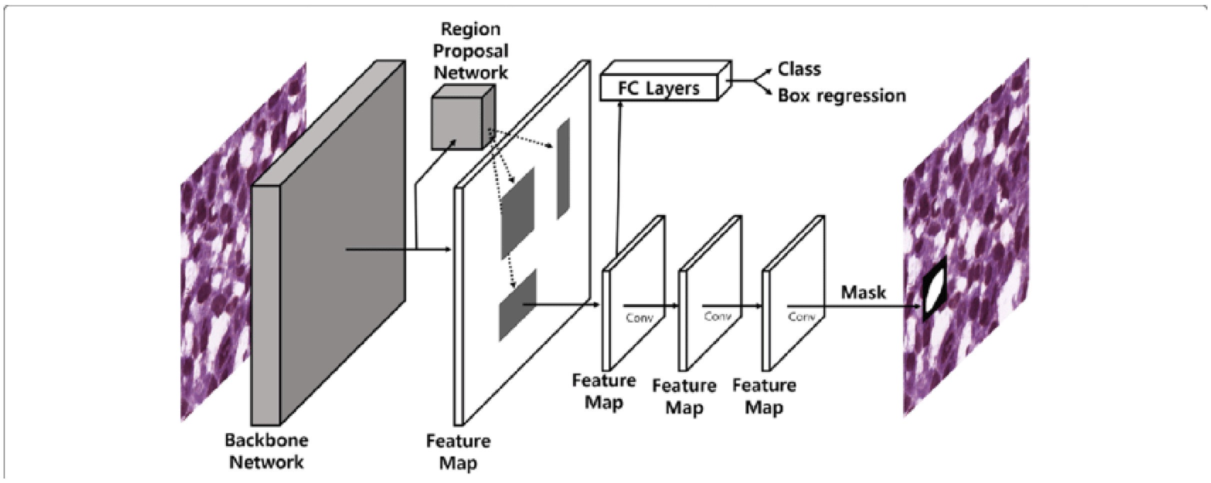


Fig. 3 The overall network architecture of Mask R-CNN

The Loss function for each sampled Region of Interest (RoI) in Mask R-CNN

$$L = L_{class} + L_{box} + L_{mask}$$

3.4.1 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.

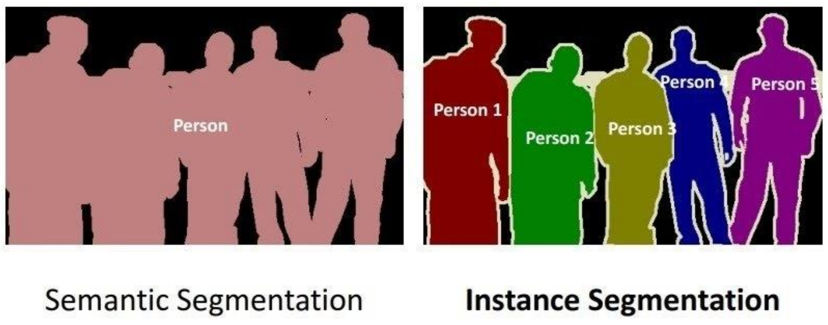


Fig. 4 Difference between Semantic Segmentation and Instance Segmentation

3.4.2 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 Fig.4, 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.

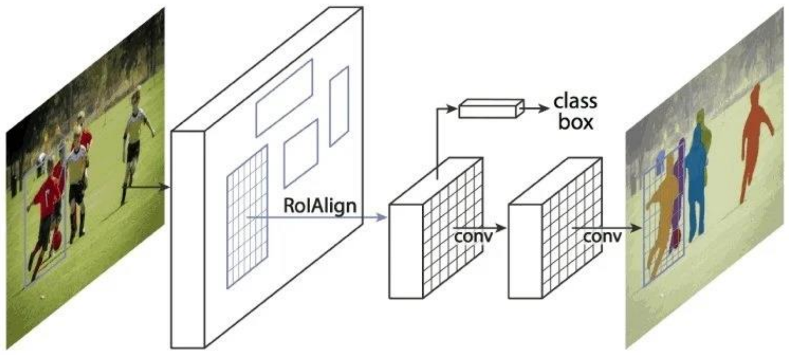


Fig. 5 The Mask R-CNN framework for Instance Segmentation

3.5 OBJECT DETECTION

Object Detection is a part of the computer vision technique where a software framework can detect, trace, and locate the object from any given video or image. One of the special traits of object detection is that it recognizes the object's class (for instance, person, table, chair, and so forth) and the coordinates of the location of the object in the given picture. By drawing a bounding box around the object, it states that the region is being pointed out. The bounding box might or might not precisely find the location of the object. The capability of locating the object inside a picture describes the accuracy of the performance of the algorithm which is being used for detection. Face identification is one of the instances of object detection.

For the most part, the object detection task is completed in three stages:

- The little rectangular portions are being produced upon the input image.
- Feature extraction is completed for each portion of the rectangular region to predict whether the rectangular shape contains a valid object.
- Finally, the rectangular boxes which are overlapped get combined into a single bounding box showing the detection result of the object.

3.6 DETECTRON2

Detectron2 is Facebook AI Research's (FAIR) next generation library that provides state-of-the-art detection and segmentation algorithms. It is the successor of **Detectron** and **maskrcnn-benchmark**. It supports a number of computer vision research projects and production applications in Facebook. Since its release in 2018, the Detectron object detection platform has become one of FAIR's most widely adopted open source projects.

It is a ground-up rewrite of Detectron that started with maskrcnn-benchmark. The platform is now implemented in **PyTorch**. With a new, more modular design, Detectron2 is flexible and extensible, and able to provide fast training on single or multiple GPU servers. Detectron2 includes high-quality implementations of state-of-the-art object detection algorithms, including *DensePose*, *panoptic feature pyramid networks*, and numerous variants of the pioneering *Mask R-CNN* model family also developed by FAIR. Its extensible design makes it easy to implement cutting-edge research projects without having to fork the entire codebase.

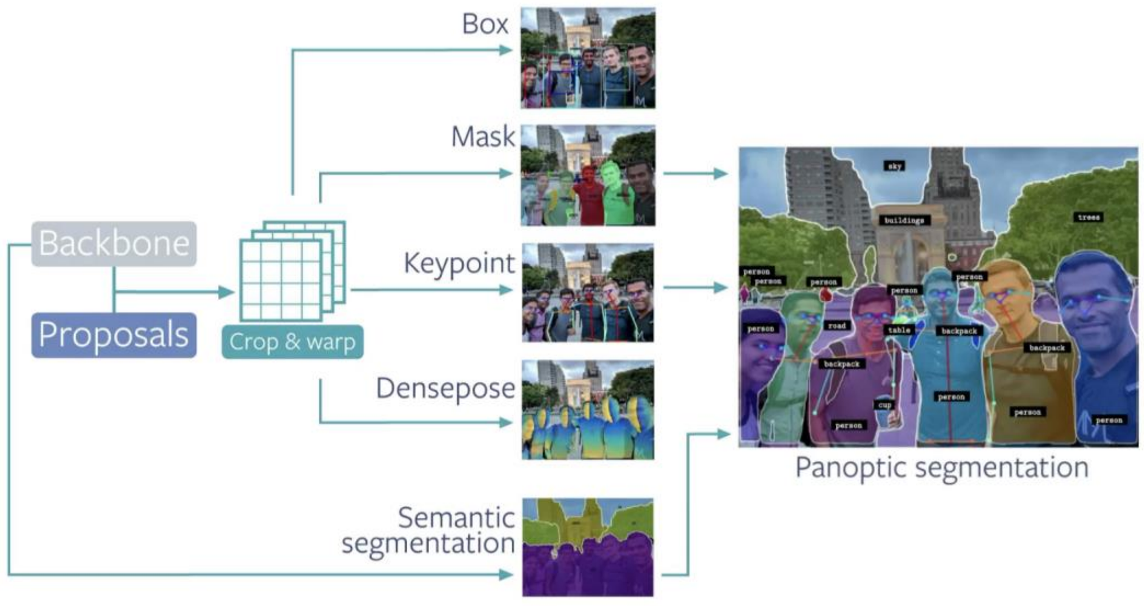


Fig. 6 Detectron2's modular design allows users to take an image and easily switch to custom backbones, insert different prediction heads, and perform panoptic segmentation.

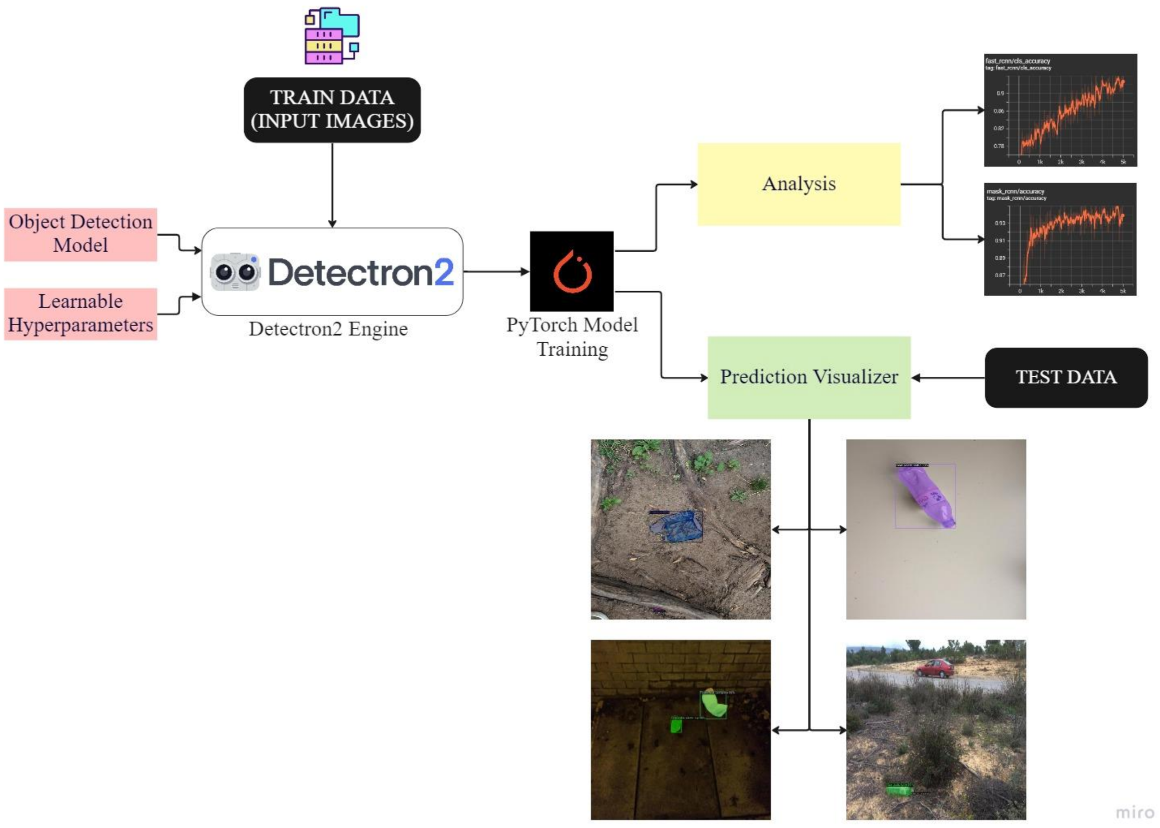


Fig. 7 Mechanism of Object Detection in Detectron2

4. SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENT

PROCESSOR	:	11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00 GHz
RAM	:	8.00 GB
HARD DISK	:	236 GB
SYSTEM TYPE	:	64-bit operating system, x64-based processor
GPU 0 (LOCAL)	:	Intel(R) UHD Graphics
GPU MEMORY (LOCAL)	:	3.9 GB
GPU 0 (COLAB)	:	Tesla T4
GPU MEMORY (COLAB)	:	16 GB GDDR6

