MINI PROJECT

## TITLE: Netraa: An app for visually impaired people

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date of Perform**  **ance** | **Date of**  **Submission** | **Marks (10)** | | | | | **Sign / Remark** |
| **A** | **B** | **C** | **D** | **E** |
| **2** | **3** | **2** | **2** | **1** |
|  |  |  |  |  |  |  |  |
| **Total Marks** | | | | |
|  | | | | |

**Netraa: An app for visually impaired people**

Submitted in partial fulfilment of requirement of

### University of Mumbai

For the Degree of

### Third Year (Sem VI) of Computer Engineering

Submitted By

Karunesh Palekar(42)

Dikshita Padte(40)

Ekta Vayanan(71)

Under Guidance of

### Prof. Deepti Vijay Chandran

Department of Computer Engineering

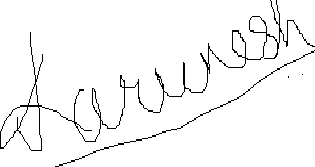
### Smt. Indira Gandhi College of Engineering

Affiliated to University of Mumbai (2021-2022)

## DECLARATION

We declare that this written submission represents our own ideas in our own words and where others ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any act/data/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

###### (Dikshita Padte [41])



(Karunesh Palekar [43])

Ekta Vayanan [74]

**CERTIFICATE**

This is to certify that the Mini Project entitled ,**“** **Netraa: An app for visually impaired people”** is a bonafide work by **Karunesh Palekar (42) , Dikshita Padte ( 40) , Ekta Vayanan ( 74 )** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the completion of **“Mini Project (SEM - VI)”** in **“Computer Engineering” .**

#### (Prof. Deepti Vijay Chandran)

Internal Guide

#### (Dr. K.T Patil ) (Dr. Sunil Chavan)

Head of Department Principal

**MINI PROJECT APPROVAL**

This Mini Project entitled **“Netraa: An app for visually impaired people”** by **Karunesh Palekar( 42 ) , Dikshita Padte ( 40 ) , Ekta Vayanan (71)** is approved for the degree of **Bachelor of Engineering (SEM-VI)** in **Computer Engineering.**

## Examiner

**1………………………………………**

(Internal Examiner Name & Sign

#### 2…………………………………………

(External Examiner name & Sign)

Date:

## ABSTRACT

**Title**: **Netraa: An app for visually challenged**

Visually challenged people face lot of problems in ther day to day life making their life a bit more challenging than others. For this there is a need of various technologies which could help them to overcome these difficulties in their day to day life. For this we have come with a project which could be the eyes of visually challenged people enabling them to visualize the surrounding, use the smart phone and carry out all the activities which a normal person does.

In this project we will be implementing EfficientDet Object Detection Model to help visually

impaired people to visualize the surroundings, detect the objects in front of them and get

to know the obstacles in their path beforehand using speech.

## List of Abbreviations

1. Artificial Neural Networks(ANN)
2. Neural Network(NN)
3. Application Programming Interface(API)
4. McCulloch-Pitts (MP)
5. Machine Learning(ML)
6. TensorFlow(TF)
7. You Look Only Once Version 4 tiny( YOLO v4 tiny)
8. You Look Only Once Version 5( YOLO v5)
9. Efficient Detection(EfficientDet)

**List of Figures**

[Fig 3.1 Agile Model 9](#_TOC_250004)

Fig 3.2 Timeline Chart 10

Fig 4.1 Level 2.0 DFD 13

Fig 4.2 Architecture of TensorFlow Lite 16

[**Fig 4.3** Evaluation of all the models 17](#_TOC_250003)

**Fig 5.1** Output rom Open Image Dataset 21

Fig 5.2 Flowchart of the process design for the project 22

Fig 5.3 EfficientDet Architecture…………………………………………………………...…23

[Fig 6.1 Model Scaling](#_TOC_250004)…………………………………………………………………………26

Fig 6.2 Network Architecture of Efficient Det …26

Fig 6.3 Feature Network Design……………………………………………………………….27

Fig 6.4 Model FLOPS vs COCO accuracy……………………………………………………..28

[Fig 6.5 Output from EfficientDet using chess image9](#_TOC_250004)…………………………………… ….28

Fig 7.1 Output Image 35

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | | |  |  |  |
|  |  |  | | |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

## List of Tables

**Table 1** Result table from EfficientDet Model 37

**Table 2** EfficientDet using Android…………………………………………………………..37

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Chapt er No.** | **Content s** | **Pag e No.** |
|  | **Abstract** | **i** |
| **List of Abbreviations** | **ii** |
| **List of Figures** | **iii** |
| **List of Tables** | **iv** |
| **1** | **INTRODUCTON** | **2** |
| 1.1 Problem Statement | **2** |
| 1.2 Objective | **2** |
| 1.3 Scope | **3** |
| 1.4 Report Organization | **3** |
| **2** | **REVIEW OF LITERATURE** | **5** |
| 2.1 Survey of Existing Paper | **5** |
| 2.2 Research and Survey based on research papers | **5** |
| 2.3 Limitations on Existing System | **6** |
| **3** | **PLANNING AND FORMULATION** | **8** |
| 3.1 Feasibility Study | **8** |
| 3.2 Project Development Model | **8** |
| 3.3 Timeline Chart | **10** |
| **4** | **REQUIREMENT ANALYSIS** | **12** |
| 4.1 Hardware Requirements | **12** |
| 4.2 Software Requirements | **12** |
| 4.3 Functional Requirements | **12** |
| 4.4 Non – Functional Requirements | **12** |
| 4.5 Data Flow Diagram | **13** |
| 4.6 Project Methodology | **14** |
| **5** | **PROPROSED SYSTEM** | **18** |
| 5.1 Data Processing | **19** |
| 5.2 System Design | **21** |

|  |  |  |
| --- | --- | --- |
| **6** | **IMPLEMENTATION** | **25** |
| 6.1 Algorithm | **25** |
| 6.2 Important Implementation Techniques | **25** |
|  | **RESULT** | **35** |
| 7.1 Output from the EfficientDet model | **36** |
| 7.2 Result Table of EfficientDet | **37** |
| 7.3 EfficientDet using Android | **37** |
| **8** | **CONCLUSION** | **39** |
| 8.1 Conclusion | **39** |
| 8.2 Future scope | **39** |
|  | **REFERENCES** | **40** |
| **ACKNOWLEDGEMENTS** | **41** |

