Islamic University of Technology (IUT)

Department of Computer Science and Engineering (CSE)

An Indoor Object Dataset for Mobile-based Detection and Recognition Systems for the Visually Impaired

Authors

Shehreen Azad, 170041005

Abdullah Abu Sayed, 170041004

Noshin Faiyrooz, 170041018

Supervisor

Dr. Md Kamrul Hasan

Professor, Department of CSE, IUT

Systems and Software Lab (SSL)

A thesis submitted to the Department of CSE

in partial fulfillment of the requirements for the degree of B.Sc.

Engineering in CSE

Academic Year: 2020-2021

April, 2022

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by Shehreen Azad, Abdullah Abu Sayed and Noshin Faiyrooz under the supervision of Dr. Md Kamrul Hasan, Professor, Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Gazipur, Dhaka, Bangladesh. It is also declared that neither this thesis nor any part of it has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others have been acknowledged in the text and a list of references is given.

Authors:

Author 1
Student ID: 170041005

Author 2
Student ID: 170041004

Author 3
Student ID: 170041018

Supervisor:

Dr. Md. Kamrul Hasan
Professor
Department of Computer Science and Engineering
Islamic University of Technology

Acknowledgement

First and foremost, we would like to praise Allah SWT for giving us strength and His blessings for successful completion of this thesis. We would like to express our gratitude towards our supervisor, Dr. Md Kamrul Hasan, Professor, Department of Computer Science and Engineering (CSE), IUT, and Dr. Hasan Mahmud, Assistant Professor, Department of Computer Science and Engineering (CSE), IUT, for being our mentors. Their motivation, suggestion and insightful knowledge for this research have been invaluable. Without their proper guidance, support and time, this research would never have been possible. We are indebted to them for guiding us at every stage of our journey. Finally, we would like to express our heartiest appreciation towards our family members for their continuous support and motivation, without which we could not have achieved the scale of implementation that we have achieved.

Abstract

Indoor object detection is a challenging area of computer vision where comparatively lesser work has been done compared to its outdoor counterpart. Surely, such a task requires huge amount of training data to make any classifier detect indoor objects with high precision. This indoor object detection can become way more challenging when it has to be specifically tailored for visually impaired people's mobility and interaction with everyday use objects. This report presents a novel indoor object dataset containing 5196 unique images of 8 everyday use indoor object category relevant to daily interaction of visually impaired people. The uniqueness of this dataset compared to existing indoor objects dataset is this dataset deals with everyday use objects and presents them with more contextual information than that is available in existing literature. Moreover, the varying lighting condition, non-canonical viewpoints, occlusion and complex background makes the dataset more robust while being trained on any object detection algorithm. Instead of going for higher accuracy we aim to find a trade-off between accuracy and speed as if this dataset is to be used to build a system for visually impaired people's navigation needs, that system has to be deployed on mobile or sensor-based hand-held device which requires light-weight models. Hence our proposed dataset is tested on two light-weight model, namely, SSD MobileNet V2 FPNLite and EfficientDet D0. It has achieved a mean average precision (mAP) of 29.5 and 39.4 respectively on both the models which is better than the original mAP values achieved by these models. Our proposed dataset can be extended with other indoor object detection dataset, as well as it can be used to build a system for visually impaired people's navigation.

Keywords— indoor object dataset, object detection, light-weight model, visually impaired, mobile-based system

Contents

List of Tables

Chapter 1

Introduction

1.1 Overview

Visual impairment is deemed as a prevailing issue in the affairs of the world. It is said that the first step to overcoming obstacles and hurdles is to become aware of those. Therefore, accurate information regarding the surroundings of visually impaired people is inevitable for their smooth movement around places. According to WHO, a mean of 285 million people globally suffer from visual impairments of which 39 million are blind and 217 million are suffering from partial eye problems[1], and of course, the number is increasing with time. This global phenomenon of visual impairment and the need to help them has led many researchers to dive into ways to reduce the sufferings of these blind people, that is, to mitigate the visually impaired people's troubles while they maneuver themselves in different environments. Many researchers have made their contributions in proposing and implementing navigation and object identification systems for the visually impaired. Object detection is a relatively new method for creating systems focusing on this particular user group with visual impairments. With the advancement of technology, it is now possible to provide a visually impaired user with different services that were not possible before. Deep learning has created a huge opportunity for creating and implementing systems that could potentially change the nature of assistive technology for the visually impaired user group. It is progressing with the advent of time and various systems are being built which provide this particular user group with a way of perceiving their surroundings in a way that they could not do before.

1.2 Motivation and Scopes

Our motivation is to provide assistance to visually impaired people and help them to become independent people just like any other normal person. Visually impaired people often have to depend on other people for their day-to-day activities. The simple task of identifying an object without touching it is a critical one for a person who cannot see. Navigation is another important factor for a visually impaired person. One who cannot watch their surroundings properly cannot move around in an efficient manner as one would not be able to understand if they are about to walk towards the right path or stumble upon some object on their way. We want to address these issues that blind person faces in their day-to-day lives. Our goal is to provide them with the feeling of being independent and performing their day-to-day life chores effortlessly by proposing a robust indoor object dataset that can be used for training an object detection deep learning model. Deep learning has advanced quite a lot since it first appeared and changed the image classification and object detection sector for good. It provides us with the ability to use different methods to build systems for detecting objects which could greatly help visually impaired people to be independent on their own and build their confidence. Various object detection architectures are now available and these provide state-of-the-art detection of objects of the various classes. Creating a system by attaching a voice command with the object detection system can easily provide the visually impaired user with an ability to understand what is around their surrounding environment. Nowadays there are various object detection architectures that provide the capability to run a computation on embedded systems and mobile devices. This is a huge opportunity for creating small systems that can be head-mounted or carried around in hand by the visually impaired user group.

This research mainly focuses on creating a dataset primarily harboring objects that visually impaired people interact with indoors, and training object detection models with our custom dataset. The trained models could be used in the future to build real-time object detection systems in mobile devices or embedded systems.

1.3 Research Challenges

Creating a robust dataset is quite a challenging task when the data volume is more. Datasets are key ingredients in creating an object detection system or working with an object detection architecture. In our case, we propose a dataset that consists of indoor objects. The collection of a huge number of images by web scraping is quite tedious and time-consuming. Image data requires labeling which is another critical task for this research. We collected 8 category objects and a total of 5196 images by web scraping which took approximately 60 working hours. After augmentation, this number reached 21708. Annotating this many images required us to outsource the image labeling process to an extensive team of 10 members. All of them working together took us around 50 working hours to complete the data annotation task successfully.

Another challenging aspect of this work was finding a suitable object detection model. As we would want our proposed model to work in real-time on a mobile device, our model had to be fast without having a great loss of accuracy. An extensive study had to be done on the speed and accuracy of different models to come to a conclusion about the architecture that could be best suited for real-time application.

1.4 Research Contribution

The prime contributions of our research is as follows :

- A robust dataset containing everyday use indoor objects that are relevant to a visually impaired user's secure mobility and require fast retrieval
- Light-weight object detection models that have been trained on our custom dataset

Our trained models can be used in the future for creating indoor object detection systems for the visually impaired user group.

1.5 Report Layout

This report is arranged in the order given here: In Section 2, we have described the related works we have studied that are divided into a system-based and dataset-based approach, before going ahead with our own method. In the following section, section 3, we have discussed our proposed approach in terms of dataset collection, augmentation and model selection. In section 4, we have discussed our entire dataset construction method- method of data collection, annotation, pre-processing in an exhaustive manner. We also discussed the uniqueness and strength of our dataset in this section. Section 5 discusses the experimental setup of our work, the selected models and why they were selected, as well as the entire training process. Different evaluation metric and how our dataset performed on those are discussed in Section 6. Analysis of the evaluation metrics and their explanation is also included in this discussion. Finally, in section 7 we conclude our report by listing out some future directions for our work.

Chapter 2

Literature Review

Numerous researchers have created, studied and worked on many datasets related to object detection and recognition. However, only a few of those datasets were dedicated to the object detection and recognition systems targeted towards the visually impared people(VIP). We have studied existing systems that are specifically for visually impaired person's navigation to understand the research gap in existing systems. From there we have realized none of those systems actually incorporate datasets specified for indoor environment with interaction with everyday use objects. To understand more about the datasets we have studied indoor object and scene datasets of existing literature. From there, we have found the area where our contribution would be meaningful. In this section we have divided the related works that we have studied into two sections, namely - system based approach and dataset based approach.

2.1 Systems

2.1.1 Using Pre-trained Model

Arora, Adwitiya, et al. [2] proposed a system prototype with a combination of SSD [3] and MobileNetv2 [4] for object detection for the visually impaired people. The system prototype is integrated in a wearable head mounted device to detect objects in an outdoor setting. The model they have used is pre-trained on the MS COCO dataset and then fine-tuned using the PASCAL VOC0712 [5]. The prototype consisted of a cameran, a cap (to hold the camera), earphones for audio feedback of the detected object and Raspberry pi 3 for performing the object detections. Since their proposed system is built on a pre-trained model, it performs significantly well in terms of detecting objects that have a high detection accuracy in the original MS COCO dataset.

Noman, M. et al.[6] proposed an indoor assistive system for the visually impaired people to use when they move around in their surroundings, but not specifically indoors. The proposed system is portable and able to work offline. It can accurately detect objects and how far are they from the user. Three major parts that make up the proposed system are -

- **Single-Board Computer (SBC)** : reads the input, produces and provides the output to the user.
- **Object detection** : consists of a CCD camera and the object recognition module.
- **Time-of-Flight (ToF) sensor** : measures the distance to/from the object and the object's height.

The proposed system takes real-time data through the CCD camera and processes it to give output scores to the user. This score is based on the accuracy of the underlying object detection algorithm. The system ignores scores lesser than 70%. TensorFlow object detection API [7] is used for the object detection module to work in the back-end for accurate detection. They have used the SSD+MobileNetV1 [8] model pretrained with the MS COCO dataset for the object detection module. The results of the proposed system were compared with that of Microsoft Azure Cloud [9] which is a cloud based platform and Microsoft Kinect [10] which is a hardware device, both of which are successfully used in various computer vision tasks. The proposed system has performed well while being compared against Microsoft Azure Cloud (3.34% average difference of detection scores). The authors claim that this difference can be decreased by using more training and testing data.

Kayukawa, Seita, et al. [11] proposed a smartphone-based system that aids visually impaired users to navigate seamlessly while standing in a line by detecting humans already present in a line and providing tactile (vibration) feedback to the user in order to indicate the user to move forward or stop. The RGB camera of the Iphone 11 Pro is used for taking image inputs and the infrared depth sensor is used for calculating the distances. The humans are detected using the YOLOv3 tiny [12] model which is pre-trained on the ImageNet [13] dataset. Their system is evaluated on 6 blind people of 22-23 age group for its effectiveness and efficiency. The system was effective and well accepted to the test users. They were able to successfully stop (94.5% success rate) behind the target persons and they were unequivocal about the fact that the system allowed users to start/stop moving forward at the right timing.

2.1.2 Using Own Dataset

Rahman, A, et. al. [14] developed a system that enhances better mobility of the visually impaired people by detecting staircase, chair, table, person, washroom and currency. They have collected their own data using a Pi camera module v2 attached with their prototype. They have made two datasets for their proposed system, one with common objects (staircase, chair, table, person, washroom) and another with currency. After augmentation the size of the common object's dataset is 4500 and size of the currency dataset is 800. Their proposed system identifies the indoor and outdoor objects present in their dataset, notifies the user through audio message and sends data to the cloud repeatedly after a certain time. The 5 modules of the developed system are as follows:

- **Obstacle Detection -** uses 4 sensors (3 for objects in left, right and front direction and 1 for ground or knee level objects)
- **Obstacle Recognition -** recognizes common objects like chair, table, person, staircase, washroom and currency. The Pi camera module v2 is used to take the photos and reshape the pictures into frames of 720x680 pixels. MobileNet [8] along with TensorFlow Lite object detection API [15] is used for detection of the objects.
- **Fall Detection** - uses an accelerometer to detect free fall and alerts the guardian of the user if the user goes through a free fall.
- **Location Finding** – a GPS is used to locate the user using information like the city, state and country names of the current location
- **IoT-enabled automated system -** all the aforementioned modules are connected to the internet. The system sends all the data from the sensors to a remote server for processing and analysis of that data in real-time.

The performance of the system was measured according to how accurately it could tell the distance of an object from the user and how precisely it recognised the object over those distances. The results of the latter for indoor environment is over 99% with error rate $\leq 1\%$ and for outdoor environment is over 98% with an error rate of at most 1.5%. They produced confusion matrices for several classes of objects to calculate the accuracy of object detection which turned out to be 96% or above for common objects and 98% or above for currency notes recognition.

2.1.3 Others

The Intelligent Eye[16] is a mobile based application for the android platform created for the visually impaired people(VIP) in order to aid them with detecting light and color and recognising objects and banknotes. The application worked on the user's mobile device without requiring an internet connection. Light and color is detected by light sensor and the mobile's camera respectively. Objects recognition is done using the CraftAR image recognition SDK [17]. Using this SDK a annotated dataset is embedded locally in the mobile phone. This SDK also enables image recognition on the fly using neural networks. The result of each recognition was read out loud to the user. The results were based on user acceptability, accuracy on object recognition and banknote recognition. For the user acceptability testing, 10 users with visual impairments were chosen to use the application and answer a questionnaire. Each question was answered with a score between 1 and 5 where 1 meant "poor performance while 5 meant "excellent performance. All the results turned out to be good in general and the application was well accepted by those users.

2.2 Datasets

2.2.1 Large Scale Dataset

Since the advent of the MS COCO[18] dataset, a lot of experiments and research work based on object detection and recognition have used this dataset to train and test their models. Detection of objects greatly depends on the contextual information and so MS COCO has images where the objects are placed in their natural environments(non-iconic).Their work focused mainly on solving 3 fundamental research problems in scene understanding: a. Detection of non-iconic views b. Contextual reasoning between objects c. Precise 2D-localisation of objects.The dataset harbors 328k images of 91 object types with a total of 2.5 million instance detections.