4.2 SOFTWARE REQUIREMENT

OPERATING SYSTEM	:	Windows 10 Home Single Language
IDE	:	Google Colaboratory (Colab)
LANGUAGE	:	Python 3.8.16
CUDA COMPILER	:	Nvidia(R) CUDA Compiler driver
CUDA VERSION (COLAB)	:	cu116
PACKAGES VERSION	:	

- detectron2 (0.6)
- torch (1.13.0+cu116)
- torchvision (0.14.0+cu116)
- cuda – cu116
- opencv-python (4.6.0.66)
- roboflow (0.2.21)
- pip (21.1.3)
- pycocotools (2.0.6)
- Pillow (7.1.2)
- Matplotlib (3.2.2)
- TensorBoard (2.9.1)
- pandas (1.3.5)
- numpy (1.21.6)
- glob (0.7)
- json (4.3.3)
- tqdm (4.64.1)

5. SOFTWARE DESCRIPTION

5.1 SOFTWARE USED

The following software are used in this project

- PYTHON
- DRIVE
- ROBOFLOW
- GOOGLE COLABORATORY
- CUDA
- TENSORBOARD

5.1.1 PYTHON

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

HISTORY OF PYTHON

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's "benevolent dictator for life", a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a five-member Steering Council to lead the project.

Python 2.0 was released on 16 October 2000, with many major new features. Python 3.0, released on 3 December 2008, with many of its major features backported to Python 2.6.x and 2.7.x. Releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3.

PYTHON VERSIONS

- Python 3.11.0, documentation released on 24 October 2022.
- Python 3.10.7, documentation released on 6 September 2022.
- Python 3.10.6, documentation released on 8 August 2022.
- Python 3.10.5, documentation released on 6 June 2022.
- Python 3.10.4, documentation released on 24 March 2022.

PYTHON FEATURES

1. Free and Open Source
2. Easy to code
3. Easy to read
4. Object-oriented language
5. GUI Programming support
6. High-level language
7. Extensible feature
8. Easy to debug
9. Portable
10. Integrated
11. Large Standard library
12. Dynamic memory allocation

5.1.2 DRIVE

Google Drive is a file storage and synchronization service developed by Google. Launched on April 24, 2012, it allows users to store files in the cloud (on Google's servers), synchronize files across devices, and share files. In addition to a web interface, Google Drive offers apps with offline capabilities for Windows and MacOS computers, and Android and iOS smartphones and tablets.

It offers users 15 GB of free storage through Google One. Google One also offers 100 GB, 200 GB, 2 TB, offered through optional paid plans. Files uploaded can be up to 750 GB in size. Users can change privacy settings for individual files and folders, including enabling sharing with other users or making content public. On the website, users can search for an image by describing its visuals, and use natural language to find specific files, such as "find my budget spreadsheet from last December". The website and Android app offer a Backups section to see what Android devices have data backed up to the service, and a completely overhauled computer app released in July 2017 allows for backing up specific folders on the user's computer. A Quick Access feature can intelligently predict the files users need.

Google Drive is a key component of Google Workspace, Google's monthly subscription offering for businesses and organizations that operated as G Suite until October 2020. As part of select Google Workspace plans, Drive offers unlimited storage, advanced file audit reporting, enhanced administration controls, and greater collaboration tools for teams.

5.1.3 ROBOFLOW

Roboflow makes managing, pre-processing, augmenting, and versioning datasets for computer vision seamless. The official Roboflow python package that interfaces with the Roboflow API to download the datasets available in their platform. Key features of Roboflow:

- Import and Export image datasets into any supported format
- Preprocess and augment data using Roboflow's dataset management tools
- Train computer vision models using Roboflow Train and deploy to production
- Use community curated projects to start building your own vision-powered products

5.1.4 GOOGLE COLABORATORY

Colab is a Jupyter Notebook-like product from Google Research. A Python program developer can use this notebook to write and execute random Python program codes just using a web browser.

In a nutshell, Colab is a cloud-hosted version of Jupyter Notebook. To use Colab, you do not need to install and runtime or upgrade your computer hardware to meet Python's CPU/GPU intensive workload requirements. Furthermore, Colab gives you free access to computing infrastructure like storage, memory, processing capacity, graphics processing units (GPUs), and tensor processing units (TPUs). Google has specially programmed this cloud-based Python coding tool keeping in mind the needs of machine learning programmers, big data analysts, data scientists, AI researchers, and Python learners.

The best part is one code notebook for all the components needed to present a complete machine learning or data science project to program supervisors or sponsors. For example, your Colab notebook can contain executable codes, live Python codes, rich text, HTML, LaTeX, images, data visualizations, charts, graphs, tables, and more.

GPUs and TPUs :

Free Colab users get chargeless access to GPU and TPU runtimes for up to 12 hours. Its GPU runtime comes with Intel Xeon CPU @2.20 GHz, 13 GB RAM, Tesla K80 accelerator, and 12 GB GDDR5 VRAM.

The TPU runtime consists of an Intel Xeon CPU @2.30 GHz, 13 GB RAM, and a cloud TPU with 180 teraflops of computational power. With Colab Pro or Pro+, you can commission more CPUs, TPUs, and GPUs for more than 12 hours.

Pre-Installed Libraries :

It offers multiple pre-installed libraries so that you can import the required library from Code snippets. Such libraries include NumPy, Pandas, Matplotlib, PyTorch, TensorFlow, Keras, and more ML libraries.

Limitations :

Colab resources are not guaranteed and not unlimited, and the usage limits sometimes fluctuate. This is necessary for Colab to be able to provide resources free of charge. Users who are interested in more reliable access to better resources may be interested in Colab Pro.

Resources in Colab are prioritized for interactive use cases. We prohibit actions associated with bulk compute, actions that negatively impact others, as well as actions associated with bypassing our policies. The following are disallowed from Colab runtimes:

- file hosting, media serving, or other web service offerings not related to interactive compute with Colab
- downloading torrents or engaging in peer-to-peer file-sharing
- using a remote desktop or SSH
- connecting to remote proxies
- mining cryptocurrency
- running denial-of-service attacks
- password cracking
- using multiple accounts to work around access or resource usage restrictions
- creating deepfakes

Drive mount :

Mounting Google Drive on Colab allows any code in your notebook to access any files in our Google Drive. We usually require that users manually grant this access every time they connect to a new runtime by adding a code cell to the notebook. This ensures that the user fully understands the permissions being granted to the notebook.

In some cases, we only require Google Drive authorization once, and automatically re-mount Google Drive during future sessions. To protect your files, we only allow this when a notebook passes multiple checks. For example, any notebooks which have been edited by another user do not automatically mount Google Drive.

5.1.5 CUDA (COMPUTE UNIFIED DEVICE ARCHITECTURE)

It is a parallel computing platform and application programming interface (API) that allows software to use certain types of graphics processing units (GPUs) for general purpose processing, an approach called general-purpose computing on GPUs (GPGPU). CUDA is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels.

CUDA is designed to work with programming languages such as C, C++, and Fortran. This accessibility makes it easier for specialists in parallel programming to use GPU resources, in contrast to prior APIs like Direct3D and OpenGL, which required advanced skills in graphics programming. CUDA-powered GPUs also support programming frameworks such as OpenMP, OpenACC and OpenCL; and HIP by compiling such code to CUDA.

It was created by **Nvidia**. When it was first introduced, the name was an acronym for Compute Unified Device Architecture, but Nvidia later dropped the common use of the acronym.

Background

The graphics processing unit (GPU), as a specialized computer processor, addresses the demands of real-time high-resolution 3D graphics compute-intensive tasks. By 2012, GPUs had evolved into highly parallel multi-core systems allowing efficient manipulation of large blocks of data. This design is more effective than general-purpose central processing unit (CPUs) for algorithms in situations where processing large blocks of data is done in parallel, such as:

- cryptographic hash functions
- machine learning
- molecular dynamics simulations
- physics engines
- sort algorithms

Programming Ability

The CUDA platform is accessible to software developers through CUDA-accelerated libraries, compiler directives such as OpenACC, and extensions to industry-standard programming languages including C, C++ and Fortran. C/C++ programmers can use 'CUDA C/C++', compiled to PTX with nvcc, Nvidia's LLVM-based C/C++ compiler, or by clang itself.[6] Fortran programmers can use 'CUDA Fortran', compiled with the PGI CUDA Fortran compiler from The Portland Group.

In addition to libraries, compiler directives, CUDA C/C++ and CUDA Fortran, the CUDA platform supports other computational interfaces, including the Khronos Group's OpenCL, Microsoft's DirectCompute, OpenGL Compute Shader and C++ AMP. Third party wrappers are also available for Python, Perl, Fortran, Java, Ruby, Lua, Common Lisp, Haskell, R, MATLAB, IDL, Julia, and native support in Mathematica.

In the computer game industry, GPUs are used for graphics rendering, and for game physics calculations (physical effects such as debris, smoke, fire, fluids); examples include PhysX and Bullet. CUDA has also been used to accelerate non-graphical applications in computational biology, cryptography and other fields by an order of magnitude or more.