# CHAPTER 1 INTRODUCTION



## INTRODUCTION

Dealing with sight loss, already, is a challenge in itself. The lack of emotional support at diagnosis centers, the limited accessibility to activities and information, the societal stigma and the lack of unemployment, are all factors frequently leading blind or low vision individuals in isolation. Work is a whole different matter if you’re visually impaired. Considering the lack of accessible work and working spaces, one can already imagine why hiring a visually impaired individual would be considered a liability for a company. This has a negative impact on the confidence and emotional well-being of the visually impaired, while it totally cripples their economic independence. Having little to no opportunity to support oneself, blind or low vision individuals are incapacitated from their independence.Hence there is a need to come up with such a technology which could help them to gain equal respect and opportunities in the work space and also in the outdoor world.

#### 1.1PROBLEM STATEMENT

By taking into consideration the problems of visually challenged people, this project has been implemented using EfficientDet Object Detection model where the Object Detection Model is used to identify the day to day objects and obstacles coming in front of the visually challenged person. It further helps to detect currency and text reading. Every information detected by the model would be converted into voice snippets for communication the information to the user audibly.

##### OBJECTIVES

**2**

* Collecting data in the form of images and video frames.
* Imorting the dataset.
* Input the images using OpenCV.
* To perform Object Detection using OpenCv, python, numpy, Object Detection Model(EfficientDet), Deep Learning and Computer Vision.
* To detect the objects such as chair , table, To understand that People are walking, to read text and convert it to speech, currency recognition, color identification, etc using EfficientDet object detection models.
* To draw the bounding boxes around the objects.
* To recognize the objects with the help of labels given to the model.
* Once recognized, communicating the surrounding information and the decteted objects to the user with the help of speech.

#### SCOPE

 In a world where the next great invention is expected to appear on mobile phone screens, **blind and visually impaired people have been left behind**. However, some innovative projects are trying to exploit the potential of new technologies to make their life easier.

Affordable smartphone apps have empowered the blind and visually impaired. Now, **artificial intelligence (AI)** is taking those apps' capabilities to the next level. AI and machine learning technologies, specifically computer vision, have grown sufficiently robust to improve the lives of the blind and visually impaired.

People with vision loss can do numerous things such as write documents, browse the internet and send and receive emails. Screen Reading software and special talking and Braille devices allow those of us with no vision to use computers, cell phones and other electronic devices independently. Similarly, people with low vision can use screen magnification software and devices that will allow them to see letters, pictures and other objects without having to struggle or strain their remaining vision. This technology – commonly known as assistive or adaptive technology – is continually evolving, and has removed many access barriers for people with vision loss.

Besides allowing us to carry out routine tasks at work and school, assistive technology also enables people with visual impairments to be more independent at home. We can now read the mail, listen to audio books, get step-by-step walking directions to unfamiliar places, record important information and so much more with special standalone devices designed for people with no or low vision.

#### REPORT ORGANIZATION

* + - **In Chapter 2**, we will see the Planning and formulation of project is given, usage of Prototype model and how we integrated and worked around themodeL.
    - **In Chapter 3**, we will see what are the hardware and software requirements for the project as well as what are the tools needed to be installed.
    - **In Chapter 4**, we will see the system proposed is introduced which will tell the deep specification which of the project and will tell how the different modules of system will work, the flow of the project regarding data flow, control flow and other flow of the system.
    - **In Chapter 5**, we will see the implementation of the project and the screenshots of the model working.
    - **In Chapter 6**, we will display the results accordingly.
    - **In Chapter 7**, conclusion and the future scope is mentioned.



**CHAPTER 2 :**



**REVIEW OF LITERATURE**



## REVIEW OF LITERATURE

#### Survey of existing system

It’s estimated that there are about 36 million people in the world who are blind, and a further 216 million who live with moderate to severe visual impairments. Although the World Health Organization points out that up to 80% of vision impairment around the world is avoidable with better access to treatment, the number of people who are blind or have low vision is rising as the global population ages.

But technology is playing a vital role in tearing down barriers, and artificial intelligence is making real inroads into improving accessibility.

Microsoft’s Seeing AI is an app designed to help people with low vision or who are blind. It enhances the world around the user with rich audio descriptions. It can read a handwritten note or scan a barcode and then tell the user what the product is. Point a camera at something and the app will describe how many people it can see and where they are in the image – center, top left and so on.

#### Research Paper Analysis

For a sighted person, walking along the street can mean taking in every detail that surrounds them. Microsoft Soundscape replicates that behavior by building a detailed audio map that relates what’s taking place around a person with visual impairment.

It creates layers of context and detail by drawing on location data, sound beacons and synthesized 3-D stereo sound to build a constantly updating 3-D sound map of the surrounding world.

Tremendous progresses have been made in recent years towards more accurate object detection; meanwhile, stateof-the-art object detectors also become increasingly more expensive. For example, the latest AmoebaNet-based NASFPN detector [45] requires 167M parameters and 3045B FLOPs (30x more than RetinaNet [24]) to achieve state-ofthe-art accuracy. The large model sizes and expensive computation costs deter their deployment in many real-world applications such as robotics and self-driving cars where model size and latency are highly constrained. Given these real-world resource constraints, model efficiency becomes increasingly important for object detection.