The COCO dataset has 80 object categories called the "COCO classes" under which falls "things" for which the objects can be easily identified and annotated (e.g. dog, cat, bottle etc.) and 91 "stuff" under which falls boundary-less objects, that provide significant contextual information(e.g. sky, street, grass etc.). This resulted in a collection of 328,000 images in which objects have contextual relationships between themselves.

Apart from MS COCO, we have come across ImageNet[13] and PASCAL VOC[5]. Despite covering a huge range of objects, none of these focused solely on any specialized target like the visually impaired people. The object classes are more generalized and thus not suitable for specialized application for efficient maneuvering for the visually impaired people in indoor environments. The “Scene UNderstanding(SUN)” dataset proposed by Xiao, Jianxiong, et al.[19] harbors 130,519 images targeted for scene understanding which fall under 899 scene categories. This dataset although has a varied collection of indoor scenes, however, this dataset can be used for recognition of "stuff" rather than "things".

2.2.2 Indoor Object and Scene Dataset

Quattoni et al.[20] created a dataset that specifically tackles problems related to indoor scene recognition having a wide variety of 67 categories of indoor scenes, which are clustered into 5 big scene groups, namely store, home, public space, leisure, working space. The dataset contains 15620 unique images. They have tested their proposed dataset on 5 multi-class classifier models popular at that time, namely Gist SVM, ROI Segmentation, ROI Annotation, ROI+Gist Segmentation and ROI+Gist Annotation. Global regularity was observed in case of greenhouse, computer-room, inside-bus, corridor and pool-inside, which were in the top half best performing categories. However, among this top half other four categories - buffet, bathroom, concert hall, kitchen was also present for which no global regularity was observed. Their results are almost close to that of first attempts on Caltech 101 [21].

Bashiri et al.[22] developed a novel indoor object detection dataset MCIndoor20000 using 20,000 indoor images and contains 3 classes, namely - door, hospital signs, stairs. Although their dataset size is 20k, the dataset has only 2055 unique images, rest of which are augmented versions of those 2055 images. To make the dataset more comprehensive, they have incorporated several object model variations including viewpoint variation, intra-class variation, rotation, noisy conditions (e.g., Gaussian, Poisson), and occlusion. Their work aims to help visually impaired people’s mobility in unfamiliar environments. They have tested their proposed dataset on AlexNet [23] which is pretrained on the large scale object recognition dataset ImageNet [13]. They have achieved 99.8% accuracy while using a 80:20 split on their dataset for train and test set, which is higher than that of ALexNet’s original achieved accuracy.

Adhikari, Bishwo, et al. [24] constructed a dataset named TUT indoor dataset which consists of 2213 images containing 7 object categories, namely - fire extinguisher, chair, exit sign, clock, printer, trashbin, screen. There are in total 4595 object instances in the dataset. However, their dataset distribution is quite skewed as both fire extinguisher and chair has over 1600 instances, whereas the exit sign has around 500 instances. Clock and trashbin has around 300 instances and screen, printer has approximately 100 instances. The number of images in the dataset are increased 3-4 times through different data augmentation techniques. They have originally proposed a method for semi-automatic data annotation process which they have applied in their proposed TUT dataset and got satisfactory results on Faster R-CNN [25], SSD [3] and RetinaNet [26].

Afif, Mouna, et al.[27] constructed a dataset called IODR specifically for the visually impaired. They proposed [28] a novel DCNN design based on pre-trained YOLO v3 [12]. They used the IODR dataset to train and test their architecture. The dataset harbors 8000 indoor images with 16 categories' of landmark indoor objects, namely - table, chair, smoke detector, fire extinguisher, security button, trash can, door, electricity box, heating, stairs, notice table, signs, window, light switch, light and elevator. The images of these objects were taken under varied lighting conditions, poses, angles and partial or no occlusions.These pictures were high in inter and intra class variation. The images were labeled using Labellmg software. Of the dataset, 66% was used as training data and the rest as testing data. YOLO v3 [12] was used here for object detection which in turn used the "Darknet-53 [12]" CNN architecture to extract the features from the images. They used the transfer DCNN learning technique as it uses less data. The results showed that their proposed dataset and system gave accuracies of 85.55% for the class "door", 99.18% for "chair" and 99.35% for "table" in comparison with the work of Ding, Xintao, et al.[29].

The details of the aforementioned indoor objects datasets present in current literature is shown in Table 1.

Table 1: Description of different existing indoor datasets

Dataset	**Dataset Size**	**No. of Class Labels**	**Average no. of Images per Category**	**Type of Objects**	**Average no. of Objects per Image**
MCIndoor20000 [22]	20k (2055 unique images)	3	685	Landmarks	1
IODR [27]	8k	16	500	Emergency situation	1.5
TUT indoor dataset [24]	2213	7	316	Office	2.03

2.3 Limitations of Existing Systems and Datasets

The main lacking in these system based approaches are most of these [6, 2, 11] are based on models pre-trained with other large scale dataset and hence those systems can only detect objects which are present in those specific datasets. Rahman, A, et. al. [14] even though created their own two datasets for specific object recognition (bank notes), the dataset size is way too small to be valuable for robust detection in generalized scenario. Moreover, their common object dataset holds objects that are already present in other large scale datasets (chair, table, person). On the other hand, there is other large scale dataset specified for scene recognition (washroom). So, the need for creating another smaller dataset with same object categories was irrelevant. Using such small scale dataset the authors did even though generate promising results for their system prototype, in the bigger picture this small dataset would fail to generalize and thus would not be of much value. Hence from the system based approach in the available literature we can conclude that there is a gap in systems specifically tailored for everyday use indoor objects' detection which might aid the mobility of visually impaired people and thus contributing to such a system is necessary.

On the other hand, the objects that were considered in MS COCO are very generalized, a mixture of outdoor and indoor objects that may or may not be used in our daily lives(e.g. surf boards, snowboard, frisbee etc).They fail in indoor object detection. The object classes of other large-scale datasets [13, 5, 19] are more generalized and thus not suitable for specialized application for efficient maneuvering for the visually impaired people in indoor environments. In case of a visually impaired person's mobility it's more important and crucial to accurately detect "things" as those are what that might pose as more dangerous to people having eyesight disability. The already available indoor objects and scene datasets still do not solve the mobility issue in indoor environments of the visually impaired user where they have to interact with everyday use objects. Moreover, all of the indoor datasets in literature are tested on different highly accurate but heavy-weight models. The accuracy of these datasets on a light-weight model is still unknown and if a visually impaired person has to use a system trained on these datasets, that system has to be deployed in hand-held devices which would require much lighter weight models. Here comes our proposed dataset which would incorporate everyday use objects and it is tested on lighter models so that as a future work of ours a light weight model trained with our dataset can be deployed in a system in handheld sensor or mobile-based device. If a system is built specifically for visually impaired user's mobility in indoor environment with model that is trained on our dataset, the gap that is currently present in the research can be bridged.

Chapter 3

Proposed Approach

We will be preparing our own dataset for this work. The image data are collected by web scraping. Later they will be augmented following standard augmentation techniques. Finally, the augmented data will be used to train different deep learning models in order to facilitate the creation of different systems to help benefit the visually impaired user in their day-to-day life. This section covers the overview of the design and the logical analysis behind the steps taken in our work.

3.1 Data Collection

This work revolves around preparing an ideal dataset that is specifically concerned with indoor objects. The purpose of collecting such a dataset is to provide assistance to visually impaired users in their day-to-day life. From the literature review, we observed that there is no such work that specifies indoor objects detection in depth. Most of the works are based on different objects altogether without taking into account the context of the user and their environment. Large scale datasets are available [27] for indoor objects, but they are considered only in emergency situations and took into account the objects that are necessary for user navigation only in case of an emergency. Furthermore, there are large datasets available for user navigation indoors [22] but their work is also bounded by only three objects which are stairs, doors, and hospital signs. It is not a generalized approach that could be used for visually impaired people all around the world but a more specific work for only a particular user group in one particular environment.

This work focuses on creating a generalized dataset that contains objects that are available indoors on a day-to-day basis. These are common objects that one uses in their home or any indoor environment. The main goal of such a dataset is to provide an ability to the visually impaired user such that they can identify objects in their surroundings without any help from any other entity. Not only in case of any emergency situation or only in particular environments, but a generalized environment considering the spatial context of the user of all circles.

3.2 Data Augmentation

After the collection of our data by web scraping, the images were processed by augmenting them. Augmentation is a great way to generate more data from a small collection of data. Popular image datasets such as MS COCO and Pascal VOC have tens of thousands and more images available in their dataset. A large dataset has a great impact on training a deep learning model. The performance of a model largely depends on the size of the dataset it is given.

The process of training such a neural network is by tuning the parameters such that the model is able to map the inputs to a specific and accurate class label. Parameter tuning is done so that loss of the model is minimized which would eventually lead to better accuracy afterward. Now a neural network would learn these parameters from the input image that is given to it. Thus for a model which has been trained on a larger dataset, its parameters would be better tuned to avoid overfitting for that model. Such a model would perform better in the case of an unknown test image. So it is of significance that neural networks are trained on larger datasets to improve their performance. But the availability of large datasets is very rare. Thus to scale up the dataset, augmentation has a huge impact as it helps to increase the size of the dataset by performing different actions like scaling, rotation, affine translation, gaussian noise, etc. The newly formed images after this action would seem like the same image to us but to a neural network, a slightly rotated image is also termed a new image. The augmentation process is discussed in more detail in section 4.4.

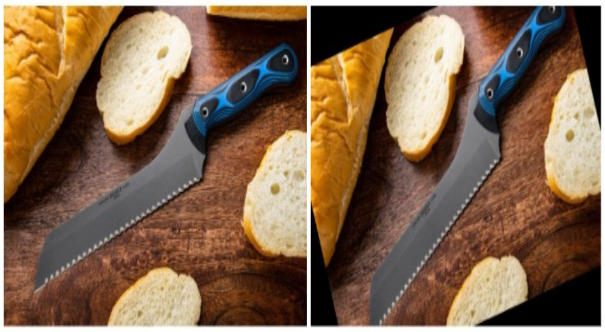


Figure 1: Original image (left) and slightly rotated image (right)

For the human eye, the two images of Figure 1 might look the same, but for a neural network, these are two totally different images and the model would learn the parameters differently for these two images.

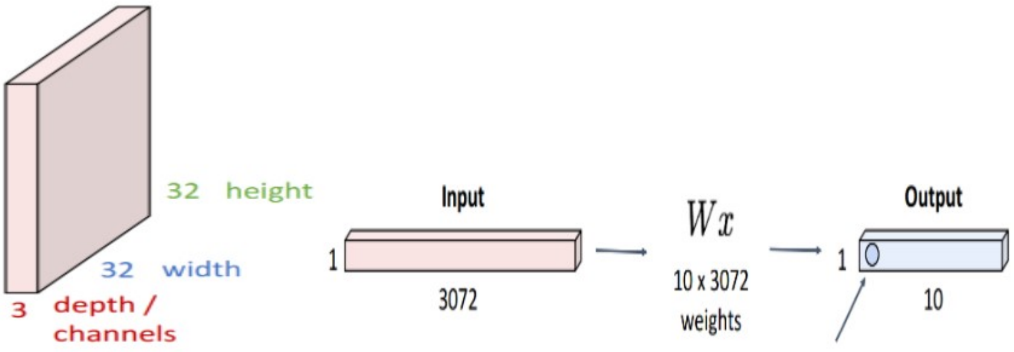
Another improvement of augmenting data is that it helps a model to be invariant. Invariance is a property of a neural network that implies a network is able to accurately give output class labels even if the objects are placed in different orientations. A model can have size, translation, viewpoint, and illumination invariance. By synthetically modifying our data, we increase its ability to learn more and give better results.

3.3 Model selection

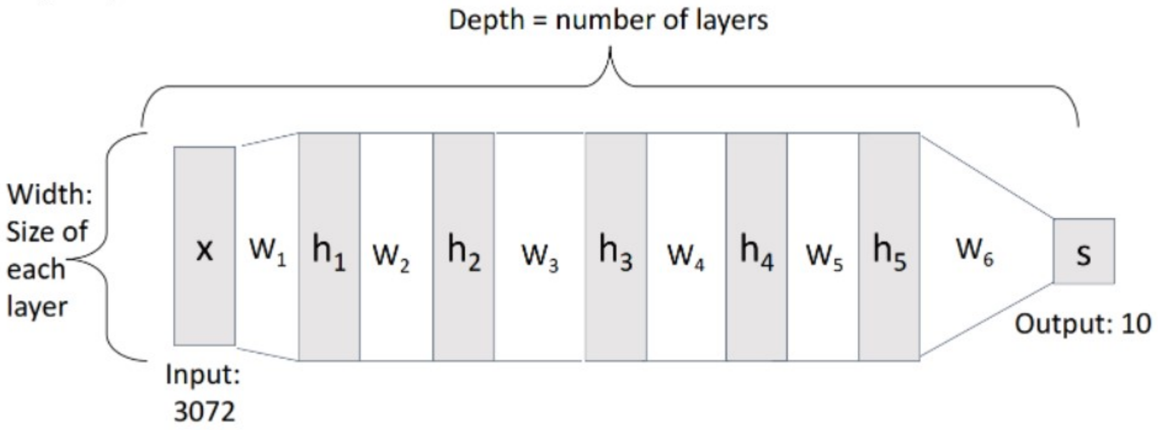
Our purpose of collecting images and augmenting them was so that we would use those images and train a deep learning model allowing it to give accuracy for those class labels that we defined. The purpose of this process is to allow a machine to be able to work as a human by perceiving the world as a human eye would and performing various tasks like object detection and recognition.