CUDA provides both a low level API (CUDA Driver API, non single-source) and a higher level API (CUDA Runtime API, single-source). The initial CUDA SDK was made public on 15 February 2007, for Microsoft Windows and Linux. Mac OS X support was later added in version 2.0, which supersedes the beta released February 14, 2008. CUDA works with all Nvidia GPUs from the G8x series onwards, including GeForce, Quadro and the Tesla line. CUDA is compatible with most standard operating systems.

CUDA 8.0 comes with the following libraries (for compilation & runtime, in alphabetical order):

- cuBLAS – CUDA Basic Linear Algebra Subroutines library
- CUDART – CUDA Runtime library
- cuFFT – CUDA Fast Fourier Transform library
- cuRAND – CUDA Random Number Generation library
- cuSOLVER – CUDA based collection of dense and sparse direct solvers
- cuSPARSE – CUDA Sparse Matrix library
- NPP – NVIDIA Performance Primitives library

- nvGRAPH – NVIDIA Graph Analytics library
- NVML – NVIDIA Management Library
- NVRTC – NVIDIA Runtime Compilation library for CUDA C++

CUDA 8.0 comes with these other software components:

- nView – NVIDIA nView Desktop Management Software
- NVWMI – NVIDIA Enterprise Management Toolkit
- GameWorks PhysX – is a multi-platform game physics engine

CUDA 9.0–9.2 comes with these other components:

- CUTLASS 1.0 – custom linear algebra algorithms,
- NVIDIA Video Decoder was deprecated in CUDA 9.2; it is now available in NVIDIA Video Codec SDK

CUDA 10 comes with these other components:

- nvJPEG – Hybrid (CPU and GPU) JPEG processing

CUDA 11.0-11.8 comes with these other components:

- CUB is new one of more supported C++ libraries
- MIG multi-instance GPU support
- nvJPEG2000 – JPEG 2000 encoder and decoder

5.1.6 TENSORBOARD

In machine learning, to improve something we often need to be able to measure it. TensorBoard is a tool for providing the measurements and visualizations needed during the machine learning workflow.

- Tracking and visualizing metrics such as loss and accuracy
- Visualizing the model graph (ops and layers)
- Viewing histograms of weights, biases, or other tensors as they change over time
- Projecting embeddings to a lower dimensional space
- Displaying images, text, and audio data
- Profiling TensorFlow programs
- And much more

6. METHODOLOGY

This segment gives a short outline of the methodology utilized in this project. Here the developed approach mainly covers four significant stages. The first stage is mainly related to the collection of the data(images) and labelling them, the second stage includes model development, the third stage consists of model training, and in the last stage testing of the model is done. Apart from this approach, a short portrayal of the various libraries and tools utilized in this project has also been explained. The most important necessary libraries used in this project are NumPy, Json, Matplotlib, OS, PyTorch, Detectron2, and OpenCV.

6.1 LIBRARIES USED

6.1.1 NumPy

An array of the multidimensional matrix which supports high-level mathematical calculations. Operations on array included in mathematics such as algebraic, statistical, and trigonometric patterns can be performed with the assistance of NumPy. The Image gets converted to matrix form. The Matrix form of the image is used for interpretation and analysis by using the Convolutional Neural Network. Afterward, the annotations of the image changed to a NumPy array style. At last, the exact labels of the images are included in the dataset. SciPy is also created on this. It offers more noteworthy execution which works on NumPy arrays and is needed for different engineering and logical applications.

6.1.2 Matplotlib

Plotting functions for python programming languages are supported by Matplotlib. In this project, it will be used to draw the bounding boxes to display the image name and score range. The bounding box is utilized to show the object detection name with a score range of an image. The Matplotlib comes up with an object-oriented application programming interface. The NumPy is one of the mathematical numerical expansion of Matplotlib.

6.1.3 OS

OS is one of the import libraries of python. It is utilized to provide a path of operating system dependent functionality and it is also used for the manipulation of a path. It comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality. **os.path** module is sub-module of OS module in Python used for common pathname manipulation. **os.path.join()** method in Python join one or more path components intelligently. This method concatenates various path components with exactly one directory separator (‘/’) following each non-empty part except the last path component. If the last path component to be joined is empty then a directory separator (‘/’) is put at the end. If a path component represents an absolute path, then all previous components joined are discarded and joining continues from the absolute path component.

6.1.4 pycocotools

Pycocotools is a Python API that assists in loading, parsing and visualizing the annotations of image datasets in COCO format.

6.1.5 Pillow

Python Imaging Library is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux. The latest version of PIL is 1.1.7, was released in September 2009 and supports Python 1.5.2–2.7.

Development of the original project, known as PIL, was discontinued in 2011. Subsequently, a successor project named Pillow forked the PIL repository and added Python 3.x support. This fork has been adopted as a replacement for the original PIL in Linux distributions including Debian and Ubuntu.

Some of the file formats supported are PPM, PNG, JPEG, GIF, TIFF, and BMP. It is also possible to create new file decoders to expand the library of file formats accessible.

Capabilities

PIL offers several standard procedures for image manipulation. These include:

- per-pixel manipulations,
- masking and transparency handling,

- image filtering, such as blurring, contouring, smoothing, or edge finding,
- image enhancing, such as sharpening, adjusting brightness, contrast or color,
- adding text to images and much more.

6.1.6 opencv-python

OpenCV is utilized for the analysis of a wide range of images and videos, similar to the recognition of the face, and object detection, editing of an image, advanced robotic vision, optical character identification, and significantly more. Image processing is done through OpenCV. It focuses on real-time computer vision. Here in this model by utilizing the OpenCV python library, python scripts will be written to test the newly trained waste detection classifier to read the images for evaluation and inference.

6.1.7 pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself. Wes McKinney started building what would become pandas at AQR Capital while he was a researcher there from 2007 to 2010.

Library features :

- DataFrame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and subsetting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.

- Time series-functionality: Date range generation[7] and frequency conversions, moving window statistics, moving window linear regressions, date shifting and lagging.
- Provides data filtration.

6.1.8 json

JSON stands for JavaScript Object Notation and is a lightweight format for storing and transporting data. JSON is often used when data is sent from a server to a web page.
Python has the built-in module **json**, which allow us to work with JSON data.

JSON Data Types

A JSON object is similar to a Python dictionary, but has the following differences:

- JSON Keys are always string.
- Strings are always enclosed with double quotes.
- A JSON boolean start with lowercase letters.
- **null** is the JSON equivalent of Python **None**.

The data types JSON supports are:

- String
- Number
- boolean
- null
- Object
- Array

6.1.9 glob

The glob module is a useful part of the Python standard library. “glob” (short for global) is used to return all file paths that match a specific pattern. We can use glob to search for a specific file pattern, or perhaps more usefully, search for files where the filename matches a certain pattern by using wildcard characters. According to Wikipedia, “**glob** patterns specify sets of filenames with wildcard characters”.

These patterns are similar to regular expressions but much simpler.

- Asterisk (*): Matches zero or more characters
- Question Mark (?) Matches exactly one character

While “glob” can be used to search for a file with a specific filename, I find it especially handy for reading in several files with similar names. After identifying these files, they can then be concatenated into one dataframe for further analysis.

6.1.10 torch

PyTorch is a machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing, originally developed by Meta AI and now part of the Linux Foundation umbrella. It is free and open-source software released under the modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface.

A number of pieces of deep learning software are built on top of PyTorch, including Tesla Autopilot, Uber's Pyro, Hugging Face's Transformers, PyTorch Lightning, and Catalyst. It provides two high-level features:

- Tensor computing (like NumPy) with strong acceleration via Graphics Processing Units (GPU)
- Deep neural networks built on a tape-based automatic differentiation system

It provides the following key features:

- **Tensor computation** : Similar to NumPy array -- an open source library of Python that adds support for large, multidimensional arrays -- tensors are generic n-dimensional arrays used for arbitrary numeric computation and are accelerated by graphics processing units. These multidimensional structures can be operated on and manipulated with application program interfaces (APIs).
- **TorchScript** : This is the production environment of PyTorch that enables users to seamlessly transition between modes. TorchScript optimizes functionality, speed, ease of use and flexibility.
- **Dynamic graph computation** : This feature lets users change network behavior on the fly, rather than waiting for all the code to be executed.
- **Automatic differentiation** : This technique is used for creating and training neural networks. It numerically computes the derivative of a function by making backward passes in neural networks.

- **Python support** : Because PyTorch is based on Python, it can be used with popular libraries and packages such as NumPy, SciPy, Numba and Cynthon.
- **Variable** : The variable is enclosed outside the tensor to hold the gradient. It represents a node in a computational graph.
- **Parameter** : Parameters are wrapped around a variable. They're used when a parameter needs to be used as a tensor, which isn't possible when using a variable.
- **Module** : Modules represent neural networks and are the building blocks of stateful computation. A module can contain other modules and parameters.
- **Functions** : These are the relationships between two variables. Functions don't have memory to store any state or buffer and have no memory of their own.

6.1.11 torchvision

This library is part of the PyTorch project. PyTorch is an open source machine learning framework. This package consists of popular datasets, model architectures, and common image transformations for computer vision.

Features

- Since it is an accompaniment to PyTorch, it automatically comes with the GPU support. (So, it is FAST !)
- Its development philosophy is to be simple in implementation (eg: without an extensive argument set for its functions) . The developers have kept it separately from PyTorch to keep it lean and lightweight.
- It is ready to use (comes with sample data-set (CIFAR10, CelebA etc) , some commonly used pre-trained models (ResNet18, maskRCNN_resnet50 etc) and even sample starter codes for some of the typical AI/ Machine Learning Problems (Image Classification, Semantic Segmentation , Keypoint detection etc) — all inbuilt into its library
- It is developed and maintained by the Facebook AI team, and supported by the python community.

6.1.12 PyYAML

YAML is a data serialization format designed for human readability and interaction with scripting languages. PyYAML is a YAML parser and emitter for Python. It features a complete YAML 1.1 parser, Unicode support, pickle support, capable extension API, and sensible error messages. PyYAML supports standard YAML tags and provides Python-specific tags that allow to represent an arbitrary Python object. And it is applicable for a broad range of tasks from complex configuration files to object serialization and persistence.

6.1.13 pip

Pip is a package-management system written in Python and is used to install and manage software packages. The Python Software Foundation recommends using pip for installing Python applications and its dependencies during deployment. Pip connects to an online repository of public packages, called the Python Package Index. Pip can be configured to connect to other package repositories (local or remote), provided that they comply to Python Enhancement Proposal 503. Most distributions of Python come with pip preinstalled. Python 2.7.9 and later (on the python2 series), and Python 3.4 and later include pip (pip3 for Python 3) by default.