Model efficiency has become increasingly important in computer vision. In this paper, we systematically study neural network architecture design choices for object detection and propose several key optimizations to improve efficiency. First, we propose a weighted bi-directional feature pyramid network (BiFPN), which allows easy and fast multiscale feature fusion; Second, we propose a compound scaling method that uniformly scales the resolution, depth, and width for all backbone, feature network, and box/class prediction networks at the same time. Based on these optimizations and better backbones, we have developed a new family of object detectors, called EfficientDet, which consistently achieve much better efficiency than prior art across a wide spectrum of resource constraints. I

#### Limitation on Existing system

Many applications , softwares and systems have been already made in order to assist in either ways .But no software is so capable to include all the technologies which could help the visually challenged person in every sense. Visually challenged person cannot completely rely on this existing apps and models for livin g. Hence we tend to combine all the existing and more innovative technologies in this project .



# CHAPTER 3 PLANNING AND FORMULATION



## PLANNING AND FORMULATION

#### FEASIBILITY STUDY

The feasibility study is a major factor that contributes to the analysis and development of the app. The decision of the app analyst whether to design a particular app or not depends on its feasibility study. A feasibility study is undertaken whenever a possibility of the probability of improving the existing system or designing a new system.

A feasibility study helps to meet user requirements.

1. Technical-Feasibility: As far as TensorFlow & Yolo model is concerned a high processing system is required for the maximum accuracy. Our goal is to achieve maximum throughput in less configuration.
2. Operational-Feasibility: The basic operation of what Object Detection is and how it will help in object recognition should be reflected through this feasibility report.

c.Economic Feasibility: As in the future this project can be adopted by Government authorities,it will require some hardware cost for cameras and drones to make this project work on input as spatial videos.Also as scalability will increase,the demand for resources would also increase.

#### PROJECT DEVELOPMENT MODEL

* + 1. **Software development life cycle**

The entire project spanned for duration of 3 months. In order to effectively design and develop a cost-effective model Agile model was practiced.

**AGILE MODEL**

"Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

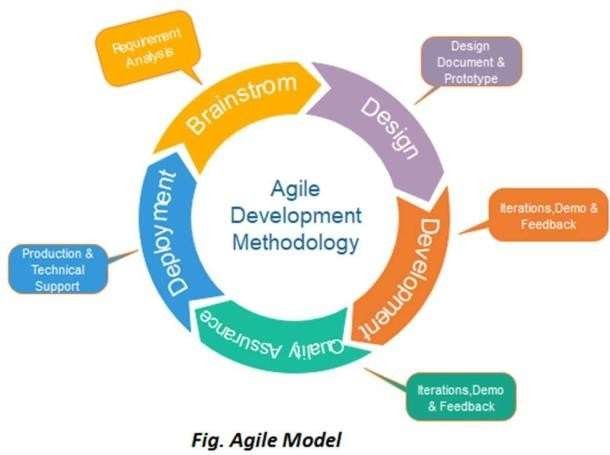


FIG.3.1: Agile Model

**PHASES IN AGILE MODEL**

1. **Requirements gathering:** In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.
2. **Design the requirements:** When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.
3. **Construction/ iteration:** When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.
4. **Testing:** In this phase, the Quality Assurance team examines the product's performance and looks for the bug.
5. **Deployment:** In this phase, the team issues a product for the user's work environment.
6. **Feedback:** After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

AGILE TESTING METHODS:

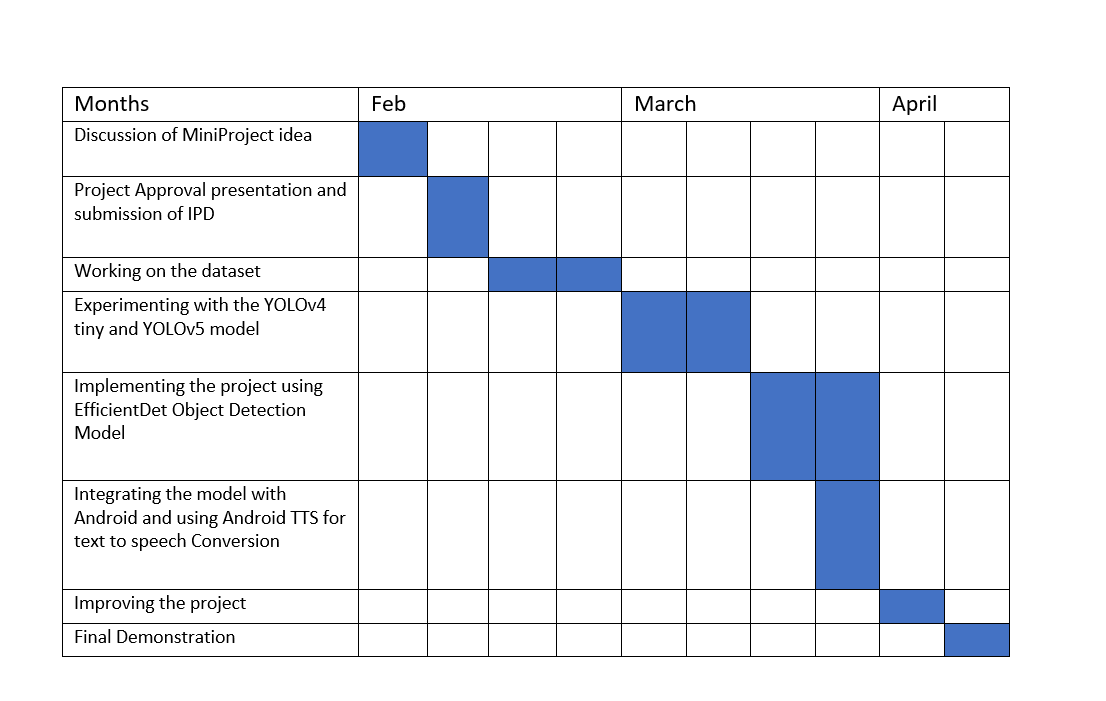
* + Scrum and Crystal
  + Feature Driven Development(FDD) and Lean Software Development
  + eXtreme Programming(XP) and Dynamic Software Development Method(DSDM)

ADVANTAGE(PROS) OF AGILE METHOD:

1. Frequent Delivery
2. Face-to-Face Communication with clients.
3. Efficient design and fulfils the business requirement.
4. Anytime changes are acceptable.
5. It reduces total development time.