3.3.1 Neural Network

A neural network is part of machine learning and lies at the core of deep learning algorithms. An artificial neural network comprises multiple nodes. There are the input layer, the hidden layer, and the output layer. The nodes are interconnected and carry weight across the model. The weights are passed onto the next layer based on the activation threshold value. Based on these learned weights, the model calculates a loss value which determines the efficiency of the network. Higher loss means that the model is more error-prone and a lower loss value indicates that the model is performing better. A neural network learns from the given training data and with more training, it becomes better in terms of accuracy. But in the case of image data, these types of models discard the spatial structure of the image by flattening the image pixel data into a single vector. There is an input layer, weights, and output layer in a sequential manner and the whole network comprises such repeating layers. This entire description can be better visualized by Figure 2a and 2b where it is assumed that the input is of size 32x32 and the number of classes is 10.



(a) Neural network discarding the spatial feature of an image



(b) A neural network with 6 layers, where X = input, W = weights, h = hidden layer and, S = output

Figure 2: Layers of a neural network

The output is calculated using ReLU on with a linear score function. The max function acts as the ReLu operation on the linear classifier. The ReLu or Rectified Linear unit helps in deciding whether to pass the value of a node to the next layer or not and thus has a critical impact on the way in which a model learns the parameters. The max function which returns either a 0 or the value of the node is the ReLu operation. This passes either 0 or the node value onto the next node where the values are multiplied by the next node value. Feature value calculation for a two-layer neural network is thus done by calculating $f = W_2 max(0, W_1 x)$.

The main aspect of the neural network is that each node of the input layer will be connected to every other node of the output layer. The weight values in between those two layers will be initialized at the beginning of training and at later stages, those weight values will be updated according to the loss calculated in the output. Figure **??** shows a basic represenation of a fully connected neural network.

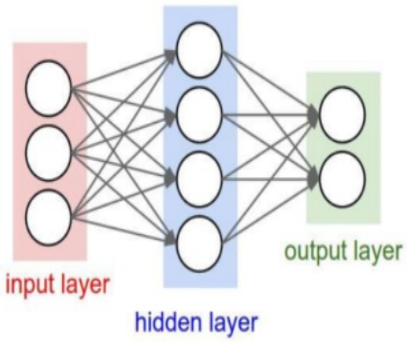


Figure 3: Fully connected neural network representation

3.3.2 Convolutional Neural Network

A convolutional neural network (CNN) is a deep learning algorithm that takes images as input and extracts values from them by assigning learnable weights and biases to those image pixels in such a way that the extracted values can provide a way of identifying a category of the image later on. A CNN obtains the spatial and temporal features of an image successfully, (shown in Figure 4), in contrast to a feed-forward network where the spatial feature of an image is discarded in the very beginning. These types of networks have a lower number of learnable parameters involved. It reuses the weight values throughout the network using backpropagation to traverse the weight back and forth over the whole network. This type of network has three main components. These are the convolutional layer, pooling layer, and normalization.

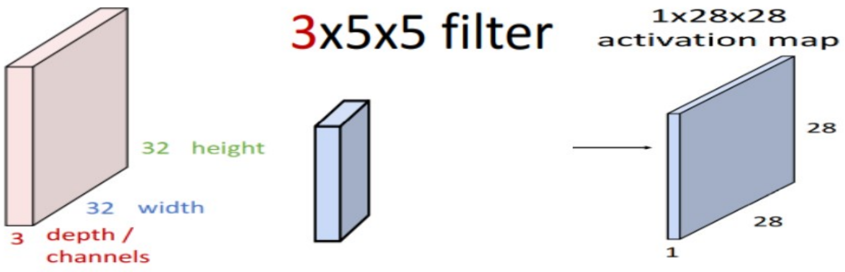


Figure 4: Convolutional layer preserves the spatial feature of image

Multiple convolutional filters can be stacked upon a single input image to obtain the output. The key point in CNN is that the pixels have an impact based on their position. The pixels now hold spatial data for the image and they are preserved in a CNN. After the convolutional layers are multiplied with the input, there is a ReLU just like that present in a neural network. After the ReLU function, padding is introduced as each convolutional layer reduces the dimension of the input image to some extent. So in order to retain the size of the image from shrinking too much, zero padding is done in most cases. After the convolutional layers, the image is downsampled, (shown in Figure 5), using a pooling layer. Most deep learning algorithms use a max-pooling operation where the max value from a block of pixels is chosen and they form a new downsampled image with lower resolutions. After the pooling operation, normalization is done and it helps to better optimize the model and backdrop through the different layers efficiently.

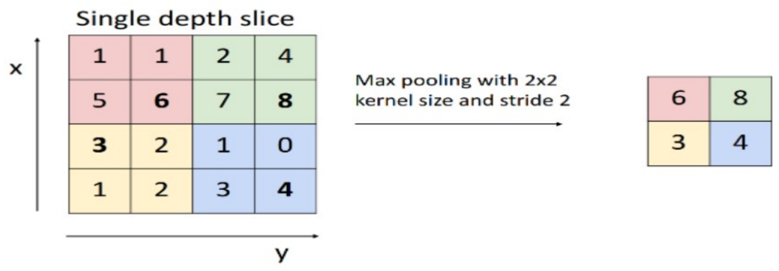


Figure 5: Max pooling layer

3.3.3 Comparison between Different CNN Architectures

The number of object detection models available currently is huge. Every now and then, a new model is surfacing in the deep learning community. So it is very important to decide which model would be the best choice for a particular work. Generally for choosing a model, there are two main aspects to consider. The first one is the accuracy that the model can provide. Secondly, the speed of a model is the time that a model takes to give a prediction.

Keeping these two factors in mind, a deep learning model should be chosen. These two characteristics of an object detection model are in the opposite spectrum. Accuracy and speed are disproportionate in nature. A model's accuracy can be developed in many ways. One of the most prominent ways for increasing model accuracy is to make a larger model by increasing the number of layers. This pattern of larger models can be observed from the advent of deep learning models. In the beginning, AlexNet [23] started with 8 layers and had 62.3 million learnable parameters. VGG-16 [30] is a 16-layer deep CNN model with 138 million learnable parameters. GoogLenet [31] contains 7 million parameters and has 22 layers. Resnet 50 [32] is 50 layers deep with a total number of learnable parameters of 23 million. The details of these model architectures are shown in Table 2.

Table 2: Number of layers, learnable parameters and error rates of different models

Model name	No. of layers	Parameter (M)	FLOPS (B)	Error rate (ImageNet [33])
AlexNet [23]	8	62.3	0.72	16.4
VGG-16 [30]	16	138	16	9.4
GoogLenet [31]	22	7	1.5	6.7
Resnet 50 [32]	50	23	3.8	5.25
MobileNet [8]	28	13	0.6	10.5

It is quite evident that the number of layers and learnable parameters has a great impact on reducing the error rate. But alongside reducing the error rate, we can see that the FLOPS is also increasing at a much higher rate. FLOPS is the number of floating-point operations done by a device per second. So higher FLOPS indicate that the number of operations needed to perform is more which will eventually take more time to produce a result thereby reducing the speed of a model.

In the case of different object detection architectures, the accuracy and speed are thus contrasting. Keeping that in mind, we have to work our way around to select a model that best fits our purpose. Our goal is to provide a dataset that is suited for indoor environments and which can be used to train a model that would work on a mobile device. A mobile is limited by its computational capabilities. It does not have higher memory and ram like a computer. So the number of computations that can be performed effortlessly is limited. Furthermore, our idea concerns on real time detection of objects in the indoor environment. In order for a system to be real time, it need to have a high speed so that it can recognize the objects by taking input from fewer frames of the video feed. As more accurate models tend to take more time to give class predictions, so we cannot use these heavier models for our work. In case of a mobile based object detection system which is real time, we can give up the accuracy of the model to some extent in order to achieve the required speed to make the model work in real time. So for our work, we have to select a model that would give us higher speed in contrast to high accuracy.

3.3.4 Comparison of Object Detector Models

There are a number of object detection models available and the number of models are increasing in a rapid pace. The most popular models are Faster R-CNN [25], Faster R-FCN [34], YOLO (you only look once)[35], SSD (single shot detector)[3]. There are three main metrics based on which an object detection model can be evaluated. These are :

- IoU (intersection over union)

- mAP (mean average precision)
- Rendered frames per second (FPS)

Intersection over union is a measure of how closely the bounding boxes are placed by the model overlaps with the actual bounding box in the test image. In some cases the IoU value has a particular threshold value by which the IoU values are accepted or not for every test data. Then we have the mean average precision which is the average value over the whole dataset for their precision and recall values combined together using definite formulas. High mAP value indicates that a model is more accurate and precise but the execution speed is not guaranteed in any way. Lastly we have the FPS which is basically a measure of how fast the model can provide a result. Analysis of the models mentioned above based on these metrics are summarized in Table 3

Table 3: Comparison of mAP and FPS of different object detection models based on training on PASCAL VOC0712, and MS COCO[36]

Model	**mAP**	**FPS**
Faster R-CNN (VGG160)	73.2	7
Fast YOLO	52.7	155
YOLO (VGG16)	66.4	21
YOLO v2 288x288	69.0	91
YOLO v2 352x352	73.7	81
SSD300	74.3	46
SSD512	76.8	19
SSD300 (batch size 8)	74.3	59
SSD512 (batch size 8)	76.8	22

Based on the values of this table, we can observe that Faster R-CNN gives a mAP value of 73.2 which is quite accurate. But the frames per second are only 7 which can never be helpful for making real-time object detection on mobile devices. Different versions of YOLO and YOLO v2 are also quite well in terms of their FPS value. So their speed is better than Faster R-CNN but the drop in mAP value is really low which again causes problems for creating a robust model. Lastly from SSD, we can see that it has higher accuracy, almost the same or more than Faster R-CNN. Simultaneously, the FPS value of SSD is not dropping quite low. It's a bit lower than YOLO but the result is not drastically different. Such a drop in FPS can be considered given that the accuracy is quite better. Due to this SSD can be considered a faster model with a considerate drop in accuracy. The above mentioned model's performance is summarized in Table 4

Table 4: Performance of the “critical” points along with optimality frontier [37]

Model summary	**Minival mAP**	**Test-dev mAP**
SSD + MobileNet (fastest)	19.3	18.8
SSD + Inception v2 (fastest)	22	21.6
R-FCN + Resnet	32	31.9
Faster R-CNN (Most accurate)	35.7	35.6

Based on this study, we came to conclude that we would require a faster model. As our work concerns real-time object detection on a mobile device, so we want a model that is faster and gives the accurate class label quickly even though the precision of the detection is reduced to some extent. Thus SSD+MobileNet has been chosen for our work. The model is available in the TensorFlow model zoo. A pretrained version of SSD+MobileNet which has already been trained on the MS COCO dataset has been chosen for our work. Another similar pre-trained model Efficient Det D0 which is trained on MS COCO has been chosen as it also has low inference time i.e. faster detection speed. A list of different available models on the TensorFlow 2 Detection Model Zoo [38] is provided in Table 5.

Table 5: Speed and mAP trade-off of different models[38]

Model Name	**Speed (ms)**	**MS COCO mAP**
EfficientDet D0 512x512	39	33.6
SSD MobileNet V2 FPNLite 320x320	22	22.2
SSD ResNet50 V1 FPN 640x640 (RetinaNet50)	46	34.3
SSD ResNet152 V1 FPN 1024x1024 (RetinaNet152)	111	39.6
Faster R-CNN ResNet50 V1 640x640	53	29.3
Faster R-CNN ResNet101 V1 1024x1024	72	37.1

From Table 5 we can have a clear understanding of the speed and accuracy of the models available on TensorFlow model zoo. The highest accuracy of 39.6 can be achieved on the SSD ResNet152 V1 FPN 1024x1024 (RetinaNet152) model but it has a huge inference time which can never work for a real time object detection system. All the architectures except for SSD

MobileNet V2 FPNLite 320x320 has a high value of inference time and are not appropriate for real time object detection framework. EfficientDet D0 512x512 has been taken into consideration as the accuracy is increased a lot given the increase in inference time also. There is a decrease of 77% speed and 51% increase in accuracy from SSD MobileNet V2 FPNLite 320x320 which is a substantial improvement and can be taken into consideration even though it's lack of speed to some extent.

SSD MobileNet V2 FPNLite 320x320 and EfficientDet D0 512x512, these two models are fine-tuned with our dataset in later steps where we have dived deeper into how the images are collected, augmented, and finally the models are trained with the collected data.

Chapter 4

Dataset Construction

Our dataset consists of 5196 unique images of 8 categories with 640x480 resolution. In this section we describe the criteria based on which the object categories, as well as their candidate images are selected for our proposed dataset. Furthermore, the annotation technique as well as data pre-processing techniques are discussed in the later portion of this section. Finally our dataset is compared against other indoor object datasets to show the uniqueness of our dataset.

4.1 Object Category Selection

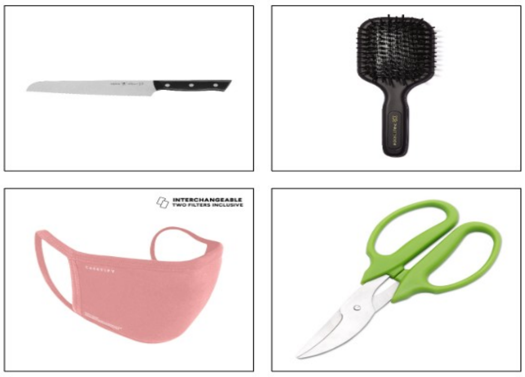
In accordance with the category selection criteria mentioned by Lin, Tsung-Yi, et al [18] in their work of the MS COCO dataset, which is one of the most widely used dataset for tasks of object detection and recognition, we have kept our dataset limited to "thing" categories only and have excluded the "stuff" categories. Here "thing" categories can be defined as object instances that can be easily identified and annotated under a certain category (mask, key), whereas objects falling in the "stuff" categories are boundary-less (grass, pavement). The scope of our proposed work is accurate detection of indoor objects, and thus objects that fall under only "thing" categories are taken into consideration for our dataset. The objects falling under "stuff" category will not be too relevant in the indoor context.