6.1.14 Detectron2

A PyTorch-based modular object detection library, it is a next-generation open-source object detection system from Facebook AI Research. With the repo we can use and train the various state-of-the-art models for detection tasks such as bounding-box detection, instance and semantic segmentation, and person keypoint detection and also Cascade R-CNN, Panoptic FPN, and TensorMask. It is the second iteration of Detectron, originally written in **Caffe2**. The Detectron2 system allows you to plug in custom state of the art computer vision technologies into your workflow.

6.2 PROCESS FLOW DIAGRAM

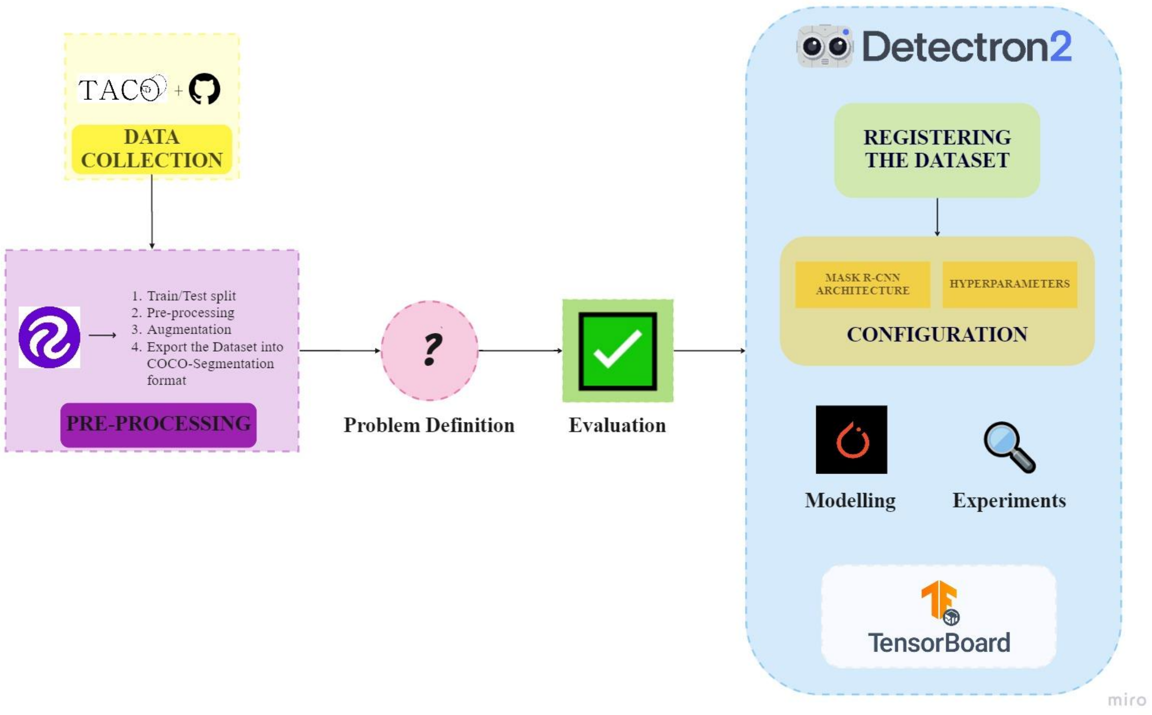


Fig. 8 Process Flow diagram of the project

6.3 DATA COLLECTION

In the online GitHub Repository, the open image dataset called Trash Annotations in Context (TACO) by the authors *Pedro F Proença* and *Pedro Simões* was downloadable as zip file which was mainly used in the classification of wastes which was found in the wild. It contains photos of litter taken under diverse environments, from tropical beaches to London streets. These images are manually labeled and segmented according to a hierarchical taxonomy to train and evaluate object detection algorithms.

Some of the images are present in the Dataset,

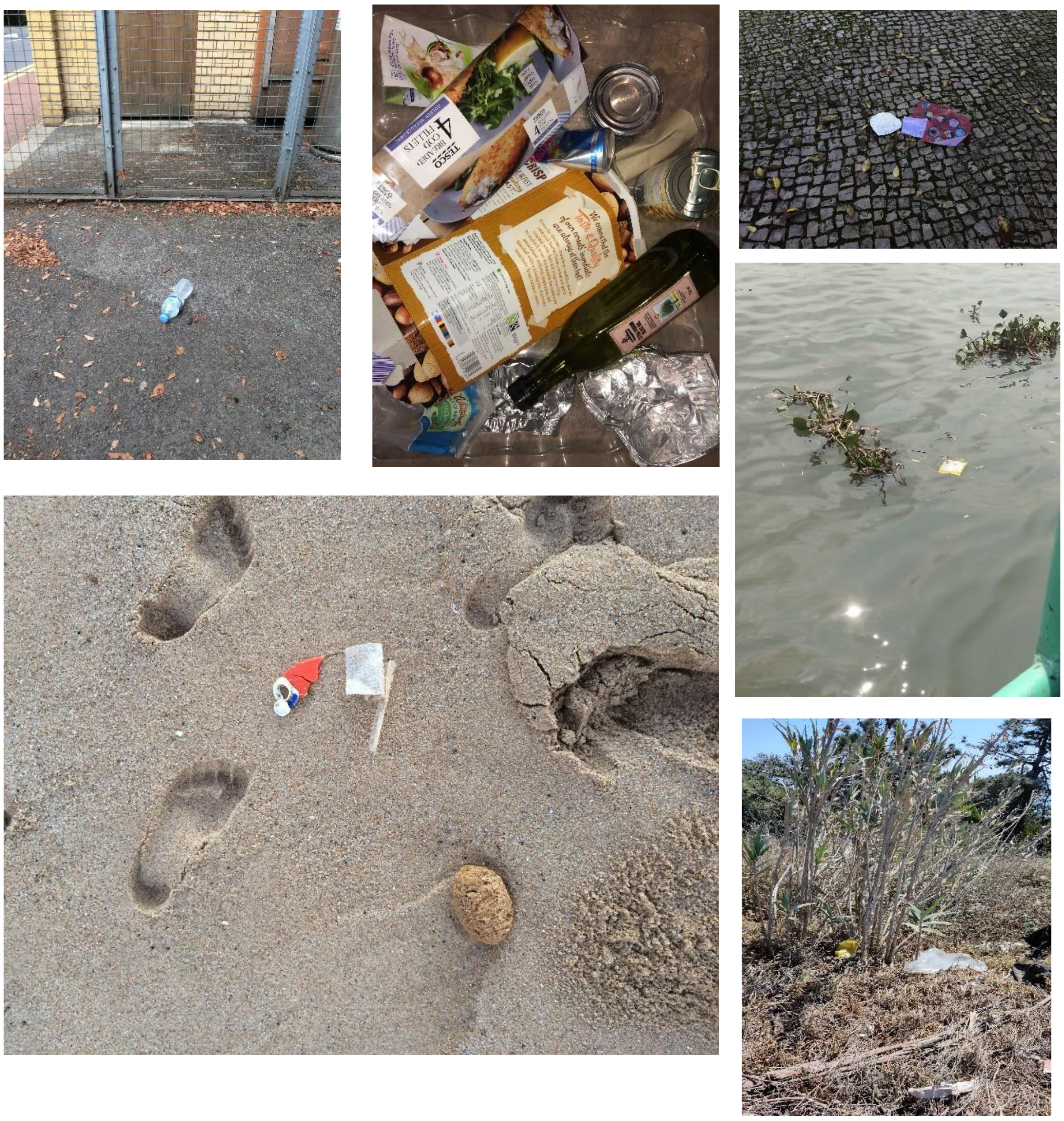


Fig. 9 Dataset images of plastic bottle, glass bottle, plastic wrappers, food can, drink can and some other plastic and wrappers are found in the various backgrounds

There were around 1500 pictures with 60 different classes present in the dataset, such as

- Aerosol
- Aluminium blister pack
- Aluminium foil
- Battery
- Broken glass
- Carded blister pack
- Cigarette
- Clear plastic bottle
- Corrugated carton
- Crisp packet
- Disposable food container
- Disposable plastic cup
- Drink can
- Drink carton
- Egg carton
- Foam cup
- Foam food container
- Food Can
- Food waste
- Garbage bag
- Glass bottle
- Glass cup
- Glass jar
- Magazine paper
- Meal carton
- Metal bottle cap
- Metal lid
- Normal paper
- Other carton
- Other plastic bottle
- Other plastic container
- Other plastic cup
- Other plastic wrapper
- Other plastic
- Paper bag
- Paper cup
- Paper straw
- Pizza box
- Plastic bottle cap
- Plastic film
- Plastic glooves
- Plastic lid
- Plastic straw
- Plastic utensils
- Polypropylene bag
- Pop tab
- Rope - strings
- Scrap metal
- Shoe
- Single-use carrier bag
- Six pack rings
- Spread tub
- Squeezable tube
- Styrofoam piece
- Tissues
- Toilet tube
- Tupperware
- Unlabeled litter
- Wrapping paper
- Plastified paper bag

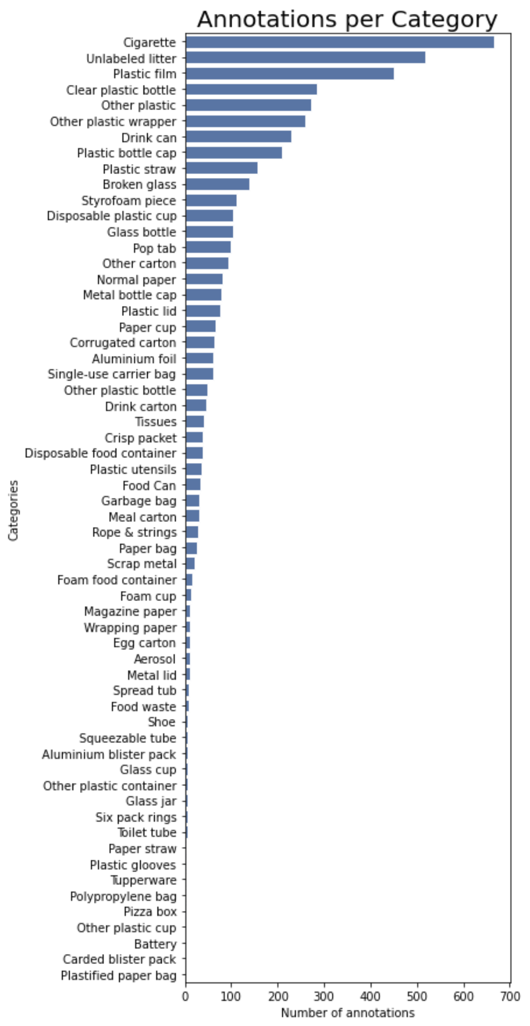
In the downloaded zip folder it contains python script file called '*download.py*' to download the images or in the data folder it contains 15 batches of folder, in that each folder there are approximately 90 to 120 images are found and there contains respective annotation file in json format and these annotations are provided in COCO format. And also another folder called detector which is uses *MatterPort* Mask R-CNN algorithm to detect the wastes. But the annotations of these images were specifically for that respective batch folder not for the all images, so we use Roboflow platform for Pre-processing, Image augmentations and to split the images into Train, Validation and Test folder. And finally export the dataset into COCO-Segmentation format for easily train the Detectron2 model.