DISADVANTAGES(CONS) OF AGILE MODEL:

1. Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
2. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.
   1. **TIMELINE CHART**

****

***Fig 3.2:Timeline Chart***



# CHAPTER 4 REQUIREMENT ANALYSIS

### REQUIREMENT ANALYSIS

##### 4.1 HARDWARE REQUIREMENTS

* + Processor: Intel i5 (8th Gen)
  + RAM: 4 GB or above
  + Hard Disk: 1 TB or more
  + GPU: NVIDIA GEFORCE

##### 4.2 SOFTWARE REQUIREMENTS

* + - Windows OS
    - Keras
    - TensorFlow
    - YOLO
    - Google Colab
    - Android Studio
    - API’s for speech

**4.3 FUNCTIONAL REQUIREMENT**

In software engineering, a functional requirement defines a function of software system or its component. A function is described as a set of inputs, the behaviour, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all the cases where the system uses functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements, which impose constraints on the design or implementation. As defined in requirements specify particular results of a system. Functional requirements drive the application architecture of a system.

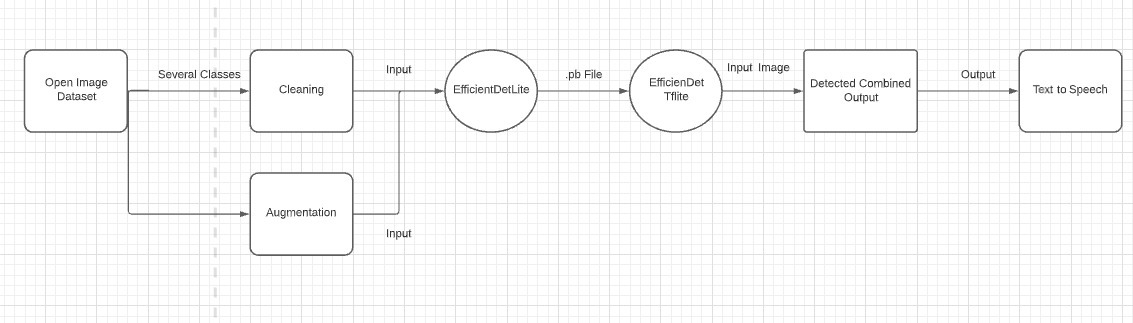
* Reporting Requirements
* Administrative functions and Authorization levels

**4.4 NON-FUNCTIONAL REQUIREMENTS**

In software engineering, a functional requirement defines a function of software system or its component. A function is described as a set of inputs, the behaviour, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all the cases where the system uses functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements, which impose constraints on the design or implementation. As defined in requirements specify particular results of a system. Functional requirements drive the application architecture of a system.

* Reliability requirement
* Maintainability requirement
* Availability requirement
* Scalability requirement
* Capacity requirement

**4.5 DATA FLOW DIAGRAM**



***FIG. 4.1 Level 2.0 DFD***

#### PROJECT METHODOLOGY

**4.6.1RoboFlow**

Roboflow is a Computer Vision developer framework for better data collection to preprocessing, and model training techniques. Roboflow has public datasets readily available to users and has access for users to upload their own custom data also. Roboflow accepts various annotation formats. In data pre-processing, there are steps involved such as image orientations, resizing, contrasting, and data augmentations.

Steps To Use Roboflow in Object Detection:

1.Dataset Loading

2.Labeling

3.Organise

4.Process

5.Train

6.Deploy

7.Display

Roboflow helps in every step of computer vision problem right from data collection to deployment. Roboflow enhances performance by its efficient parameters readily available to use.

**4.6.2 EfficientDet: Scalable and Efficient Object Detection**

Model efficiency has become increasingly important in computer vision. First, we propose a weighted bi-directional feature pyramid network (BiFPN), which allows easy and fast multiscale feature fusion; Second, we propose a compound scaling method that uniformly scales the resolution, depth, and width for all backbone, feature network, and box/class prediction networks at the same time. Based on these optimizations and better backbones, we have developed a new family of object detectors, called EfficientDet, which consistently achieve much better efficiency than prior art across a wide spectrum of resource constraints. In particular, with single model and single-scale, our EfficientDet-D7 achieves state-of-the-art 55.1 AP on COCO test-dev with 77M parameters and 410B FLOPs, being 4x - 9x smaller and using 13x - 42x fewer FLOPs than previous detectors.

Recently, the Google Brain team [published](https://arxiv.org/abs/1911.09070) their EfficientDet model for o[bject detection](https://blog.roboflow.com/the-ultimate-guide-to-object-detection/) with the goal of crystallizing architecture decisions into a scalable framework that can be easily applied to other use cases in object detection.

**4.6.4 TensorFlow Lite**

TensorFlow Lite is a mobile library for deploying models on mobile, microcontrollers and other edge devices.

TensorFlow Lite is a set of tools that enables on-device machine learning by helping developers run their models on mobile, embedded, and edge devices.

When one uses TensorFlow to implement and train a machine learning algorithm, one typically ends up with a model file that takes up a lot of storage space and needs a GPU to run inference. On most mobile devices, luxuries such as huge disk space and GPUs are not usable. TensorFlow Lite is a solution for running machine learning models on mobile devices.

The TensorFlow Lite is a special feature and mainly designed for embedded devices like mobile. This uses a custom memory allocator for execution latency and minimum load. It is also explaining the new file format supported Flat Buffers. TensorFlow Lite takes existing models and converts them into an optimized version within the sort of .tflite file.

**Advantages of TensorFlow Lite:**

Convert TensorFlow models to TensorFlow lite models quickly and easily for mobile-friendly models.