We used different sources to collect the object categories of "things". At the beginning we made a list of categories belonging to the "indoor", "accessory", "electronic" and "kitchen" super-category from the MS COCO dataset. This gave us "book", "clock", "vase", "scissors", "teddy bear", "hair dryer", "hairbrush" under the indoor category, "shoe", "hat", "backpack", "umbrella", "eye glasses", "handbag", "tie", "suitcase" under the accessory category, "laptop", "TV", "mouse", "keyboard", "remote", "cellphone" under the electronics categories, and "bottle", "plate", "wine glass", "cup", "fork", "knife", "spoon", "bowl" under the kitchen category. Among these "vase", "teddy bear", "umbrella", "suitcase", and "tie" are not relevant to everyday use and thus out of the scope of our work. Although "clock" is relevant to regular day to day life of visually impaired users, the use of traditional clocks have drastically reduced due to the emergence of other smarter technologies like smart watches and voice assisted applications in mobile phones and thus the "clock" category is dropped from our consideration. Similarly different object categories belonging to the "kitchen" super-category is dropped from consideration as a real-time system can be built using pretrained weights of the MS COCO dataset to detect those objects and even if due to lack of training data some of those objects are erroneously detected it won't pose much of a danger to a visually impaired person's daily life. "Knife" and "scissors" on the other hand, needs accurate detection as these are directly concerned with the safety of the visually impaired user and hence these two categories are taken into consideration for our dataset even if there are labeled instances of these categories present in the MS COCO dataset. On the other hand, the "hairbrush" and "shoe" category, although present in the original 91 classes of the MS COCO dataset, was later dropped in the 80 labeled instance categories. The "laptop" category was chosen even though it had labeled instances in the MS COCO dataset because its number of instances were low compared to the other object instances of the MS COCO dataset. Often it was seen that books, briefcases and other similar looking objects were being erroneously detected as laptops. Thus by compiling categories from selected super-classes of the MS COCO dataset, we decided on "shoe", "hairbrush", "laptop", "knife", "scissors" categories.

To further extend our set of object categories, various adults ranging in ages from 22 to 26 were asked to define objects they interact with in their daily life in indoor environments. From their inputs and the co-authors' discussions three other object categories were added to the list, namely "key", "mask", "watch". While taking these object categories into consideration factors like how commonly they occur and how relevant they are to a visually impaired user's daily life were taken into account. Our final list containing the 8 categories along with their number of object instances are shown in Figure 11.

4.2 Image Collection

Upon deciding on the object categories, collecting a proper set of representative images was our next aim. Both iconic [39] and non-iconic [18] object images were collected for our dataset. Typically a single object almost centrally placed in an image is considered as an iconic-object image, shown in Figure 6a. High quality object instances can be found from iconic images, but the contextual data and non-canonical viewpoint which is quite crucial in indoor object detection for visually impaired users are often missing in iconic images. These failings are fulfilled by incorporating mostly non-iconic images in the dataset, shown in Figure 6b. It is proved [40] that non-iconic images contribute to better generalization of the dataset, which can further be used to train real-time lightweight models. In our dataset, we incorporated a few iconic images and mostly non-iconic images. The iconic images contribute to the dataset having high quality object instances, whereas the non-iconic images contribute to better contextual information.


(a) Iconic object images

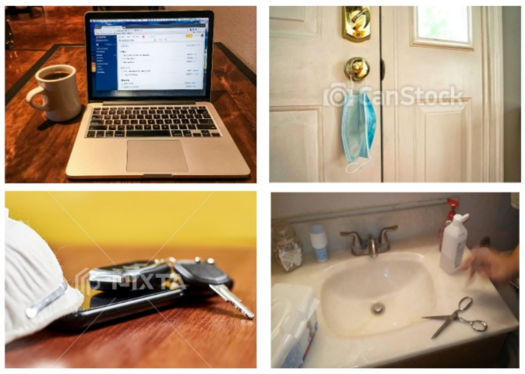

(b) Non-Iconic object images

Figure 6: Examples of iconic and non-iconic object images

All of the images were collected via web scraping on Google using Google Image Downloader [41] of Python, which is a built-in package. While scraping for the iconic images, giving only the object category was enough, however, scraping for non-iconic images required stating the context of the images. We have carefully considered most of the possible contexts in the indoor environment of every object category. We have considered color of the object, placement, and surrounding environment as important contexts. Different object category along with their contexts are mentioned below -

- **Key :** on table, beside beverages, beside food items, beside wallet, beside masks, hanging on the wall, with different keyrings, different colored (silver, gold)
- **Mask :** on table, hanging on wall, folded mask, mask in a box, different colors of the mask, different types of masks (surgical, N95, KN95, cloth)
- **Watch :** on table, beside laptop, beside books, beside phone, in clutter, smart watch, folded, straightened, different types of belt (chain, silicone, leather etc)
- **Shoes :** in rack, back view, front view, top view, single pair, men's shoes, women's shoes, different types (sandals, boots, closed toed, formals etc)

- **Scissors :** on table, in drawer, in bathroom, inside pencil holder, beside papers, beside sewing kit, different types (gardening, sewing, surgical etc)
- **Knife :** on table, on knife stand, in drawer, beside food, beside beverage, beside vegetables, different types (boning, slicer, bread etc)
- **Laptop :** on table, on bed, beside food, beside beverages, beside pet, with sticker, different types (dell, acer, hp, apple), open, closed
- **Hairbrush :** on table, in drawer, in ziplock bag, beside makeup, different types (comb, round, rollers etc)

We downloaded the image instances with almost even distribution of contextual information. The images downloaded through the web scraper had repetitions due to the same image being available on multiple websites uploaded by multiple authors. We have manually removed those images and selected the ones having more contextual information. In the rare cases where the web scraper returned no significant images, manual searching was done to find image instances with contextual information using the contexts as keywords while searching. In this way a total of 5196 images were collected in 8 categories spanning over approximately 60 worker hours. Some examples of every category are shown in Figure 7.

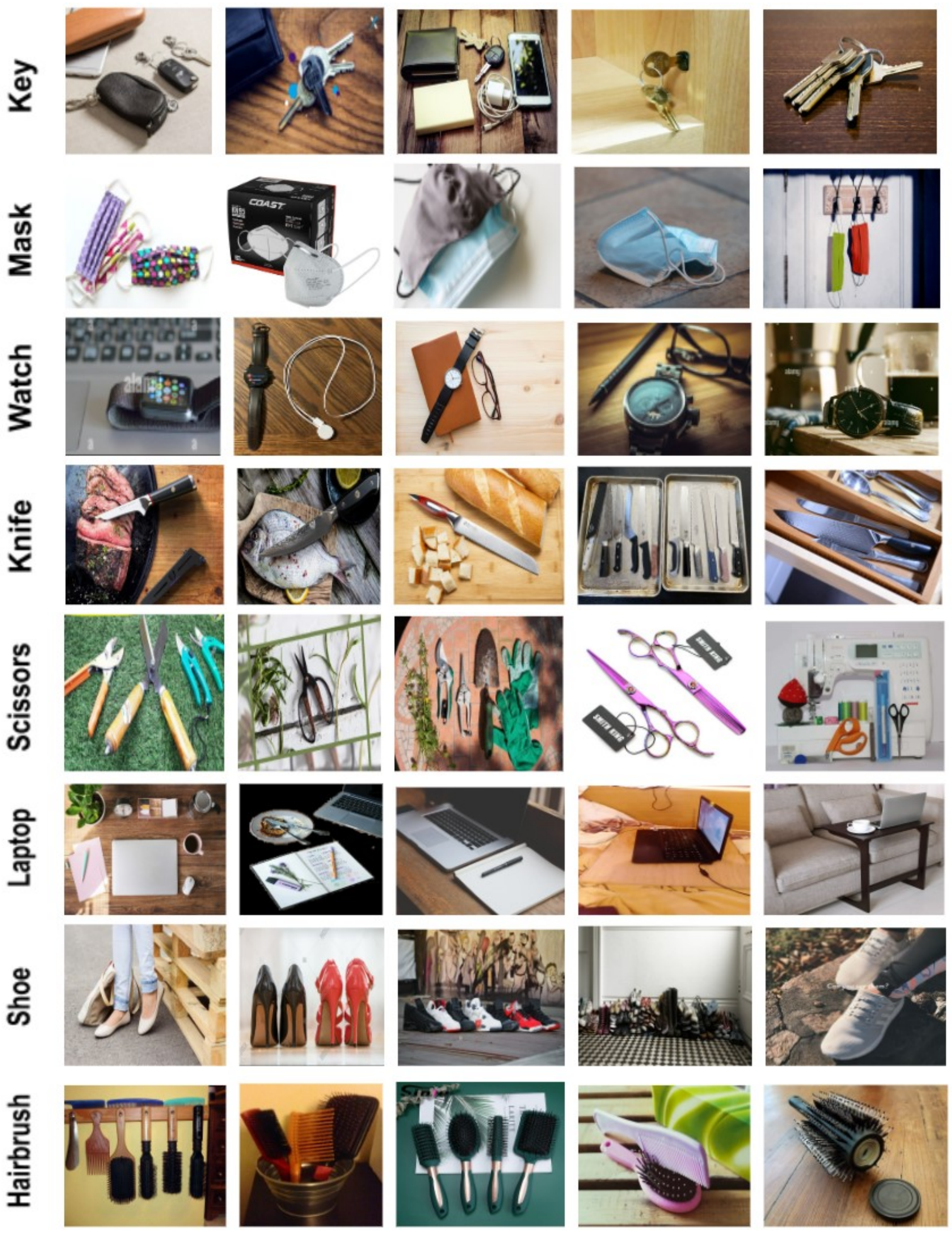
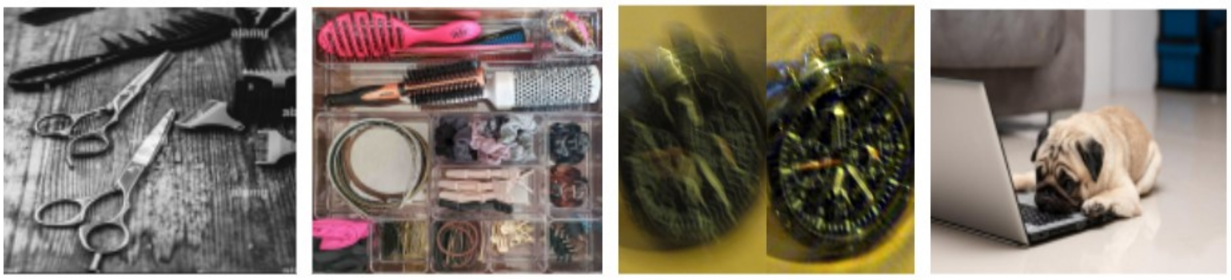
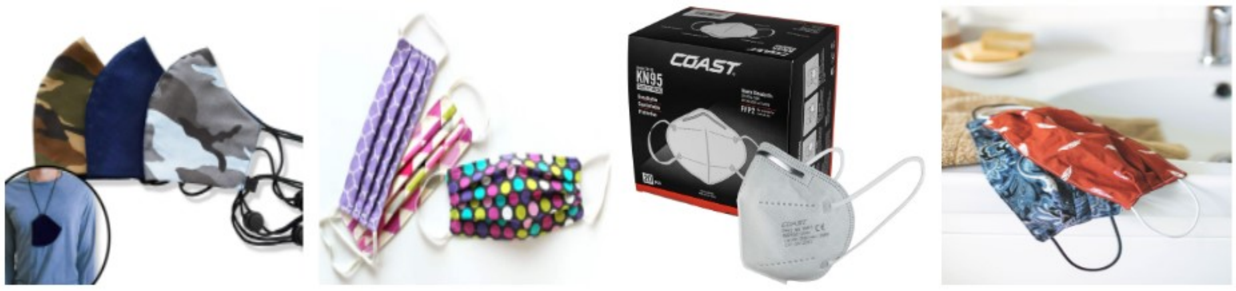


Figure 7: Samples of images of all object categories present in our proposed dataset

The object instances present in our dataset have various characteristics which are significant in terms of a visually impaired person's mobility. For example, different lighting conditions, occlusion of the object of interest, geometrical change etc are taken into consideration while collecting the images. This dataset can be incorporated into real-time systems be it mobile based or sensor based. In that case, the visually impaired person has to point his/her mobile phone/handheld device towards the object of interest that needs to be detected. Often it might be the case that the object is not properly in the view of the camera field, i.e the object is occluded, and/or the movement is done too quickly resulting in a blurry object in the view of the camera field. These factors are taken into account while collecting the images and some of such examples are shown in Figure 8a. On the other hand, most of the objects present in the real world has different variations available, which is also reflected in our dataset by having high intra-class variance. This is shown using the "mask" object category in Figure 8b.



(a) Different lighting conditions (blurred), background clutter and occluded object of interest



(b) High intra-class variance of "mask"

Figure 8: Image characteristics significant to visually impaired user's mobility

In contrast to the existing indoor datasets, the main strength of our proposed dataset is the object instances present in our dataset provides different conditions that contribute to a certain image having good features, such as: moderate to heavily occluded objects, light variance, cluttered background, high intra-class variance etc. to increase the accuracy of the detector that can be built based on this dataset.

4.3 Image Annotation

We have used the LabelImg software [42] for annotation of the object instances. It is a graphical image annotation tool that is built on Python and uses Qt for its GUI. The annotations generated by this tool are saved as XML files, which follows the format of PASCAL VOC [43]. LabelImg labels the objects using bounding box that is later saved in the XML files along with its x and y coordinates. Along with the x, y coordinates of the bounding boxes of the objects, the height and width of the image, and the class label is also saved in the XML files.

All of the images are resized to 640x480 which is the same as the images of the MS COCO dataset. During the training process of different machine learning, the images are again resized by the model itself according to the input size of the model that is being used. Since labeling all the object instances of 5196 images is quite a labor-intensive task, we recruited 10 data annotators from our institution who were given specific instructions for labeling the images. No prior experience was needed for the annotation process and upon recruitment the annotators were given necessary training on installation of the necessary software and the annotation process. After each annotator has completed their work they received a remuneration.