6.3.1 EXPLORATORY DATA ANALYSIS

From the image dataset, the number of images, annotations, categories and super categories are found below :

- Number of super categories: 28
- Number of categories: 60
- Number of annotations: 4784
- Number of images: 1500

From the Annotations we classify the 60 different categories and its 28 super categories, among all Cigarette is found high in the dataset and Plastic bag & wrapper super category group is very high across the dataset.



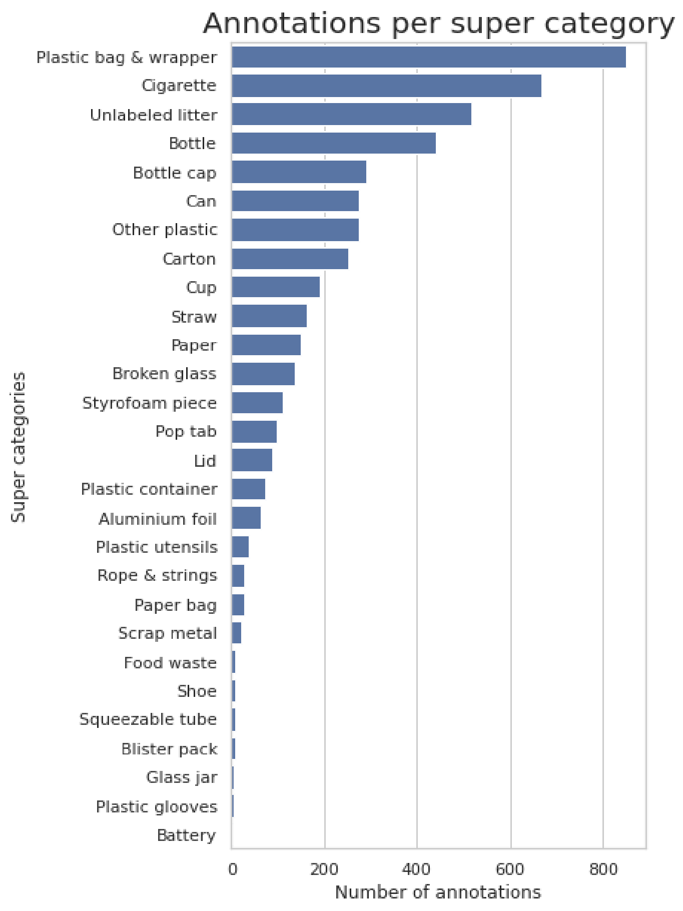


Fig. 10 & 11 Annotations per Category & Annotations per Super category

As these wastes are found in the different types of background scenes, so among all the scenes the Vegetation & Sand, Dirt, Pebbles background seems to be equal number of images or wastes found.

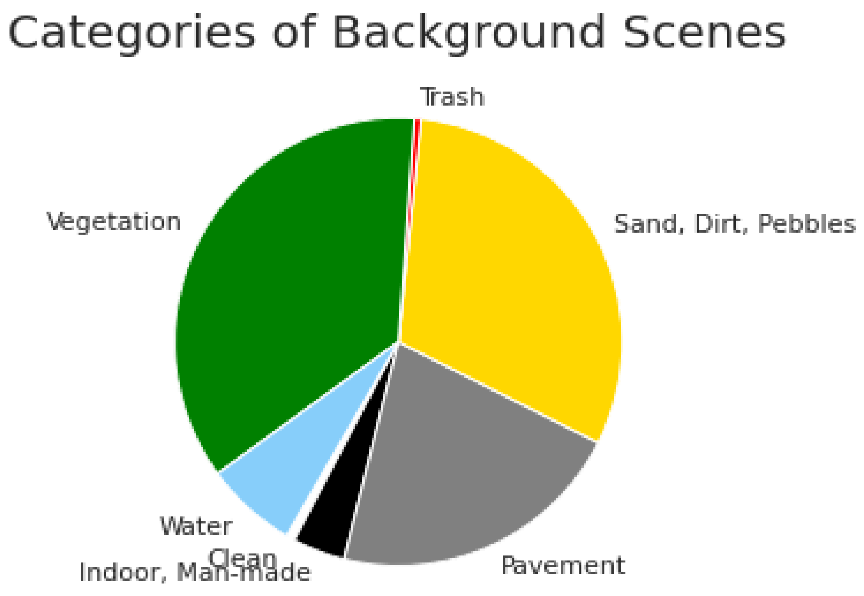


Fig. 12, Categories of Background scenes of the wastes found

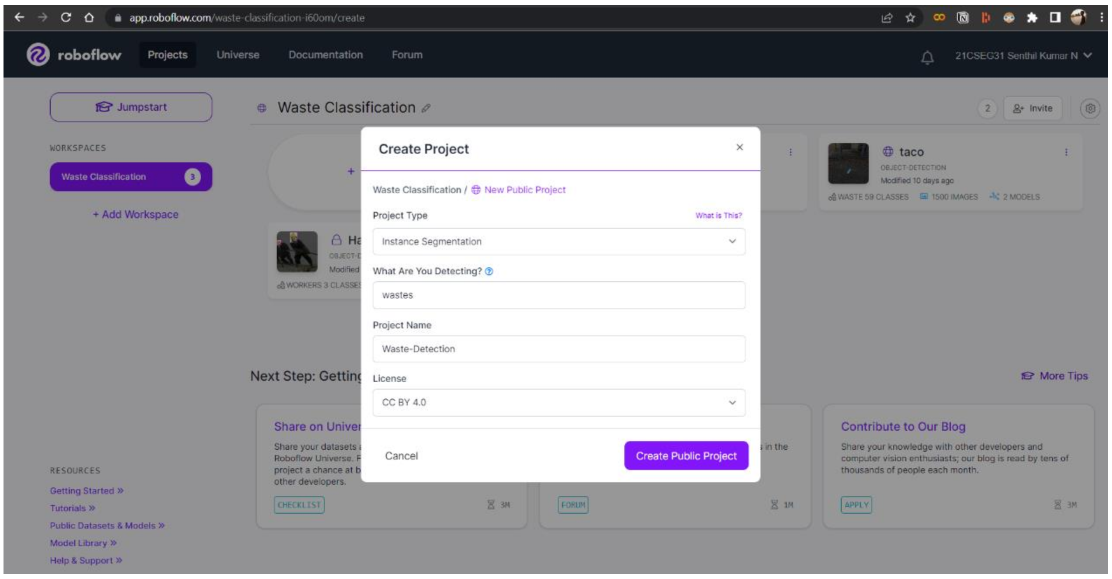
6.4 PRE-PROCESSING

The Roboflow platform is used mainly for splitting the dataset in train, test and validataion sets and also this platform gives many facilities to pre-process the images such as resize, auto-orient, flip, shear, rotate etc to generate different samples from original dataset to the training set to learn the model well. It very much useful for changing the annotations format into many formats such PASCAL, VOC, YOLO, JSON, CSV and TXT, and once pre-processing steps over, we can also train our model in their platform and deploy it as well.

The following are the steps for uploading the dataset and for the pre-processing steps as well.

STEP 1:

To create a Project in the Roboflow platform and fill the required fields to create the project in our respective account.



Create Project

Waste Classification / New Public Project

Project Type: Instance Segmentation

What Are You Detecting?: wastes

Project Name: Waste-Detection

License: CC BY 4.0

Cancel | Create Public Project

Fig. 13, Create project in the Roboflow platform

STEP 2 :

Upload the images of 15 batches extracted from the zip folder, then upload it in the platform.