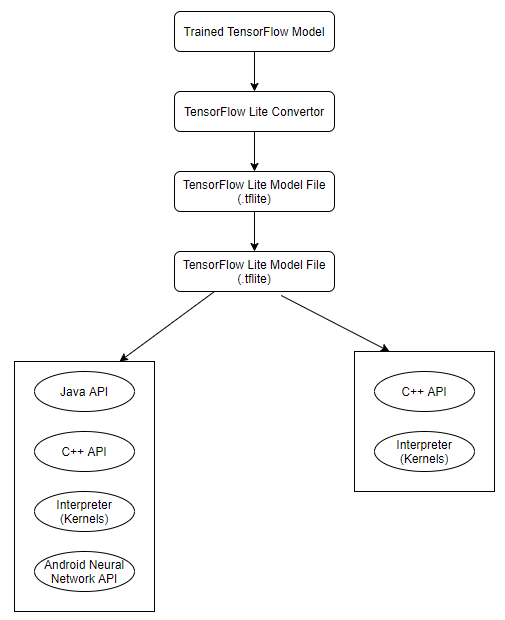
With simplicity, builds machine learning apps for iOS and Android devices.

In contrast to server-based architectures, a more effective alternative to mobile model enablement.

On mobile devices, it allows offline inference.

Tensorflow Lite allows one to execute machine learning models easily on a smartphone, allowing one to perform traditional machine learning tasks without the need for an external API or server. As a result, the models will operate on devices that are not connected to the internet.

* **The Architecture of TensorFlow Lite:**



***Fig 4.2 Architecture of TensorFlow Lite***

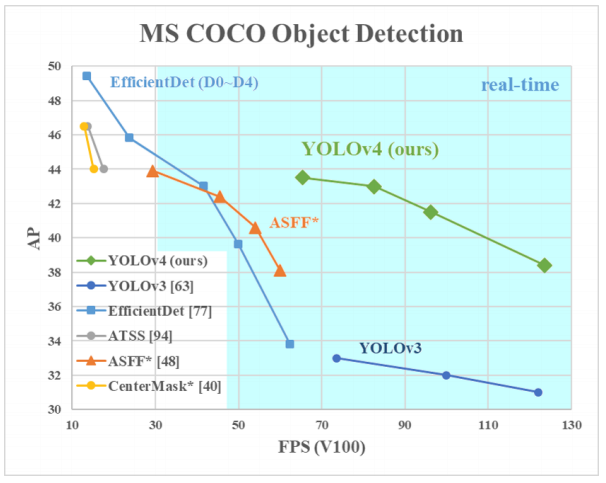
**4.6.5 YOLO v4 tiny**

YOLOv4-tiny is the compressed version of YOLOv4. It is proposed based on YOLOv4 to make the network structure simpler and reduce parameters so that it becomes feasible for developing on mobile and embedded devices.

We can use YOLOv4-tiny for faster training and faster detection. It has only two YOLO heads as opposed to three in YOLOv4 and it has been trained from 29 pre-trained convolutional layers as opposed to YOLOv4 which has been trained from 137 pre-trained convolutional layers.

The FPS (Frames Per Second) in YOLOv4-tiny is approximately eight times that of YOLOv4. However, the accuracy for YOLOv4-tiny is 2/3rds that of YOLOv4 when tested on the MS COCO dataset.

For real-time object detection, YOLOv4-tiny is the better option when compared with YOLOv4 as faster inference time is more important than precision or accuracy when working with a real-time object detection environment.



***Fig 4.3 Evaluation of all the models***

We can see that EfficientDet D4-D3 achieves better AP than YOLO v4 models, but they run at speed of < 30 FPS on a V100 GPU. On the other hand, YOLO is able to run at a much higher speed (> 60 FPS) with very good accuracy.

**4.6.6 Text-to-Speech**

Text-to-speech (TTS) is a type of assistive technology that reads digital text aloud. It’s sometimes called “read aloud” technology. TTS can take words on a computer or other digital device and convert them into audio.

The voice in TTS is computer-generated, and reading speed can usually be sped up or slowed down. Voice quality varies, but some voices sound human.

# CHAPTER 5 PROPOSED SYSTEM

## PROPOSED SYSTEM

* 1. **Data Processing**

**5.1.1 Open Image Dataset**

Open Images is a dataset of ~9 million URLs to images that have been annotated with labels spanning over 6000 categories.

The annotations are licensed by Google Inc. under [CC BY 4.0](https://creativecommons.org/licenses/by/4.0/) license. The contents of this repository are released under an [Apache 2](https://github.com/openimages/dataset/blob/main/LICENSE) license.

The images are listed as having a [CC BY 2.0](https://creativecommons.org/licenses/by/2.0/) license. **Note:** while we tried to identify images that are licensed under a Creative Commons Attribution license, we make no representations or warranties regarding the license status of each image and you should verify the license for each image yourself.

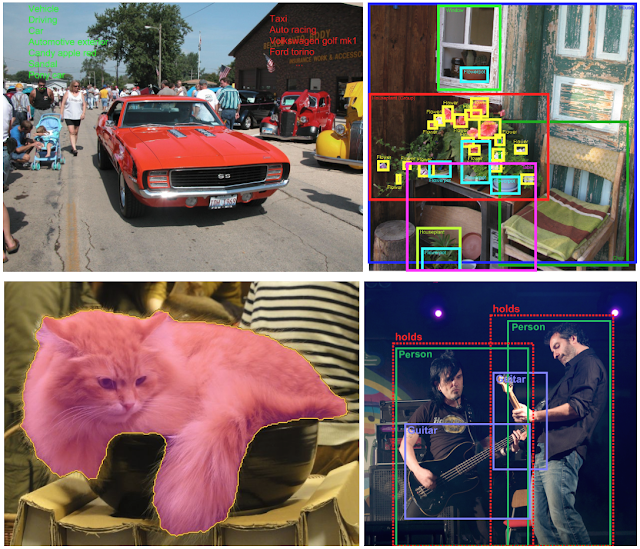
**5.1.2 Data Organization**

Each image has a unique 64-bit ID assigned. In the CSV files they appear as zero-padded hex integers, such as 000060e3121c7305. The dataset is split into a training set (9011219 images) and a validation set (167057 images). Each image has zero, one or more image-level labels assigned. Both sets have machine-populated annotations, while the validation set also has human annotations. The raters have been asked to validate the machine annotations, which allowed to practically eliminate false positive from the validation set (but not false negatives).