The bounding boxes are drawn as tightly around the object of interest as possible to avoid extra background pixels. In the cases of occlusion, a bounding box is drawn surrounding the part of the object that is visible. In case of multiple instances of the same category separate bounding boxes are drawn around each instance and labeled separately. In case of multiple objects of different categories, instances of each category are labeled separately with their corresponding category. Figure 9 shows an original image along with corresponding annotation.

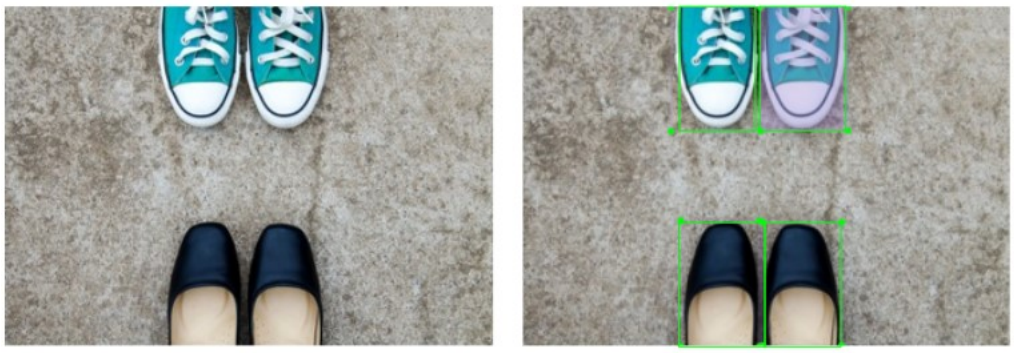


Figure 9: Annotated image example

The original LabelImg software is cloned from its repository and a text file containing the labels of the object categories were added with the master repository. The annotators were given the modified version of the LabelImg software, where after drawing the bounding box, they had to select the class label from a drop down menu which showed the object categories added by us. This modification was done to the original software to avoid any spelling mistake during the labeling process as the labels are class sensitive as well. This process took almost 140 worker hours to be completed. After completion of the annotation process, we checked the annotations for discrepancies and in the rare occasion of any error being found, those were manually corrected.

4.4 Data Pre-processing

After collection and annotation of the images, the images are separated into an approximate 80:20 split for training purposes. This split is applied to each context as well. The training set consists of 4128 unique images, whereas the test set consists of 1068 unique images. For better generalization of the training process, the training set is then augmented using affine scaling, affine rotation, affine translation and gaussian noise. Each of these augmentation techniques is applied on the entire training set and the resulting images are added with the original training data. After the augmentation process the size of the training set becomes 20,640. This augmentation is useful in improving accuracy and efficiency of different machine learning models by generating new and different examples of the same image. Figure 10 shows four augmentation techniques applied on the same image.

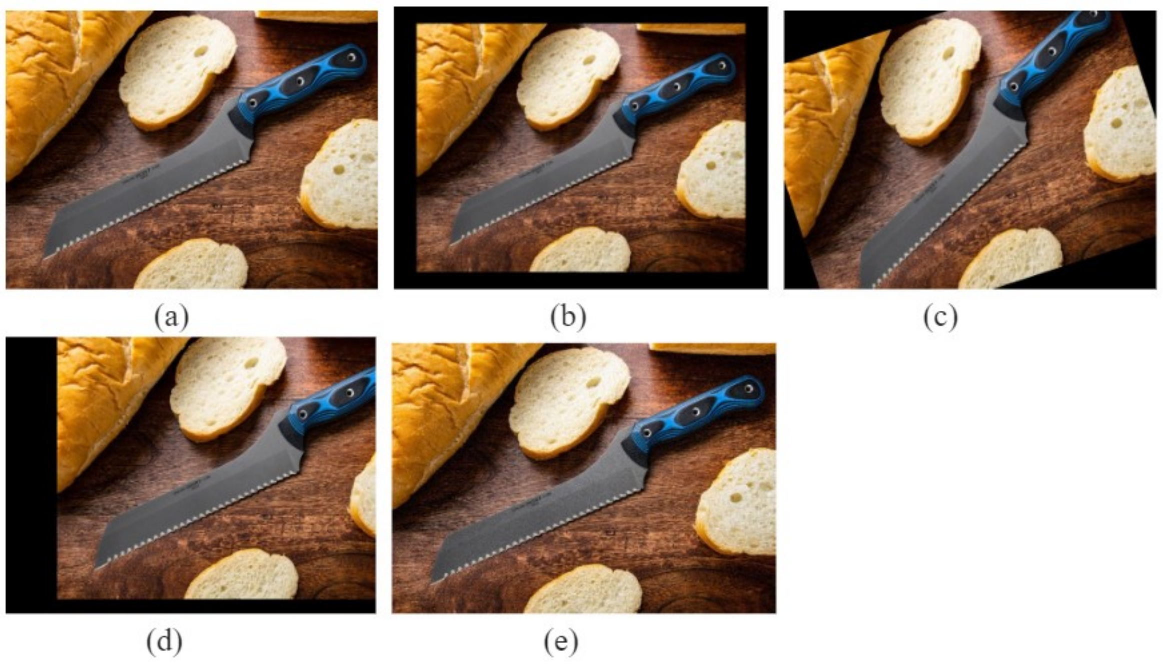


Figure 10: (a-e) Original image, scaling, rotation, affine translation, gaussian noise

The original training data had separate XML files for each of the 4128 images. These XML files are converted to a single CSV file where each row corresponds to a bounding box of an image. This in turn created multiple rows for a single image in case of multiple object instances being present in one image. After augmentation the newly generated bounding boxes are saved in a separate CSV file. If any bounding box falls out of the image pane, that is discarded and any bounding box that is partially out of the image pane is clipped. The new CSV file is then concatenated with the original CSV file to make a single CSV file. The XML files of the test set are kept as is.

4.5 Dataset Statistics

On average our dataset contains around 650 instances per object category. A more detailed version of per category object instances is shown in Figure 11. While our dataset has fewer categories than IODR, it has more instances per category which is hypothesized [18] to be useful for learning complex models capable of more accurate detection. On the other hand, although the MCIndoor20000 dataset has slightly more object instances per category, our dataset has more categories which will be again useful for more robust detection.

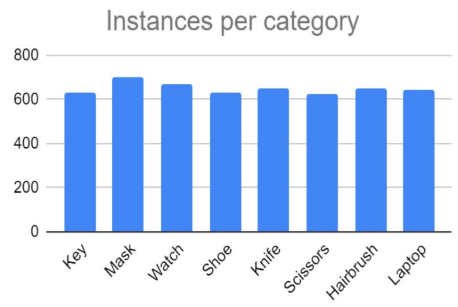


Figure 11: Number of annotated instances per category

Our dataset contains multiple images having more than one object instance and categories, whereas all the images of the MCIndoor20000 dataset has a single object instance and object category in all the images. IODR has a few images with more than one object instance and categories, however, that number is quite low compared to our dataset. Since our dataset has images containing more than one object instance and multiple categories of objects, it can be hypothesized in accordance with the hypothesis of Lin TY et al. [18] that our dataset contains more contextual information. Table 6 shows number of images (containing at least one object of the corresponding class) and number of object instances of all the categories present in our dataset along with average number of object instances per image.

Table 6: Statistics of our proposed dataset

Object Category	Train		Test		Average
	Images	Object	Image	Object	
Key	504	1313	130	328	2.61
Mask	543	1202	157	301	2.21
Watch	524	890	137	228	1.7
Scissors	500	912	128	228	1.82
Knife	520	1473	130	369	2.83
Laptop	516	550	129	140	1.1
Shoe	501	2167	127	545	4.33
Hairbrush	520	1450	130	362	2.79
Sum	4128	9957	1068	2501	

4.6 Uniqueness of our Dataset

Our presented dataset holds 5196 images of everyday use indoor objects spanning across 8 categories, namely, key, mask, watch, hairbrush, laptop, shoes, scissors, and knife. Our dataset has on average 650 images in per object category and 2.42 object instances in per image. The main goal of our work is to introduce a dataset that can be used for robust detection and recognition of everyday use indoor objects. Our work is unique from the existing works by having day to day use objects. For example, even though a large scale face mask recognition dataset is available [44], this dataset does not have instances of face masks in indoor context, for example on the bed or hanging on a hook on the wall, which are significant contexts in indoor environments for a visually impaired person. However, object categories of laptop, scissors and knife are already present in the Microsoft COCO [18] dataset, but those instances often collide with other similar looking objects and thus lead to erroneous detection, due to the smaller number of instances present in the dataset compared to other object classes, which is shown in Figure 12.

Detecting objects such as scissors and knives erroneously might lead to mild to serious danger in the daily activity of a visually impaired user. Our dataset, when used to fine-tune a model pre-trained on the MS COCO dataset, will provide more accurate detection of the objects already present in the MS COCO dataset, namely scissors, knife and laptop.

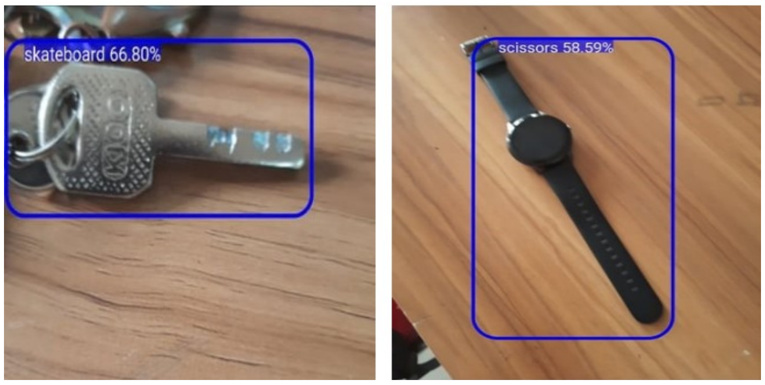


Figure 12: Detection inaccuracy of the MobileNetv2+SSD model pre-trained on the COCO dataset

On the other hand, currently available indoor object dataset presented by different authors [22, 27, 24] have their own limitations. The detailed description of currently existing indoor datasets can be found in Table 1, where we can see that Bashiri et al. [22] presented a dataset named MCIndoor20000 containing 20,000 images of 3 categories of landmark indoor objects like doors, signs, stairs, which are although useful for navigation however does not hold instances of objects a user interacts with in his daily life for performing regular activities. However, in the MCIndoor20000 dataset, there is 2055 unique images in 3 categories, which makes their average instance per category almost similar to ours. In fact ours is better as we have more number of object category. Afif, Mouna, et al. [27] presented a dataset named IODR which contains 8000 images of 16 object categories having instances of objects required for navigation in emergency situations. However, our presented dataset can be used in everyday situations as those are more common to a visually impaired person's life rather than emergency situations. Adhikari, Bishwo, et al. [24] presented the TUT dataset which has 2213 unique images of 7 object categories. However, the objects present in their dataset may be useful in office environment's navigation for the visually impaired user, however, the research gap of not having dataset containing everyday use indoor objects still pertains. The main strength of our dataset is that it contain on an average 2.42 object instances per image, which is the highest among all available indoor object datasets. This proves the higher contextual information present in our dataset, according to the hypothesis of Lin, Tsung-Yi, et al [18]. Our work can be further extended with the IODR for a more robust detection in both regular and emergent situations.

Chapter 5

Experimental Design

Our work focuses on dataset generation and training different models using our custom dataset. We took different pre-trained models from the TensorFlow model zoo and finetuned those already trained models using our custom model so that they can be used effortlessly in indoor object navigation for visually impaired users. The models are trained on Google Colab. In this section, we'll be discussing the processes and steps that need to be followed to get an object detection model up and running using a custom dataset in Colab.

5.1 TensorFlow Object Detection API

TensorFlow [7] is an open source library used largely for building deep learning applications. It is developed and maintained by Google and has an extensive community. The main objective of this library is to provide fast numerical computing which is a necessity for training any deep learning model. TensorFlow uses data flow graphs to carry out faster numeric computations. The TensorFlow object detection API [7] is an open source framework which is based on the TensorFlow library. It helps in creating an object detection model easily and train on different platforms. The trained models can further be froze and deployed into different devices. The framework also has different pre-trained models available. These pre-trained models are pre-trained on MS COCO dataset and can be fine tuned with other datasets if required. For our work, we have chosen -

- SSD MobileNet V2 FPNLite 320x320
- EfficientDet D0 512x512

The intention of our work is to provide a model which will be fine tuned with our custom dataset that is focused on indoor objects. Our proposed dataset is specifically tailored for different indoor objects so that visually impaired people can detect and identify different objects in their indoor environment.

5.1.1 Platform Specification

Our pre-trained models have been fine-tuned with our custom dataset. We have performed model training on Google Colaboratory or Colab for short. It is a web platform built by google for research purpose. Colab enables the user to run python code from their browser. It is a platform which specifies on work related to machine learning, deep learning, artificial intelligence, data analysis etc. allowing users to run their code on the cloud. It grants users the ability to train machine learning or deep learning models without necessarily having a powerful machine. Colab provides both GPU and TPU instances. This largely contributes to users without access to personal GPU, to work on deep learning models without any hassle. Colab offers either Nvidia K80 or T4 GPU but it is not ensured which GPU will be allocated due to limited GPU availability. Table 7 provides a detailed view of the hardware specifications of Google Colab.