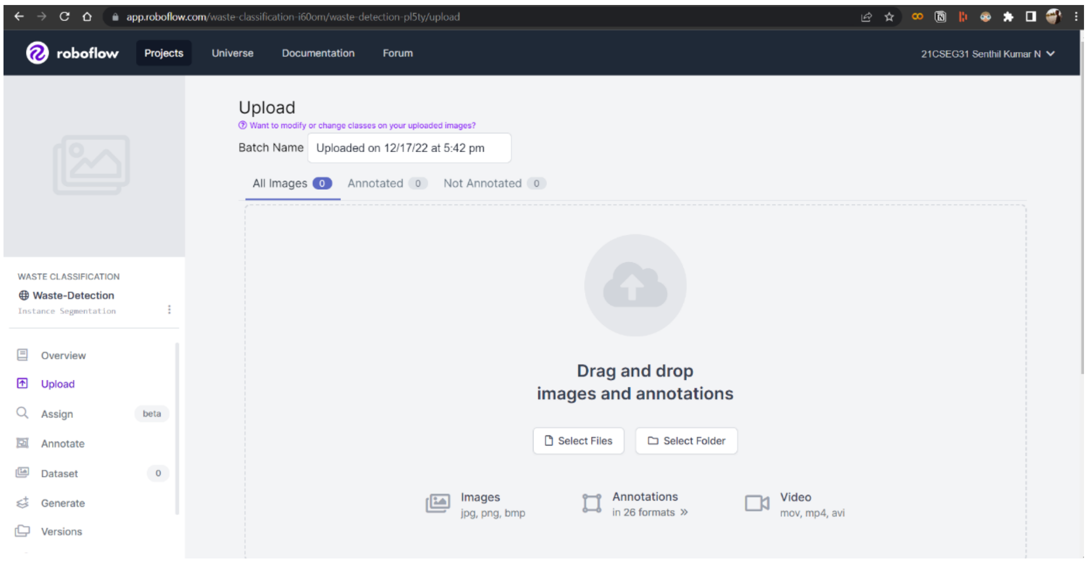


Fig. 14.a, Upload the images in the platform

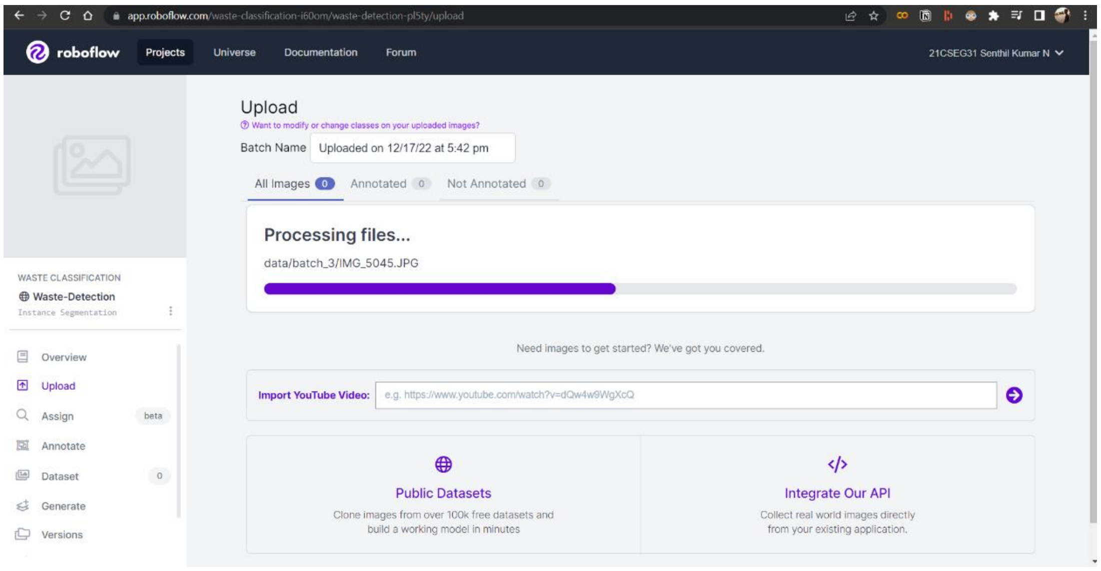
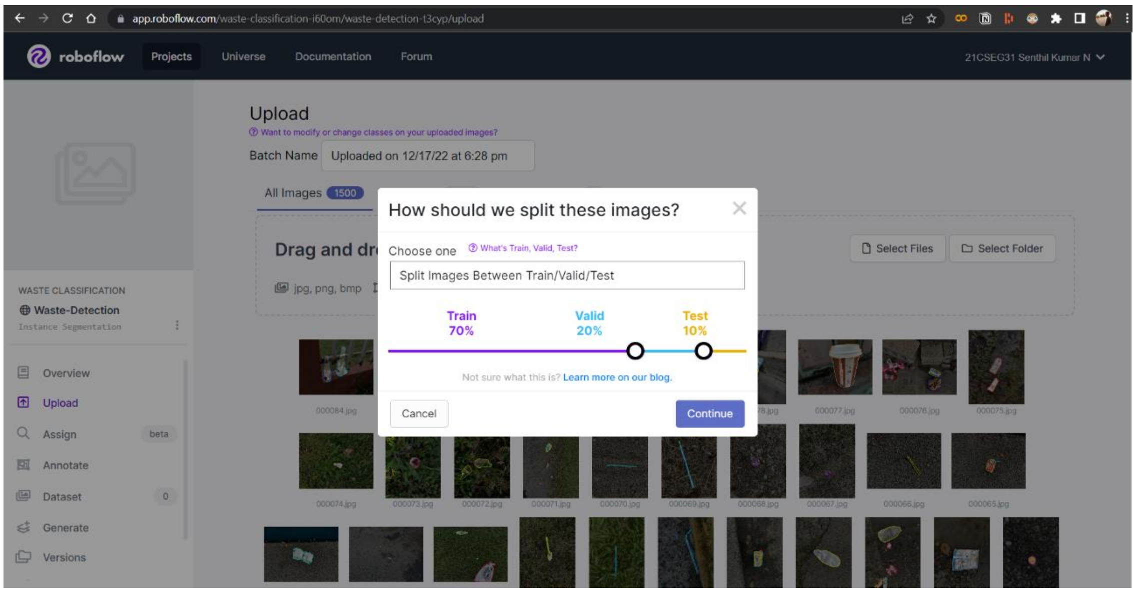

app.roboflow.com/waste-classification-i60om/waste-detection-pl5ty/upload
roboflow
Projects
Universe
Documentation
Forum
21CSEG31 Senthil Kumar N
Upload
Want to modify or change classes on your uploaded images?
Batch Name
Uploaded on 12/17/22 at 5:42 pm
All Images 0
Annotated 0
Not Annotated 0
Processing files...
data/batch_3/IMG_5045.JPG
Need images to get started? We've got you covered.
Import YouTube Video:
e.g. https://www.youtube.com/watch?v=dQw4w9WgXcQ
Public Datasets
Clone images from over 100k free datasets and build a working model in minutes
Integrate Our API
Collect real world images directly from your existing application.
WASTE CLASSIFICATION
Waste-Detection
Instance Segmentation
Overview
Upload
Assign
beta
Annotate
Dataset
0
Generate
Versions

Fig. 14.b, Images are uploading in the platform

STEP 3 :

Once the images are uploaded, then we can split the images dataset into the standard ratio of 70:20:10 for the Training set into 70%, Validation set into 20% and Test set into 10%.



app.roboflow.com/waste-classification-i60om/waste-detection-t3cyp/upload

Upload

Batch Name: Uploaded on 12/17/22 at 6:28 pm

All Images 1500

How should we split these images?

Choose one: Split Images Between Train/Valid/Test

Train 70%

Valid 20%

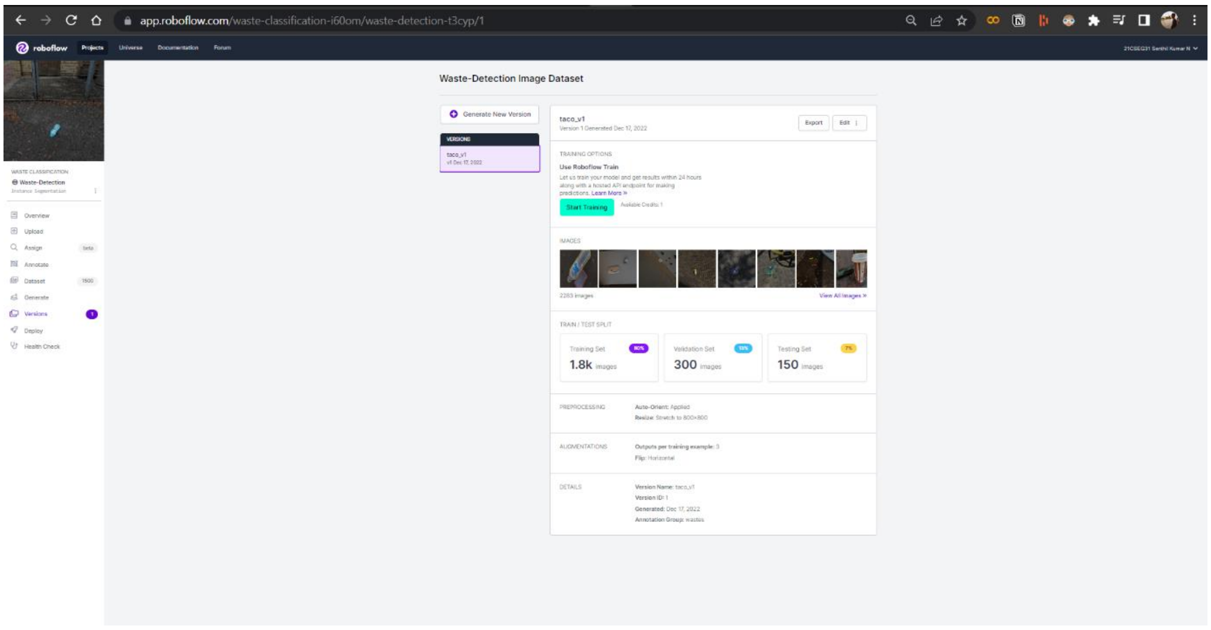
Test 10%

Not sure what this is? Learn more on our blog.

Fig. 15, Split the image dataset.

STEP 4 :

Then, we select the appropriate pre-processing techniques available in the platform, here we applied Auto-orient and Resize to the images into 800 x 800 respectively. And apply augmentation techniques that are also available in the platform such as image-level augmentations and bounding box level augmentations, here we applied image-level augmentations such as flipping the image into horizontal level only for generating samples from the original dataset. Finally, generate the dataset and it takes some time to complete.



Waste-Detection Image Dataset

taco_v1
Version 1 Generated Dec 17, 2022

TRAINING OPTIONS
Use Roboflow Train
Let us train your model and get results within 24 hours along with a hosted API endpoint for making predictions. Learn More »

IMAGES
2283 images

TRAIN / TEST SPLIT
Training Set 80% 1.8k images
Validation Set 13% 300 images
Testing Set 7% 150 images

PREPROCESSING
Auto-Orient: Applied
Resize: Stretch to 800×800

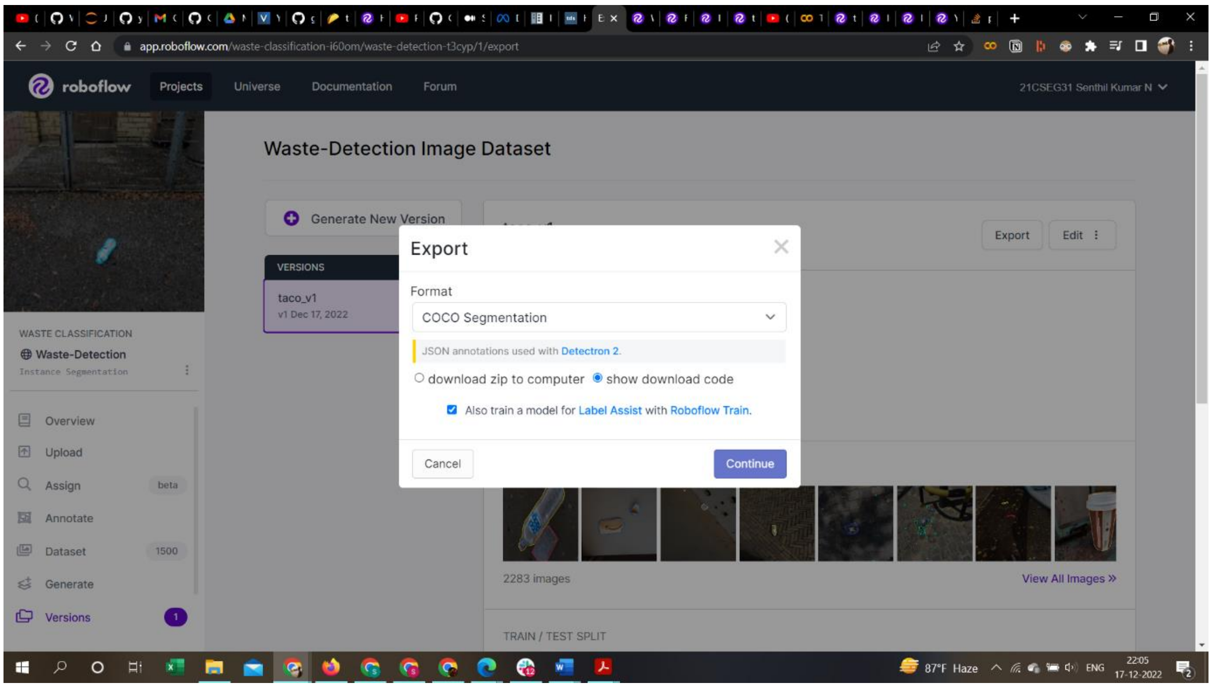
AUGMENTATIONS
Outputs per training example: 3
Flip: Horizontal

DETAILS
Version Name: taco_v1
Version ID: 1
Generated: Dec 17, 2022
Annotation Group: wastes

Fig. 15, After the Pre-processing and augmentations techniques applied, the training set has 1,850 images are generated, and 300 images are generated for validation set, and 100 images for test set.