Labels are so called mids as can be found in [Freebase](https://en.wikipedia.org/wiki/Freebase) or [Google Knowledge Graph API](https://developers.google.com/knowledge-graph/). A short description of each label is available in [dict.csv](https://github.com/openimages/dataset/blob/main/dict.csv). There are 7844 distinct labels attached to at least one images, but only around 6000 labels are considered 'trainable' with at least 50 images in the validation set and at least 20 images in the training set.

Each annotation has a confidence number from 0.0 to 1.0 assigned. The human annotations are definite (either positive, 1.0 or negative, 0.0), while machine annotations have fractional confidences, generally, >= 0.5. The higher confidence, the smaller chance for the label to be a false positive.

Open Images V6 is a significant qualitative and quantitative step towards improving the unified annotations for image classification, object detection, visual relationship detection, and instance segmentation, and takes a novel approach in connecting vision and language with localized narratives. We hope that Open Images V6 will further stimulate progress towards genuine scene understanding.





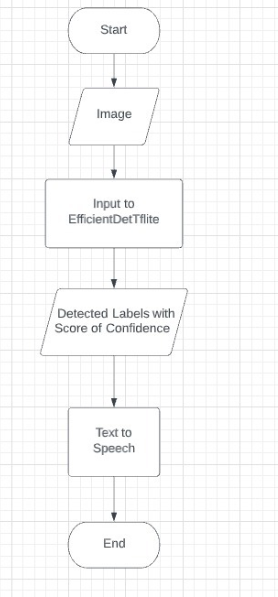
***Fig 5.1 Output from open image data set***

#### 5.3 SYSTEM DESIGN

Design is significant engineering illustration of whatever that's to be developed. Program design is a process design is the excellent option to effectively translate necessities in to a completed application product. Design creates a representation , presents element about software information structure, architecture, interfaces and add-ons which are vital to put into effect a procedure.

5.3.1 Flowchart of the Project

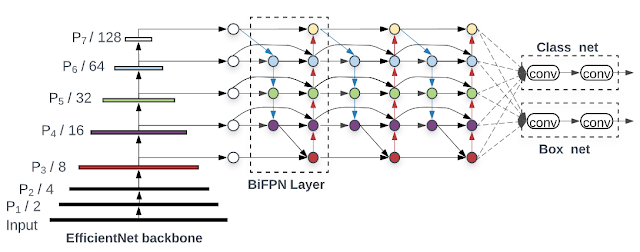
* An Open Image Dataset V6 was used as an dataset.
* The data is then passed through the RoboFlow Framework for annotation and Labeling.
* If the input data is an video is divided into frames and stored .
* The frames / images are passed to the EfficientDet model for output where object detection and object recognition is performed.
* The output is categorized into various indoor and outdoor images which a person sees in daily life.
* The Object Detection Model is deployed using TensorFlow Lite to use it on mobile devices.
* It undergoes android integration.
* Further, all the output received from the model is converted to speech by using Text to speech API.



***Fig 5.2 Flowchart of the project***

* + 1. **ARCHITECTURE/ FRAMEWORK**

**EfficientDet Architecture**



***Fig 5.3 EfficientDet Architecture***

# CHAPTER 6 IMPLEMENTATION

### IMPLEMENTATION

#### ALGORITHM

* + - Planning and designing the application based on the needs
    - Comparing different techniques available for object detection and classification
    - Exploring resources available for YOLOv4, YOLOv5 and EfficientDet Object Detection Model.
    - Converting TensorFlow model and weights for EfficientDet to TensorFlow lite format
    - Loading the EfficientDet model in memory in the device and feeding images to the model.
    - Using Android TTS the output from the EfficientDet will be converted to specch.

##### 6.2 EXPLANATION OF ALL THE IMPORTANT IMPLEMENTATION TECHNIQUES

**6.2.1 EfficientDet Model : Everything you need to know and how the EfficientDet works?**

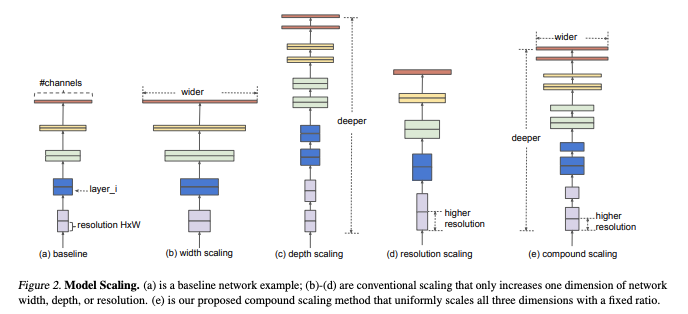
* **EfficientNet: Motivation and Design**

Recently, the Google Brain team released their own ConvNet model called EfficientNet. **EfficientNet forms the backbone of the EfficientDet** architecture, so we will cover its design before continuing to the contributions of EfficientDet. EfficientNet set out to study the scaling process of ConvNet architectures. There are many ways — it turns out 💭- that you can add more parameters to a ConvNet.