Table 7: Hardware Specification of Google Colab

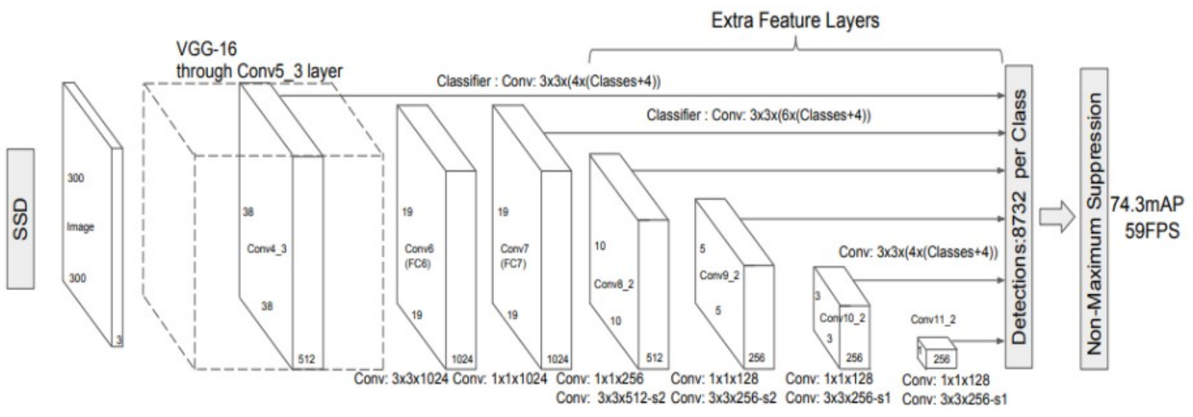
Properties	**Values**
GPU	Nvidia K80/T4
GPU Memory	12GB/16GB
GPU Memory Clock	0.82GHz/1.59GHz
Performance	4.1TFLOPS/8.1TFLOPS
No. CPU Cores	2
Available Ram	12GB
Disk Space	358GB

5.2 Selection of Baseline Models

While choosing a deep convolutional neural network model for an object detection task, accuracy used to be the main concern. But for our case, we want to present the model such that it can be run in low end devices such as mobile phones and iot. In these smaller devices, computation is a key factor as they don't have much resources due to their smaller size and capacity. Keeping that into consideration, we had to work our way around and choose a model that would provide a mediocre accuracy value, but the model would be very fast. As our goal is to develop a model which can be used on a mobile device and it would run in real time, so we have to consider two things simultaneously. Accuracy and speed would thus play a huge role in our case due to the proposed mobile application being real time. More accurate models usually require higher computational power. Thus we chose models which were less accurate but has a high speed. Inference time is the formal term for defining the speed of a model. Higher inference time means that the model is slow and lower inference time indicates that the model is faster. The pre-trained model **SSD MobileNet V2 FPNLite 320x320** and **EfficientDet D0 512x512**, both provides medium accuracy and has a low inference time, i.e. they are faster than other heavy weight models. Both of these models mAP and inference times have been shown in Table 5.

5.2.1 Architecture of SSD MobileNet V2

This is a single stage object detection model first proposed by Sandler, Mark, et al. [4]. In SSD MobileNet V2, the underlying network takes the input image and generates bounding box co-ordinate along with class label prediction. A simplified architecture of the SSD model is shown in Figure 13. Inverted residual structure is the core of this architecture where thin bottleneck layers are present for every input and output of each residual block.



Extra Feature Layers
VGG-16 through Conv5_3 layer
Classifier : Conv: 3x3x(4x(Classes+4))
Classifier : Conv: 3x3x(6x(Classes+4))
SSD
Image
Conv4_3
Conv6 (FC6)
Conv7 (FC7)
Conv8_2
Conv9_2
Conv10_2
Conv11_2
Conv: 3x3x(4x(Classes+4))
Detections:8732 per Class
Non-Maximum Suppression
74.3mAP 59FPS
Conv: 3x3x1024
Conv: 1x1x1024
Conv: 1x1x256 Conv: 3x3x512-s2
Conv: 1x1x128 Conv: 3x3x256-s2
Conv: 1x1x128 Conv: 3x3x256-s1
Conv: 1x1x128 Conv: 3x3x256-s1

Figure 13: Single stage object detection model architecture (SSD)[3]

In case of SSD MobileNet V2, the first segment of feature extraction is done using MobileNet architecture, which is shown in Figure 14.

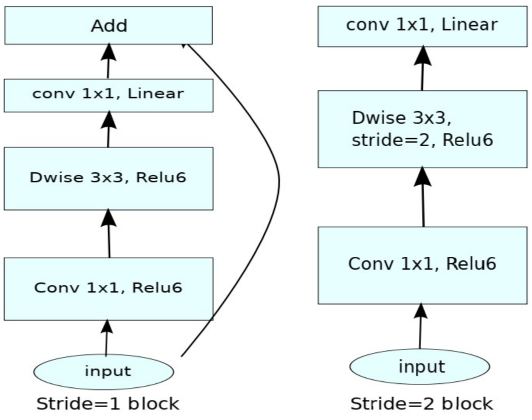


Figure 14: Convolutional block of MobileNetV2 [4]

MobileNetV2 has two types of block in their architecture. One block is of stride 1, other of stride 2 whose purpose is to downsize the input images. It has an inverted residual structure for removing the non linearities of the narrow layers. This helps in reducing the number of calculations and thus resulting in efficient use of memory which is helpful for mobile devices. This architecture has adjustable hyperparameters which are the input image resolution and width multiplier. These two hyperparameters can be tuned to obtain the required tradeoff between accuracy and speed.

5.2.2 Architecture of EfficientDet

EfficientDet[45] comes with several optimizations that improves the performance of the model. They introduce a weighted bi-directional feature pyramid network (BiFPN) whose purpose is to allow rapid fusion of features that are multiscaled. Secondly they introduce a compound scaling method for scaling the resolution, depth and width throughout the network. It strives to attain a good speed without necessarily reducing the accuracy of the model. The architecture of EfficientDet is shown in Figure 15

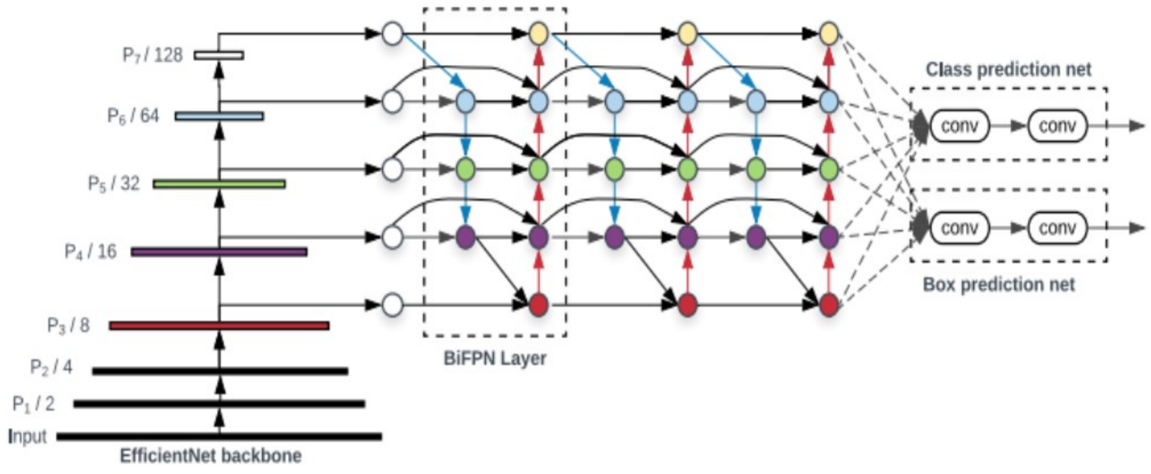


Figure 15: Architecture of EfficientDet [45]

Instead of using typical FPN for multi scale feature fusion, they used BiFPN. The problem with FPN is that while fusing the input features, it sums up all of them without distinction. But the input features are at different resolutions and hence just summing them up is not an efficient way of feature fusion as each feature would contribute to the output differently. A BiFPN network learns the important features by introducing learnable weights to the features.

Another improvement from their part is that instead of relying on increasing the size of the model to increase the accuracy, they take advantage of the scaling factor of the feature network and the class label prediction network to improve the accuracy and speed of the network. A compound scaling is done for the architecture which scales up the resolution, depth and width of the architecture, the feature network and the class label prediction network. Along with the BiFPN network, they used Efficientnet model to create EfficientDet which improves the accuracy and performance without increasing the size of the model.

5.3 Model Training

5.3.1 Setting up the Folders in Colab

Initially we setup our directories inside the Colab root folder. Paths to various directories are saved and the directories are created. We will require 3 files which are instantiated initially. These are the pipeline config, generate tfrecord, and lastly the labelmap. Their descriptions will be provided later on. Figure 16 shows the folder structure of the Colab notebook.

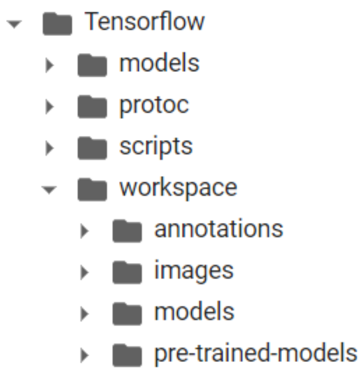


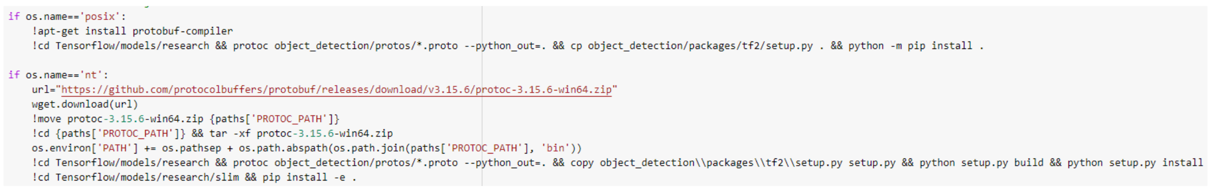
Figure 16: Folder structure of the notebook of Google Colab

5.3.2 Downloading TensorFlow Model Zoo

After instantiating the files, we're going to download our model from the TensorFlow model zoo. We use "wget" to download the model from TensorFlow zoo and save them on the "Tensorflow/-models" folder.

5.3.3 Installing Protoc (Protocol Buffers), TensorFlow Object Detection API, TensorFlow

Protoc is downloaded through the commands shown in Figure 17 and it will help for the installation of Tensorflow object detection API by helping it to communicate over the network and store the relevant data successfully.



```
if os.name=='posix':
    !apt-get install protobuf-compiler
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && cp object_detection/packages/tf2/setup.py . && python -m pip install .

if os.name=='nt':
    url="https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/protoc-3.15.6-win64.zip"
    wget.download(url)
    !move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
    !cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-win64.zip
    os.environ['PATH'] += os.pathsep + os.path.abspath(os.path.join(paths['PROTOC_PATH'], 'bin'))
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && copy object_detection\\packages\\tf2\\setup.py setup.py && python setup.py build && python setup.py install
    !cd Tensorflow/models/research/slim && pip install -e .
```

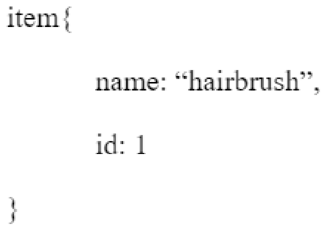
Figure 17: Downloading protobuf and installing tensorflow object detection api

5.3.4 Downloading the Pre-trained Model from TensorFlow Model Zoo

Various state-of-the-art pre-trained models are available on TensorFlow model zoo. It is available on TensorFlow website and maintained by google. These models are already trained on the COCO dataset and can be used out of the box for object detection. Our work is to leverage that pre-trained model's attributes so that we can train our custom dataset using that model. This process of leveraging an already trained model is achieved by transfer learning. We downloaded the two models mentioned above one by one and for each runtime, a single model is being used. Our pre-trained models will be downloaded on our "Tensorflow/workspace/pre-trained-models" folder mentioned above. The pre-trained model is downloaded and unzipped in that directory.

5.3.5 Creating Labelmap and TF Records

A labelmap is created which is a map containing the name of the class and an id for each class. A part of the labelmap is shown in Figure 18. This map would be created in the previously initiated labelmap file mentioned in section 5.3.1.



```
item{
        name: “hairbrush”,
        id: 1
}
```

Figure 18: Format of a portion of the labelmap

After a labelmap is created, we would create a TF record. It is a way of storing data to a file in binary format. This file and it’s data is actually being used while training the model. As the format of this type of file is in binary, so it is easier to read from this file and thus using a TF record speeds up the process of training a model with a new dataset.

Our dataset is converted to a .tar file so that it can be uploaded to Colab easily. Then it is unzipped and only the test images and XML files are converted to TF record format. At first the XML files of all the images are converted to a single CSV file and then the images are encoded and the encoded images and their respective data from CSV are converted to a TF record file named test.record.

For the train images, they are augmented and their XML is converted to a CSV format which was previously discussed in section 4.6. The images are encoded and the data from the CSV are together converted into a TF record separately on local machine. The file name is given train.record. It is then uploaded to Colab and moved to the respective folder alongside the test.record file.

5.3.6 Tweaking Pipeline Config File for Transfer Learning

The pre-trained model that we downloaded from the TensorFlow model zoo will have some other files along with it. The contents of the pre-trained model folder is shown in Figure 19.

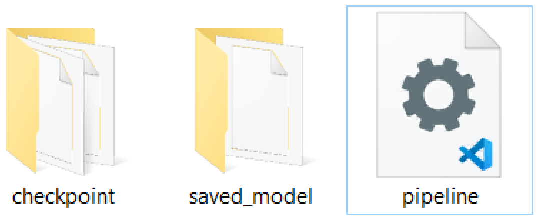
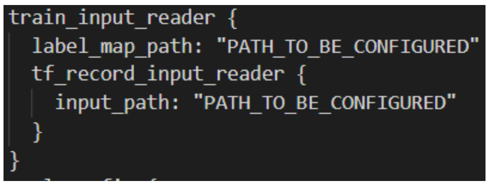


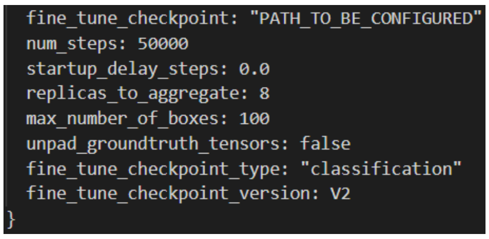
Figure 19: Content of the pretrained model folder

Now this pipeline config file is a baseline for this pre-trained model and it defines the architecture of the model and how the model actually looks like. We can take this pipeline config file and customize it according to our requirements so that we can use apply transfer learning on this pre-trained model and train this model on our custom dataset. It has a fine tune checkpoint, label map path and input path , the details of which is shown in Figure 20a, Figure 20b. We have to update this config file so that we can use this for training our custom dataset. The batch size has also been changed to 16. This is the maximum possible size that could be achieved on colab. The commands for updating the config file is shown in Figure 20c. The paths to the fine tune check points, input reader and label maps are all set to their particular directories and then the pipeline config file is saved for our particular model.