STEP 5 :

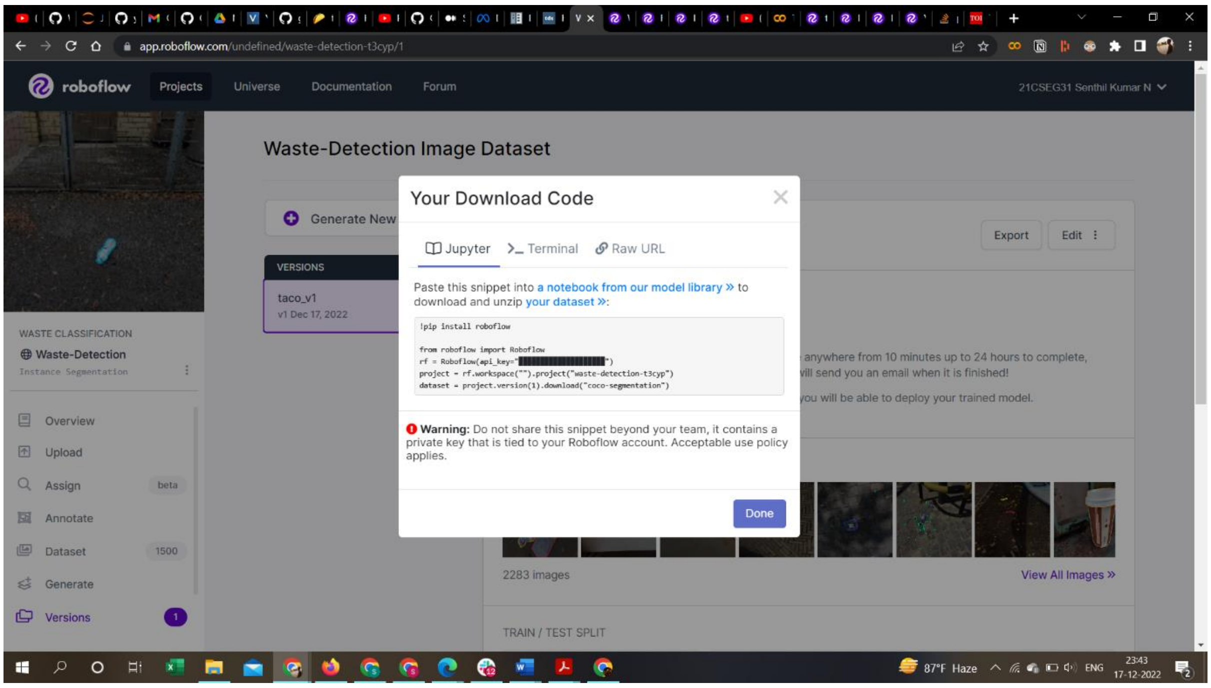
Finally export the dataset into the COCO-Segmentation format and click show download code to copy and paste the code into the Colab IDE.



Export

Format: COCO Segmentation
JSON annotations used with Detectron 2.
download zip to computer [✓ show download code]
[✓] Also train a model for Label Assist with Roboflow Train.

Fig. 16.a, Export the dataset


Your Download Code

Paste this snippet into a notebook from our model library » to download and unzip your dataset »:

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="[redacted]")
project = rf.workspace("").project("waste-detection-t3cyp")
dataset = project.version(1).download("coco-segmentation")
```

Warning: Do not share this snippet beyond your team, it contains a private key that is tied to your Roboflow account. Acceptable use policy applies.

Fig. 16.b, Copy the code and paste it into the IDE.

Finally the dataset have this following structure to register and to train the model.