* **Challenges in Deep Learning EfficientDet Addresses**

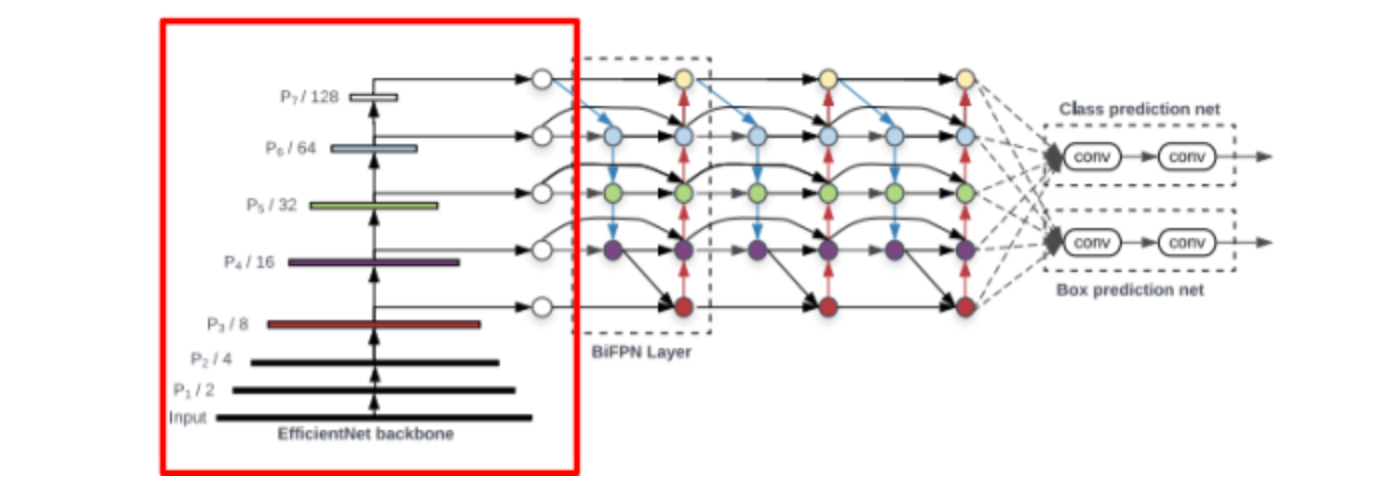
Before exploring the model, here are some key areas that have prevented image detection systems from being deployed to real life use cases.

1. **Data Collection** — With model architecture and pretrained checkpoints, EfficientDet cuts down on the amount of data required to generalize to a new domain.
2. **Model Design and Hyper Parameterization**— Once the data has been collected, machine learning engineers need to carefully set up the model design and tune a number of hyper parameters.
3. **Training Time** — the amount of time required to train the model on the gathered dataset. In the EfficientDet paper, this is measured in FLOPS (floating point operation per second).
4. **Memory Footprint** — Once the model is trained, how much memory is required to store the model weights when called upon for inference?
5. **Inference Time** — When the model is invoked, can it perform predictions quick enough to be used in a production setting?



***Fig 6.1 Model Scaling***

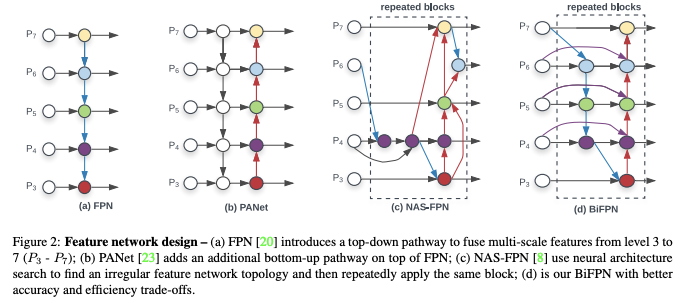
EfficientNet set out to define an automatic procedure for scaling ConvNet model architectures. The paper seeks to optimize downstream performance given free range over depth, width, and resolution while staying within the constraints of target memory and target FLOPs. They find that their scaling methodology improves the optimization of previous ConvNets as well as their EfficientNet architecture.



***Fig 6.2 Network Architecture of Efficient Det***

* **EfficientDet Feature Fusion**

Feature fusion seeks to combine representations of a given image at different resolutions. Typically, the fusion uses the last few feature layers from the ConvNet, but the exact ne



***Fig 6.3 Feature Network Design***

In the above image, FPN is a baseline way to fuse features with a top down flow. PA net allows the feature fusion to flow backwards and forwards from smaller to larger resolution. NAS-FPN is a feature fusion technique that was discovered through neural architecture search, and it certainly does not look like the first design one might think of. The EfficientDet paper uses “intuition” (and presumably many, many development sets) to edit the structure of NAS-FPN to settle on the BiFPN, a bidirectional feature pyramid network. The EfficientDet model stacks these BiFPN blocks on top of each other. The number of blocks varies in the model scaling procedure. Additionally, the authors hypothesize that certain features and feature channels might vary in the amount that they contribute to the end prediction, so they add a set of weights at the beginning of the channel that are learnable.

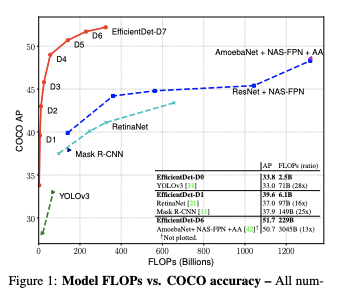
* **EfficientDet Model Scaling**

Previous work on model scaling for image detection generally scaled portions of the network independently. For example, ResNet scales only the size of the backbone network. But a joint scaling function had not yet been explored. This approach is very reminiscent of the joint scaling work done to create EfficientNet.

The authors set up a scaling problem to vary the size of the backbone network, the BiFPN network, the class/box network, and the input resolution. The backbone network scales up directly with the pretrained checkpoints of EfficientNet-B0 through EfficientNet-B6. The BiFPN networks width and depth are varied along with the number of BiFPN stacks.

* **EfficientDet Model Evaluation**

The EfficientDet Model is evaluated on the COCO (Common Objects in Context) data set, which contains roughly 170 image classes and annotations across 100,000 images. COCO is considered to be the general purpose challenge for object detection. If the model performs well in this general domain, it will likely do very well on more specific tasks. EfficientDet outperforms previous object detection models under a number of constraints. Below, we look at the performance of the model as a function of FLOPS.



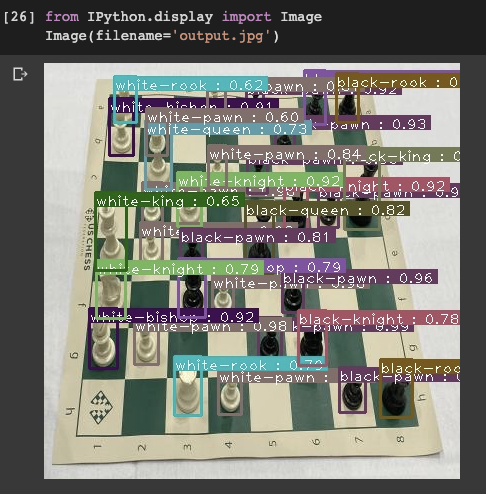
***Fig 6.4 Model FLOPS vs COCO accuracy***

* **Why is EfficientDet Useful?**

The pretrained checkpoints of EfficientNet crystallize all of the findings and automaticity that the researchers at Google Brain placed into building a ConvNet, along with all of the supervision that image classification on ImageNet can provide. The EfficientNet checkpoints are further leveraged with feature fusion and all components of the architecture are efficiently scaled. Finally, these model weights are pretrained on COCO, a generalized image detection dataset.