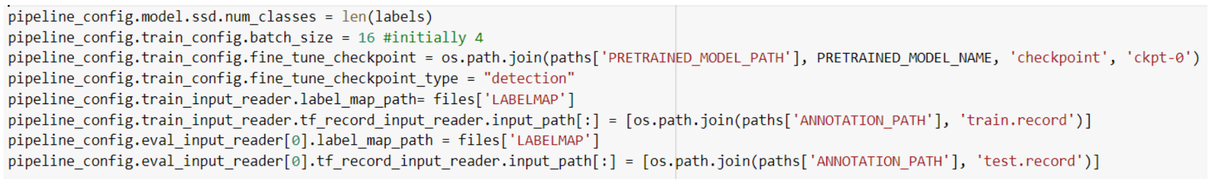
```
train_input_reader {
  label_map_path: "PATH_TO_BE_CONFIGURED"
  tf_record_input_reader {
    input_path: "PATH_TO_BE_CONFIGURED"
  }
}
```

(a) Labelmap path and input path



```
  fine_tune_checkpoint: "PATH_TO_BE_CONFIGURED"
  num_steps: 50000
  startup_delay_steps: 0.0
  replicas_to_aggregate: 8
  max_number_of_boxes: 100
  unpad_groundtruth_tensors: false
  fine_tune_checkpoint_type: "classification"
  fine_tune_checkpoint_version: V2
}
```

(b) Finetune checkpoint



```
pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 16 #initially 4
pipeline_config.train_config.fine_tune_checkpoint = os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'checkpoint', 'ckpt-0')
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= files['LABELMAP']
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'train.record')]
pipeline_config.eval_input_reader[0].label_map_path = files['LABELMAP']
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'test.record')]
```

(c) Update command

Figure 20: Details of the pipeline config file

5.3.7 Training the Model

The commands shown in Figure 21 is run to train the model with the pipeline config file just defined earlier. The TensorFlow object detection API provides a file named "model_main_tf2" which is used to train the pre-trained model that has been downloaded earlier.

```
TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'model_main_tf2.py')
```

```
command = "python {} --model_dir={} --pipeline_config_path={} --num_train_steps=8000".format(TRAINING_SCRIPT, paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'])
```

Figure 21: Training command for the model

The TRAINING_SCRIPT variable contains the path to the "model_main_tf2" file. On the command variable, we are specifying the pipeline config path to our custom pipeline config file. Also the number of steps has been defined to be 8000 for our training. The model directory will start training the model from the last checkpoint it was saved on which implies that the model has already been trained on a dataset. So training will start from that point onward using transfer learning. The dataset is trained with a batch size of 16 and number of steps 8000.

5.3.8 Evaluating the Model

Evaluation of the model is also done based on that particular config file. Instead of the number of steps, the checkpoint directory is provided so that the model can now be evaluated based on the training checkpoints. Both the models were evaluated separately. The command for evaluation is shown in Figure 22.

```
command = "python {} --model_dir={} --pipeline_config_path={} --checkpoint_dir={}".format(TRAINING_SCRIPT, paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'], paths['CHECKPOINT_PATH'])
```

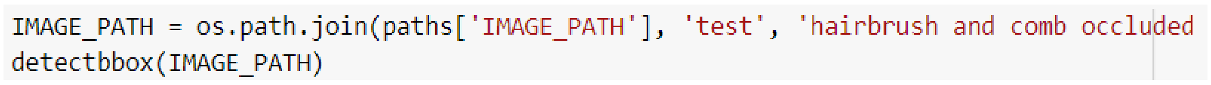
Figure 22: Evaluation command for the model

Evaluation is done based on precision, recall and loss. Precision gives the proportion of the correct detections among all the detections. Recall gives the proportion of actual objects that were captured. Finally loss values of the model gives a method of evaluating the performance of the model on the data it is trained on.

For SSD MobileNet V2 FPNLite 320x320, the mAP value computed was 29.5. The average recall value computed was 53.7. For EfficientDet D0 512x512, the mAP value computed was 39.4. The average recall value computed was 56. Detailed discussion on these results are done in the following sections.

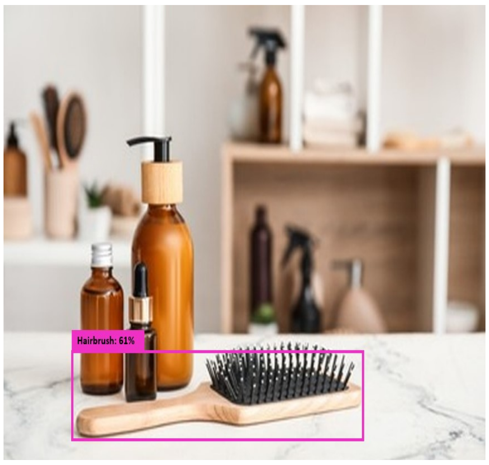
5.3.9 Testing an Individual Image

The model can be tested on Colab using individual images from the test set. To test an image, we select the directory of the particular image and use our finetuned models checkpoints to obtain the detection result which is the bounding box around the object along with an accuracy percentage of the class label. Running the command shown in Figure ?? would provide an output, which is shown in Figure 23b in colab where we will obtain a bounding box around the object in the image. Also the model will give a class label prediction and it's confidence value along with the bounding box.



```
IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'test', 'hairbrush and comb occluded
detectbbox(IMAGE_PATH)
```

(a) Command for testing a single image using its path



(b) Bounding box and class label prediction on test image

Figure 23: Testing on single image

We can check few of our test images in this way to see if our model is working correctly or not. Based on that performance, we can further fine-tune our model or change certain aspects of our dataset and retrain our model again if needed.

Chapter 6

Result Analysis and Discussion

Our dataset is split into a 80:20 split for training and test data, where the training set consists of 4128 unique images and the test set consists of 1068 images. This training dataset was later augmented and the training data size became 20640. Our dataset has been trained and tested on two light weight models namely SSD MobileNet V2 FPNLite and EfficientDet D0.

6.1 Evaluation Metrics

The evaluation metrics that were selected for result evaluation are Intersection Over Union (IoU) and mean Average Precision (mAP).

6.1.1 Intersection Over Union (IOU)

The IOU metric returns the difference between the area of ground truth annotations bounding boxes and the bounding boxes that are predicted by the model. This metric is specifically used by the PASCAL VOC challenge [46]. The ground truth bounding boxes are the original bounding boxes which were drawn around the object instances by our data annotators, whereas the predicted bounding boxes are the ones that the model predicts. For each object, multiple bounding boxes and their predicted class label and confidence score are returned by the model. The IOU formula [47] is shown in Figure 24.

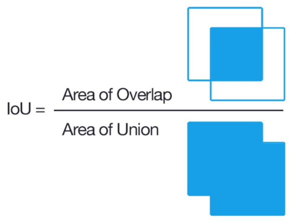


Figure 24: Intersection over union

Here the area of union is the total area covered by the ground truth bounding box and the predicted bounding box. Area of overlap is the intersection of the predicted and ground truth bounding box. When the ground truth box and predicted box do not intersect at all, the IOU score is 0, on the other hand when both of the boxes completely overlap with each other IOU score is 1. Standard IOU score is considered as > 0.5.

In our case we have taken those mAP values that fall within the range of 0.5 : 0.95 IOU, with a maximum 100 bounding box threshold value. This threshold determines that for a certain object instance the model can predict a maximum of 100 bounding boxes and from there only the one with the highest confidence score would be returned.

6.1.2 Mean Average Precision (mAP)

Mean average precision (mAP) can be calculated by taking the average of the average precision values of all the class labels in a multi-class classifier. The mAP incorporates the trade-off between precision and recall, and thus considering both false positives (FP) and false negatives (FN). This property makes mAP a suitable metric for most detection applications which is why it is widely used by different detection challenges [46, 48, 33].

Calculating the mAP value require calculation of -

- Confusion matrix
- Intersection over union (IOU)
- Precision
- Recall

Confusion matrix holds the values of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). These can be defined as -

- **TP** : ground truth label true, model predicted label true
- **FP** : ground truth label false, model predicted true
- **TN** : ground truth label false, model predicted false
- **FN** : ground truth label true, model predicted false

Precision and recall both are calculated based on the threshold value of IOU, which in our case is a range from 0.5 : 0.95. The IOU value matters in this case because the threshold determines whether a certain sample falls under TP or FP. For example, a lower threshold value might determine the predicted bounding box as a FP, whereas a higher threshold value will give the predicted bounding box as true positive. The following formula are used to calculate precision and recall -

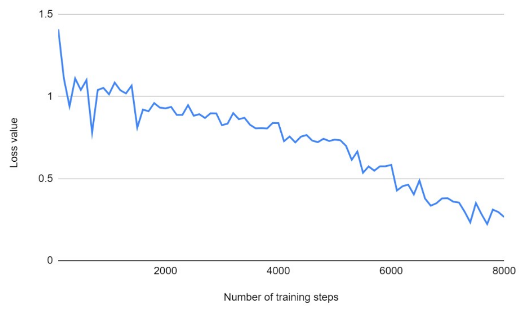
$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

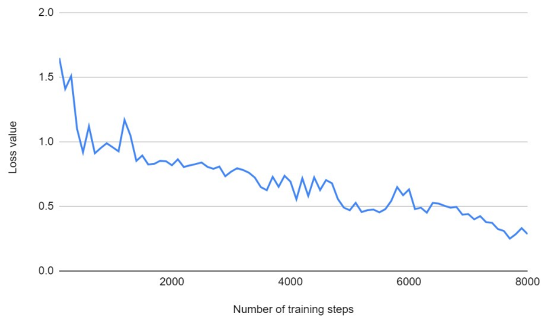
The weighted mean of precisions at each threshold of IOU gives the average precision (AP) value, where the weights are the increase in Recall value from the previous threshold value. The mean average precision (mAP) is the average of AP values for each class label.

6.2 Loss Analysis

Our dataset is trained on both the SSD MobileNet V2 FPNLite and EfficientDet D0 models for 8000 steps. The loss curve for SSD MobileNet V2 FPNLite is shown in Figure 25a shows that at the end of 8000 steps the loss value of the model is 0.26540, which started from 1.41253. The loss curve for EfficientDet D0 is shown in Figure 25b shows that at the end of 8000 steps the loss value of the model is 0.2744, which started from 1.6524. Both of these loss curves denote that the model has learnt the necessary features from the given dataset. Due to the resource limitation of Google Colab training the model for more than 8000 steps was not possible. Had that been possible, we could have obtained a smoother loss curve and the convergence of the model could also be visible within the curve. However, in current circumstances from the decreasing value of the loss we can still conclude that the model has learnt the necessary feature sets and thus is able to detect objects accurately.



(a) Loss curve of SSD MobileNet V2 FPNLite



(b) Loss curve of EfficientDet D0

Figure 25: Loss curves

6.3 Average Precision (AP) and Average Recall (AR) Analysis

The performance of our dataset on both the models is summarized in Table 8 and Table 9. The 12 metrics that are used here are primarily the detection evaluation metrics used by COCO [48].

Table 8: Different metrics of average precision (AP)

Model	$AP^{IoU=0.5:0.95}$	$AP^{IoU=0.5}$	$AP^{IoU=0.75}$	AP^{Small}	AP^{Medium}	AP^{Large}
SSD MobileNet V2 FPNLite	29.5	55.3	31.8	0.4	12.7	34.5
EfficientDet D0	39.4	70.3	39.4	1.1	21.5	43.0

Table 9: Different metrics of average recall (AR)

Model	$AR^{max=1}$	$AR^{max=10}$	$AR^{max=100}$	AR^{Small}	AR^{Medium}	AR^{Large}
SSD MobileNet V2 FPNLite	24.7	47.8	53.7	0.1	28.8	59.4
EfficientDet D0	31.6	51.0	56.0	2.3	35.0	59.8

Our dataset achieved an mAP of **29.5** and **39.4** respectively on SSD MobileNet V2 FPNLite and EfficientDet D0. In case of both models, the mAP is a reflection of the AP found from the IOU threshold of 0.5:0.95. When the IOU threshold is 0.75, the model performs extremely well, i.e. tightly localizes the object, which contributes to the higher AP value for both the models. However, this is often a too optimistic scenario because often the coordinates of the predicted and ground truth bounding boxes might not match. Having an IOU threshold value of 0.5 also might be too pessimistic, i.e. loosely localize the object, as it will take precision values too soon. Hence a range of IOU threshold is needed which is provided by 0.5:0.95 and thus this range gives the proper mAP value for both the models. In case of varying IOU threshold, the area or the size of the bounding boxes is kept constant, which in our case is all. When the area is varied, the IOU value is kept constant and that is 0.5:0.95. Generally, with small size of the bounding boxes, the AP value is the lowest as this suggests poor performance in case of detecting small objects. As the area increases, the AP value increases as the classification performance gets better. In both the models, the mAP value lies between the value of AP found from “medium” and “large” size of bounding boxes. According to the MS COCO challenges’s evaluation document [48], small size is defined as $area < 32^2$, medium is $32^2 < area < 96^2$ and large is $area > 96^2$.