```
        dataset-directory/
├── README.dataset.txt
├── README.roboflow.txt
├── train
│   ├── train-image-1.jpg
│   ├── train-image-1.jpg
│   ├── ...
│   └── _annotations.coco.json
├── test
│   ├── test-image-1.jpg
│   ├── test-image-1.jpg
│   ├── ...
│   └── _annotations.coco.json
└── valid
    ├── valid-image-1.jpg
    ├── valid-image-1.jpg
    ├── ...
    └── _annotations.coco.json
```

6.5 MODEL CONFIGURATION

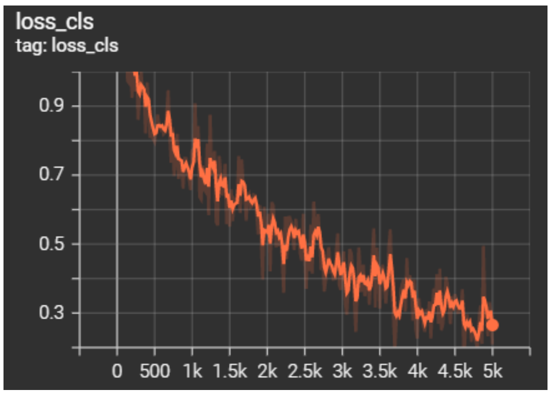
Here in this Detectron2 Engine, first we have to register the dataset to the Detectron2 Engine to recognize the images and json annotations file. We use Mask R-CNN architecture in yaml file and set the maximum number of iterations into 5000 for better learning to the model and evaluation period into 500, after every 500 iterations we want to evaluate the validation set side-by-side. The Learning rate has set to 0.001 and we have number of classes are 60. Finally define the output directory path to store the metrics file, last check point, and the model as stored in PyTorch format as '.pth'.

Here the model is configured its default GeneralizedRCNN Architecture and backbone as Feature Pyramid Network and ResNet. And the next layer is Region Proposal Network for proposal generator and Region of Interest heads as StandardROIHeads, and box head is set as FastRCNNConvFCHead, box predictor as FastRCNNOutputLayers, mask pooler as ROIpooler, mask head as MaskRCNNConvUpsampleHead is configured, because of the Hyperparameter architecture as Mask R-CNN is set to detect and classify the wastes in the images and build masks in the detected wastes respectively.

6.6 MODEL TRAINING

The Detectron2 Engine is now configured in default and we only changed the Mask R-CNN architecture hyperparameter. The model is trained on 5000 iterations with 500 evaluation period with the help of NVIDIA Tesla T4 GPU and CUDA supported PyTorch library provided by the Google Colab with 16GB RAM and the total time for the training process was near about 45 minutes.

The value of total loss was initially high during the training process. As the progression of the training occurred the loss value started getting lower. When the training started at first, the value of the initial loss was near about 5.67 and then the value of the loss gradually dropped to a point at 0.66. And for the class loss was near about 3.975 and dropped finally in the last iteration to 0.208.



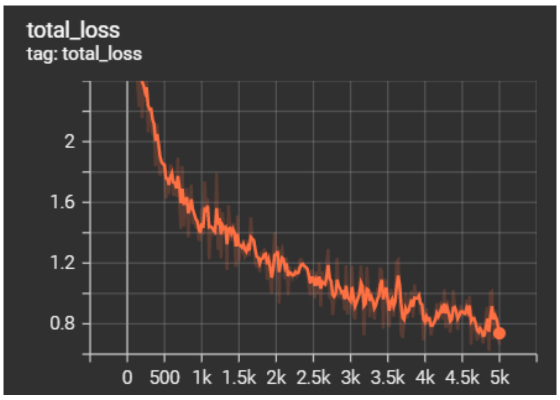


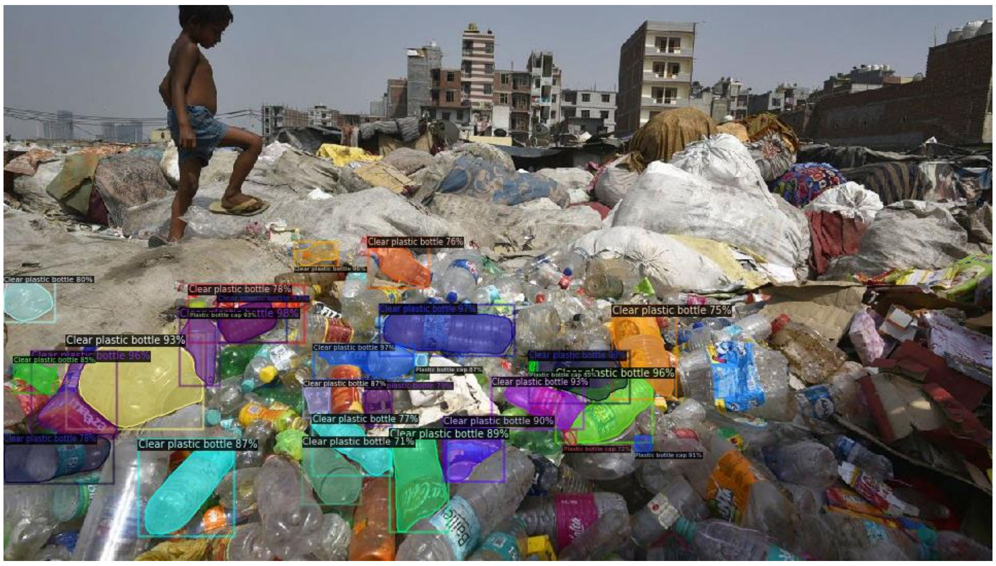
Fig. 17, Class Loss and Total Loss graphs of the training process.

6.7 MODEL TESTING

Testing the trained model is the last part of the process. After the training gets completed, we use the trained weights to predict the testing set and visualizing it. This is used for identifying and classifying the objects in a real-time feed utilizing webcam, images, and videos.. In this stage, the trained model's weights are tested utilizing the test dataset. At last, by utilizing the OpenCV python library, python scripts are written to test the newly trained object detection classifier on any images. Finally, the tested model classifies waste materials into 60 categories (Cardboard, Plastic, Metal, Paper, Glass bottle etc.,).

7. RESULT

In this project, the outcomes of the created model will be examined. The created model works reasonably on the test data. Out of 150 images, 60 – 70 images were quite accurately got predicted by the model which concludes that the Average Precision rate of the model is near about 6.7%. The model precisely classifies the type of waste materials by detecting the type of objects. The images of the testing data were utilized for testing the result of the created model. For the detection of the waste materials, the images of the test data have been specifically included in the testing code.



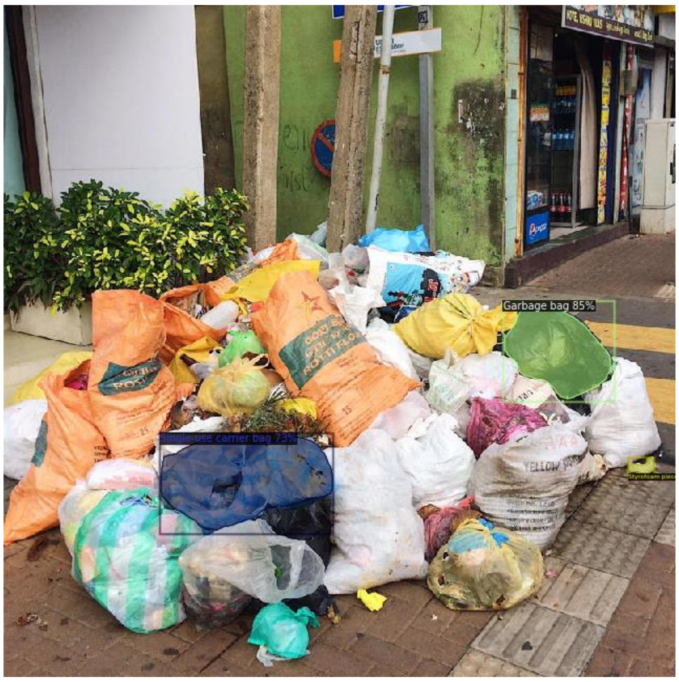


Fig. 18.a & 18.b, Accurate prediction of the model of test images

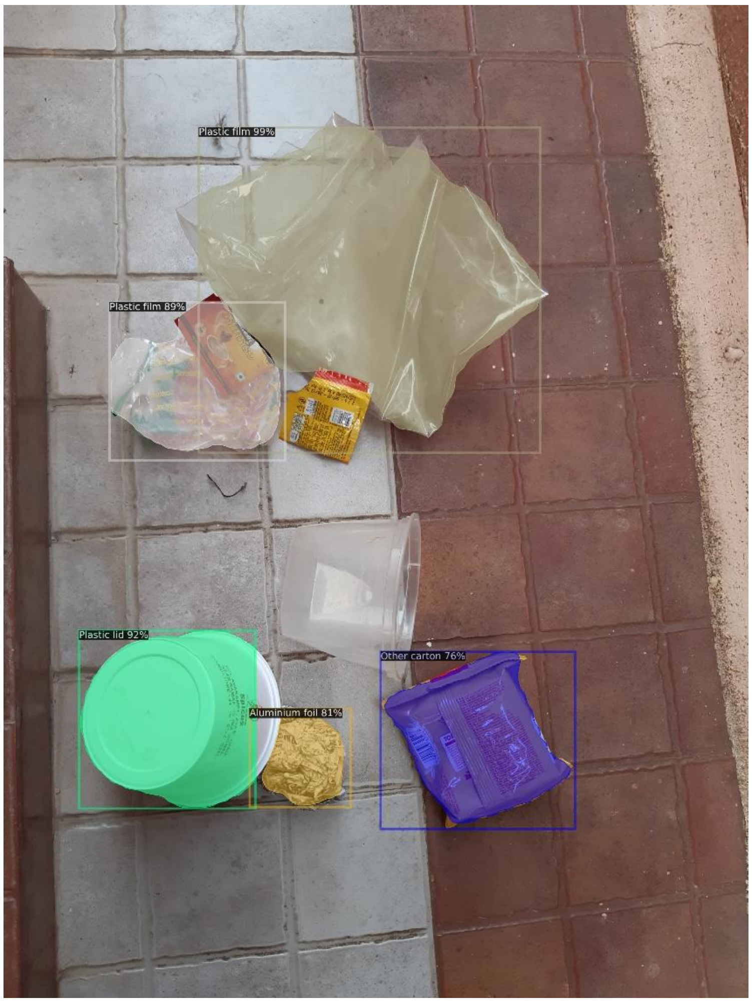


Fig. 18.c, Predicted approximately and also not detected some plastic wrappers.

Some wrong prediction has also been done by the model and the accuracy rate got decreased on the images which are very small such as Cigarette and Ropes and strings, other plastics, aluminium foil etc., because of the class imbalance in the dataset. Apart from that as the number of trash images used in the training dataset was very less in number that's why the model is having some difficulties correctly predict the trash objects. Figure 19 shows a few wrong predictions of the model.

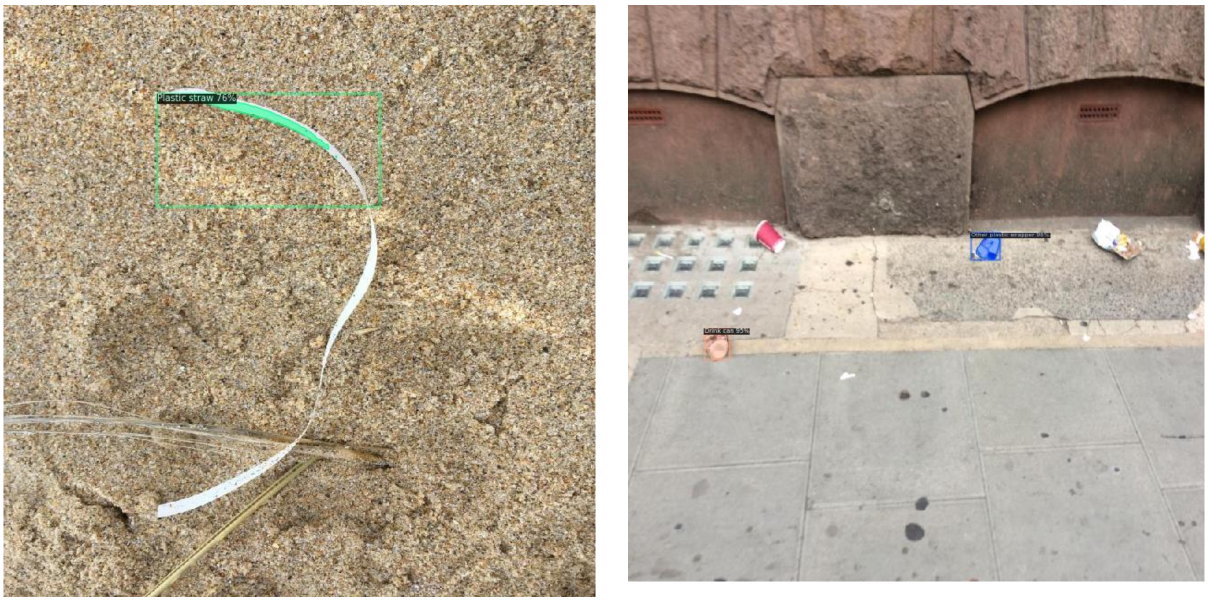


Fig. 19, Wrongly predicted by the model.

8. CONCLUSION AND FUTURE WORK

To limit the impact caused by incorrect disposal of trash, this project introduced an automated waste detection framework using Detectron2. Thus, for implementation, the framework worked with a large dataset of images, training algorithms, for predicting instance segmentation and classification. In this project, we have demonstrated that how the classification of waste materials in 60 categories (Aluminium, Cardboard, Metal, Glass, Paper, Plastic wrappers, etc) is done on multiple objects in a single image with the help of utilizing the method of Mask R-CNN algorithm. In the past, most of the experiments related to this study are done on a single object of an image with 3 or 4 categories of classification utilizing other machine learning techniques. An improvement in waste materials classification is provided with our methodology. The detection of waste materials is done correctly maintaining a quite good accuracy level. The detection of the waste materials is not only confined to images, but it can also detect and classify the waste materials from any or real-time images.

The main issue of this project was the dataset which includes images that are slightly different from local waste materials. This is the reason for which the model predicted wrongly on a few local waste images. The future work that should be taken for consideration is a similar method however an improvement in the datasets by including pictures of locally taken waste materials. There is a need of attaching images of the waste materials in the training dataset which are not clean and looks dirty. This will help the model predicting actual local waste materials which include mostly dirty household items. This can assist in getting some improved classification with a higher accuracy rate. The dataset should also include many images that contain pictures of multiple waste materials so that the framework can easily be trained to predict multiple objects in a single image without giving any error in the detection process. Further research of this project should also consider including different types of other bulky waste categories in the dataset. By intensifying the list of categories this framework will get more developed and would surely help in the improvement of the proper waste management process. Analysis and comparison of various models such as Faster R-CNN, YOLO, etc. can also be done in the future. The future work also aims to implement this technology in a mobile platform so that it becomes very easy for the user to classify waste materials and dumping the waste materials in the correct disposal bin to protect the environment and reduce pollution.

9. REFERENCES

9.1 BIBLIOGRAPHY

1. Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv:1506.01497*
2. Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick; Mask R-CNN *arXiv:1703.06870*
3. Ying Liu, Zhishan Ge, Guoyun Lv, Shikai Wang; Research on Automatic Garbage Detection System Based on Deep Learning and Narrowband Internet of Things *ResearchGate*, DOI:10.1088/1742-6596/1069/1/012032
4. SylwiaMajchrowska, AgnieszkaMikołajczyk, MariaFerlin, ZuzannaKlawikowska, Marta A.Plantykow, ArkadiuszKwasigroch, KarolMajekd; Deep learning based waste detection in natural and urban environments *ScienceDirect* https://doi.org/10.1016/j.wasman.2021.12.001

REFERENCE WEBSITES

1. FAIR's blog on Detectron2- https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/
2. Detectron2 - https://github.com/facebookresearch/detectron2
3. TACO - http://tacodataset.org/
4. TACO dataset - https://github.com/pedropro/TACO
5. Roboflow platform - https://roboflow.com/