* **How Do I Use EfficientDet?**

At [Roboflow](https://roboflow.ai/), we have provided a tutorial on this blog post on [how to train EfficientDet](https://towardsdatascience.com/training-efficientdet-object-detection-model-with-a-custom-dataset-25fb0f190555) and this [Colab Notebook on how to train EfficientDet](https://colab.research.google.com/drive/1ZmbeTro4SqT7h_TfW63MLdqbrCUk_1br#scrollTo=KwDS9qqBbMQa). Through Roboflow, you can feed in your data set with annotations and simply feed a new data download link into our example and get some results. Then, after training, the notebook exports the trained weights for deployment to an application!



***Fig 6.5 Output from Efficient Det using Chess image***

**6.2.2 TensorFlow Lite**

1. Step 1: Load Input Data Specific to an On-device ML App. The flower dataset contains 3670 images belonging to 5 classes.
2. Step 2: Customize the TensorFlow Model. Create a custom image classifier model based on the loaded data.
3. Step 3: Evaluate the Customized Model.
4. Step 4: Export to TensorFlow Lite Model.

**6.2.3 Steps for Converting Text to Speech in Android**

Step 1: Create a New Project

Step 2: Working with activity\_main.xml file

Go to the app -> res -> layout -> activity\_main.xml section and set the layout for the app. In this file add an EditText to input the text from the user, a Button, so whenever the user clicks on the Button then it’s converted to speech and a TextView to display the GeeksforGeeks text.

Step 3: Working with MainActivity.java file

Step 4: Output: Run on Emulator

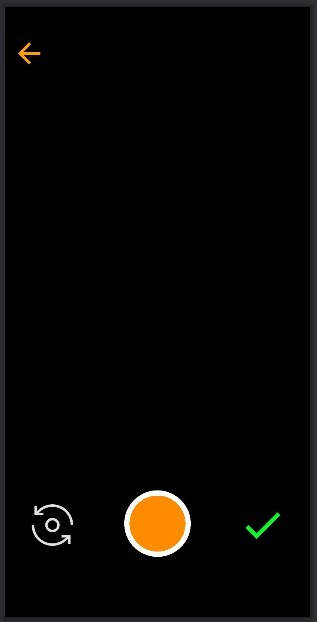
# CHAPTER 7 RESULT

# AND ANALYSIS

### RESULT

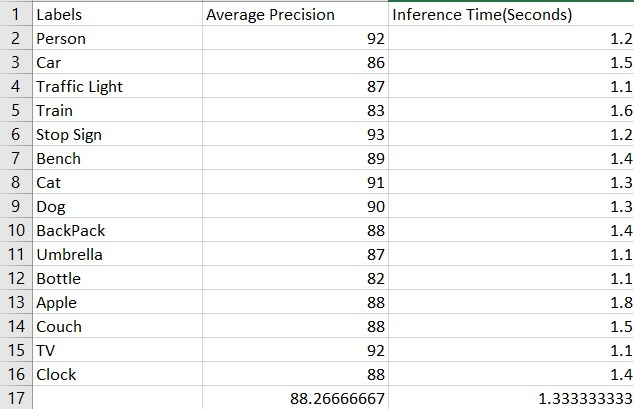
##### Output from the EfficientDet model





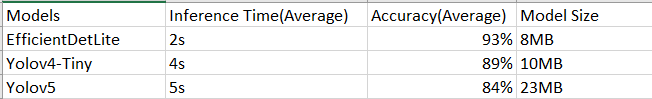
***Fig 7.1: Output Image 1***

##### Result Table Inference Result

****

***Table 1: Result table for EfficientDet Model***

* 1. **Models evaluation in consideration with android**



***Table 2: EfficientDet using Android***

# CHAPTER 8 CONCLUSION

**8: CONCLUSION**

#### 8.1 Conclusion

With the help of this project visually challenged people can detect the objects or obstacles in their path without asking for human assistance. Currency detection can also be performed which could help them to not to get fooled by performing money exchange with vendors or other people.

#### Future Enhancement

This project can be further enhances by describing the actions of the image input to the user. Also video frames can be used as an input instead of image input. Further text reading, facial recognition, braille ,color identification can also be done as the future enhancement in this project.



**REFERENCES**

* + - [1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014.
    - [2] R. Girshick, “Fast R-CNN,” 2015 IEEE International Conference on Computer Vision (ICCV), 2015.
    - [3] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, 2017.
    - [4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
    - [5] H. Caesar, J. Uijlings, and V. Ferrari, “COCO-Stuff: Thing and Stuff Classes in Context,” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.
    - [6] “TensorFlow Lite | TensorFlow,” TensorFlow. [Online]. Available: https://[www.tensorflow.org/lite.](http://www.tensorflow.org/lite) [Accessed: 24-Mar-2019].
    - [7] “EfficientDet | <https://towardsdatascience.com/a-thorough-breakdown-of-efficientdet-for-object-detection-dc6a15788b73>

**ACKNOWLEDGEMENT**

It is pleasant task to express gratitude to all those who contributed in many ways to this project and made it an unforgettable experience for us. First of all, we would like to thank our guide Prof. Deepti Vijay Chandran. This work would not have been possible without her guidance, support and encouragement.

We are highly indebted to Dr. Sunil Chavan, Principal, **Smt. Indira Gandhi College of Engineering**, Dr. KT. Patil Sir, Head of the Department, **Computer Department** and also all the faculty members.

We are thankful to lord for keeping us energized that every end seems to be beginning.