Rather than calculating the recall at a particular IOU threshold, in our case average recall is calculated over the threshold of 0.5:0.95 which summarizes the distribution of recall values. Average recall describes the area doubled under the recall vs IOU curve. Here in our case, when the area or the size of the bounding boxes is kept constant, the number of detections in the images is varied between 1, 10 and 100. Generally, when the number of detections is at most 1, the average recall value is the lowest. When the number of detections in a single image is kept constant $= 100$ and the size of the bounding box is varied, the results that are obtained are quite similar in pattern to those that are obtained from AP in the same circumstances. The average recall obtained for SSD MobileNetV2 FPNLite it 53.7 and for EfficientDet D0 is 56. Here the maximum number of detection in a single image is 100 and all area of bounding box is considered.

6.4 Mean Average Precision (mAP) Analysis

The average recall values on different parameters are used to calculate their corresponding average precision values. The average precision values of each class contribute to the overall mAP value. Comparing the achieved mAP of both models on our dataset and the original mAP values of the models trained on MS COCO dataset, we can see that our achieved mAP is significantly higher for both the models. Reason for this higher mAP can be the smaller size of our dataset compared to that of MS COCO. This mAP is obtained from IOU 0.5 : 0.95. Moreover, since our dataset is used to fine-tune the models which are pre-trained on the MS COCO dataset, the models perform better after fine-tuning. Because in our model there are few common categories of object with the MS COCO dataset. So after fine-tuning with our dataset, the AP of those common categories ("laptop", "knife", "scissors") increase. These increased AP in turn contributes to the increment of the overall mAP. The AP values for each category of object instances can be found from Table 10.

Table 10: Average precision (AP) of each object category

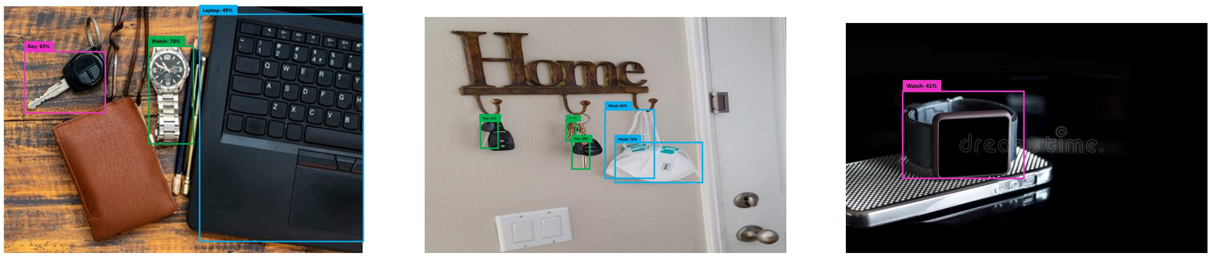
Object Category	AP of SSD MobileNetV2 FPNLite (%)	AP of EfficientDet D0 (%)
Key	26.19	32.63
Mask	28.51	36.01
Watch	24.93	30.43
Scissors	30.77	41.27
Knife	34.82	53.52
Shoe	30.14	39.24
Hairbrush	27.73	37.23
Laptop	32.37	44.87
	mAP = 29.5	**mAP = 39.4**

The “knife” object class has the highest AP value of all the object categories. Among all the object categories present in our dataset, instances for “knife”, “scissors” and “laptop” are already present in the MS COCO dataset. Number of “knife” object instances in the MS COCO dataset is the highest among those object categories that are common to both ours and MS COCO dataset. Due to the more number of instances of “knife” it achieved the highest AP on our dataset. Similarly the AP values for both “scissors” and “laptop” are also higher compared to the rest of the object classes. This is again due to these categories’ presence in the MS COCO dataset. From the rest of the object categories that are not available in the MS COCO dataset, “shoe” achieved the highest AP, the reason for which is its high number of object instances per image (almost 4). “Hairbrush” achieved an AP which is a bit lesser than that of “shoe” due to its comparatively lesser number of object instances per image. On the other hand, although the “mask” category has a lesser number of object instances per image, it still achieved a somewhat close value to that of the “hairbrush” category due to its high number of image instances present in both train and test set. The “mask” category had a very high intra-class variance due to the current availability of many different types of masks, which is why the number of images of the “mask” category is the highest in our dataset. “Key” and “watch” achieved similar AP values which are less than the rest of the object categories. The reason for “watch” having the least AP can be explained by its lowest number of object instances per image. “Key” on the other hand although had a higher number of object instances per image than “mask”, the total number of images in the former category is significantly lesser than that of the latter category which eventually contributed to the comparatively lower AP of the former category.

The inference time achieved by SSD MobileNet V2 FPNLite and EfficientDet D0 on our model is respectively 22ms and 39ms, which is the same for the original models trained on MS COCO. This inference time did not change while testing with our dataset because this inference time depends on the model's size, number of parameters, FLOP counts etc. Since none of those are changed in our work, we achieved the same inference time as the original models. Our dataset gives significantly high mAP value for both the models, hence we can say that our data distribution is well and it can be used for further works.

6.5 Test on Images

For visualization purpose, few single images were taken at random from the test set and the models were run. The predicted bounding boxes along with their confidence score is generated as output, which is shown in Figure 26.



(a) (b) (c)

Figure 26: Predictions of SSD MobileNet V2 FPNLite (a) and EfficientDet D0 (b,c)

From the output of the models we can see that in both the images the models have detected multiple objects accurately. In Figure 26b all of the objects are partially occluded by objects of same object category. We can see that separate bounding boxes were drawn among each individual objects. Similarly, in Figure 26a even though the key and watch is not occluded, the laptop is more than 50% occluded and yet the model accurately detected it. This proves the occlusion invariance of our dataset. Figure 26c on the other hand shows accurate detection with somewhat good confidence score in extremely dark lighting condition. This in turn proves the lighting invariance of our dataset.

Chapter 7

Conclusion and Future Works

In this work we have proposed a new fully labeled dataset having 5196 unique images of 8 categories for detecting few of the everyday use objects found in indoor settings in their natural contexts that are highly relevant for the visually impaired people's mobility. We, along with our recruited data annotators have spent approximately 200 worker hours just to collect and annotate our proposed dataset. We believe our proposed dataset can be a robust solution for a visually impaired person's mobility if this dataset can be used to train lightweight models that can be incorporated in handheld sensor-based or mobile-based devices. We have emphasized on finding mostly non-iconic images of objects which have significant contextual information, as well as varying viewpoints and lighting conditions. From our dataset statistics, it is proven that most of the categories have more than one object present in a single image, which indicates the presence of rich contextual information of the image. Our dataset has been used to fine-tune two widely used lightweight models pre-trained on the MS COCO dataset, namely SSD MobileNetV2 FPNLite and EfficientDet D0. Our dataset has performed well on both of these models achieving an mAP of 29.5% and 39.4% respectively. This mAP is higher than the original mAP values of these models when trained on the MS COCO dataset. which indicates a uniform data distribution in our dataset. Moreover, we can claim from this achieved higher mAP that when used to fine-tune any model that is pre-trained on the MS COCO dataset, our dataset will always generate better mAP value and thus can be used for a more robust detection. This will significantly help in case of a visually impaired user's mobility in indoor settings.

Currently a lot of progress is being made in the area of object class recognition, however, to our knowledge we have not found any such work focusing on the specific set of objects that we have incorporated in our dataset. Hence, we can safely say that our work can be extended in multiple dimensions for more robust detection of such objects. Here we present few aspects of our work which can be improved or research area where our work can contribute in:

- **More object classes** : The most obvious extension of our work is to increase the number of fully labeled object categories. Due to the limitation of financial resources, even though we wanted to, we could not incorporate more object classes as annotating them would require a larger number of data annotators. In the future, other object instances that are frequently used in indoor settings and relevant to a visually impaired person's mobility can be added to our dataset and thus making it more robust. Instances of those objects may be present in the MS COCO dataset in a comparatively small number or the objects might be entirely new. However, collecting images with meaningful context of those objects can be challenging which would require careful consideration and a creative approach.

- **Alternate annotation method** : Our dataset is annotated using the LabelImg software [42], which is a tedious and quite time-consuming task. In the circumstances of extending our dataset with new fully annotated data, an alternate annotation method needs to be generated. Semi-automated annotation method [24] can be taken into account as an alternate annotation method.

- **Combining with other indoor dataset** : Our dataset can be combined with other indoor object detection datasets like IODR [27] and/or MCIndoor20000 [22] for a complete dataset with more landmark objects. The IODR dataset contains object instances relevant in emergent situations. Having a dataset that is a combination of our work and the IODR will be a complete dataset that is useful in both emergent and non-emergent situations. This combined dataset can further be used to train models that can be further deployed in hand-held devices used by visually impaired person for navigation.

- **Navigation system for the visually impaired** : Our dataset is tested on light-weight models, using which a navigation system can be built. The system would work by giving audio feedback to the user of the object that he/she is moving towards. Incorporating depth data as an input to the system, the system might also be able to give feedback related to the distance of the object that the user is approaching. This can be immensely helpful for a visually impaired person's mobility in an indoor environment that is unknown to him/her.

- **Robot navigation** : Our dataset can also be used for robot navigation in indoor settings. In many households intelligent robots are used for doing household tasks. If a robot has a model deployed in its system that is trained on our dataset, that robot will be able to avoid certain obstacles or pick up certain objects. Although this might not be entirely relevant for a visually impaired person, this is another direction in which our work can contribute significantly.

Immense advancement has been made the area of object detection in the past decade. We believe our work would be able to make meaningful contributions to this development by providing a more robust solution to the problem of indoor object detections.

Bibliography

[1] R. R. Bourne, S. R. Flaxman, T. Braithwaite, M. V. Cicinelli, A. Das, J. B. Jonas, J. Keeffe, J. H. Kempen, J. Leasher, H. Limburg, *et al.*, "Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis," *The Lancet Global Health*, vol. 5, no. 9, pp. e888–e897, 2017.

[2] A. Arora, A. Grover, R. Chugh, and S. S. Reka, "Real time multi object detection for blind using single shot multibox detector," *Wireless Personal Communications*, vol. 107, no. 1, pp. 651–661, 2019.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[5] M. Everingham, S. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[6] M. Noman, V. Stankovic, and A. Tawfik, "Portable offline indoor object recognition system for the visually impaired," *Cogent Engineering*, vol. 7, no. 1, p. 1823158, 2020.

[7] "Tensorflow object detection api." Accessed on 21 April 2022.

[8] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[9] "Microsoft azure cloud interface. analyze an image.." Accessed on 23 April 2022.

[10] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, 2012.

[11] S. Kayukawa, H. Takagi, J. Guerreiro, S. Morishima, and C. Asakawa, "Smartphone-based assistance for blind people to stand in lines," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–8, 2020.

[12] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[14] M. A. Rahman and M. S. Sadi, "Iot enabled automated object recognition for the visually impaired," *Computer Methods and Programs in Biomedicine Update*, vol. 1, p. 100015, 2021.

[15] "Tensorflow lite object detection api for mobile and edge." Accessed on 23 April 2022.

[16] M. Awad, J. El Haddad, E. Khneisser, T. Mahmoud, E. Yaacoub, and M. Malli, "Intelligent eye: A mobile application for assisting blind people," in *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*, pp. 1–6, IEEE, 2018.

[17] "Catchoom: Craftar pro sdk." Accessed on 23 April 2022.

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[19] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492, IEEE, 2010.

[20] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 413–420, IEEE, 2009.

[21] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *2004 conference on computer vision and pattern recognition workshop*, pp. 178–178, IEEE, 2004.

[22] F. S. Bashiri, E. LaRose, P. Peissig, and A. P. Tafti, "Mcindoor20000: A fully-labeled image dataset to advance indoor objects detection," *Data in brief*, vol. 17, pp. 71–75, 2018.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[24] B. Adhikari, J. Peltomaki, J. Puura, and H. Huttunen, "Faster bounding box annotation for object detection in indoor scenes," in *2018 7th European Workshop on Visual Information Processing (EUVIP)*, pp. 1–6, IEEE, 2018.

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[26] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[27] M. Afif, R. Ayachi, Y. Said, E. Pissaloux, and M. Atri, "A novel dataset for intelligent indoor object detection systems," *Artificial Intelligence Advances*, vol. 1, no. 1, pp. 52–58, 2019.

[28] M. Afif, R. Ayachi, E. Pissaloux, Y. Said, and M. Atri, "Indoor objects detection and recognition for an ict mobility assistance of visually impaired people," *Multimedia Tools and Applications*, vol. 79, no. 41, pp. 31645–31662, 2020.

[29] X. Ding, Y. Luo, Q. Yu, Q. Li, Y. Cheng, R. Munnoch, D. Xue, and G. Cai, "Indoor object recognition using pre-trained convolutional neural network," in *2017 23rd International Conference on Automation and Computing (ICAC)*, pp. 1–6, IEEE, 2017.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[33] "Imagenet large scale visual recognition challenge 2017 (ilsvrc2017)." Accessed on 22 April 2022.

[34] J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks,” *Advances in neural information processing systems*, vol. 29, 2016.

[35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[36] J. Redmon and A. Farhadi, “Yolo9000: better, faster, stronger,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

[37] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, *et al.*, “Speed/accuracy trade-offs for modern convolutional object detectors,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7310–7311, 2017.

[38] “Tensorflow 2 detection model zoo.” Accessed on 23 April 2022.

[39] T. Berg and A. Berg, “Finding iconic images. in, cvpr, workshops,” 2009.

[40] A. Torralba and A. A. Efros, “Unbiased look at dataset bias,” in *CVPR 2011*, pp. 1521–1528, IEEE, 2011.

[41] J. Dobies, “Google image downloader.” Accessed on 21 April 2022.

[42] D. L., “Labelimg.” Accessed on 21 April 2022.

[43] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[44] “Mask dataset.”

[45] M. Tan, R. Pang, and Q. V. Le, “Efficientdet: Scalable and efficient object detection,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10781–10790, 2020.

[46] M. Everingham and J. Winn, “The pascal visual object classes challenge 2007 (voc2007) development kit,” 2009.

[47] “Intersection over union (iou).” Accessed on 22 April 2022.

[48] “Ms coco challenge evaluation document.” Accessed on 22 April 2022.