Experiment Setup

In order to conduct the experiment a powerful GPU and TPU was needed to train, validate and a custom machine learning model. In order to do test this custom based model google Colab a free resource was used. The data set was uploaded to google drive that was then accessed by Google Colab environment. The Google Colab provided Python programming capabilities with access to popular deep learning libraries.

The model was trained using individual validation dataset to evaluate its performance, efficacy and generalization ability. To assess the performance various metrics such as accuracy, precision, recall, F1-score, mean average precision (Map) were employed. Google drive was used as a primary storage facility to store pre-processing and training data to reduce manual intervention for storage. The experimental data such as weights, evaluation metrics, and visualizations were redirected to be sored on the google drive for easy and secured analysis for future use.

5.1 Advantages of Google Colab for Experimentation

Google Colab offers free computing resources with certain limitations, making it affordable for researchers and developers. The users can leverage GPUs and TPUs for quicker calculation, enabling faster model training and experimentation. Another advantage of the colab notebooks is that the notebook can easily be shared and collaborates. This fosters teamwork and knowledge sharing.

5.2 Dataset Preparation

A dataset was prepared and classified into two groups, of five classes consisting of objects and five classes of people.

Object Classes

Images of the object classes the images were sourced from Google Open Images Dataset and converted into the YOLO (You Only Look Once) format using the OIDv4 Toolkit. This Python based package facilitated the extraction of specific parts of the Open Image dataset to create custom object dataset to create object such as Knife, Handgun, Bottle, Axe, and Hammer.

First, a repository clone was used to download the OIDv4 Toolkit.

This required a few steps: installing Anaconda and setting up and activating a virtual e nvironment came first.

After that, Git was installed to make repository cloning possible.
The repository was cloned into the specified directory using the following command:

git clone https://github.com/EscVM/OIDv4_ToolKit.git

Next, the directory was changed to the cloned OIDv4_ToolKit folder, and the necessary dependencies were installed using:

pip install -r requirements.txt

Pandas, numpy, awscli, urllib3, tqdm, and opencv-python were among these dependencies. After the setup was finished, we downloaded the pictures for the classes we had designated. The following command was run in order to accomplish this:

python main.py downloader --classes Knife Handgun Bottle Axe Hammer --type_csv train --multiclass 1 --limit 200

The classes to be downloaded are specified by the `classes` option in this command, a nd the `--type_csv train} option indicates that training data is being downloaded.
To guarantee that every image is downloaded into a separate folder, use the `--multiclass 1} option.

A limit of 200 photos per class is imposed by the `--limit 200} argument.
Thus, the goal was to download a maximum of 1000 photos using five classes.

The actual number of photographs acquired, however, may have been slightly lower b ecause some may have been deleted from the original website.

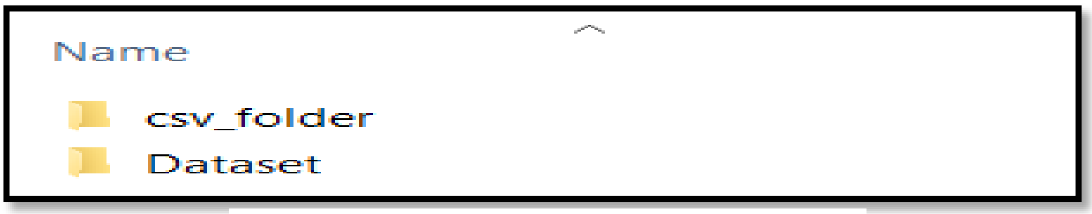
This structured approach enabled us to efficiently prepare a robust dataset for our experimental setup, ensuring the integrity and organization necessary for effective object detection model training. The number of object images downloaded are shown in Table no 5.1.

Table 5.1 : Class and no of images downloaded

CLASS	NO. OF IMAGES
Knife	200
Handgun	200
Bottle	200
Axe	115
Hammer	114

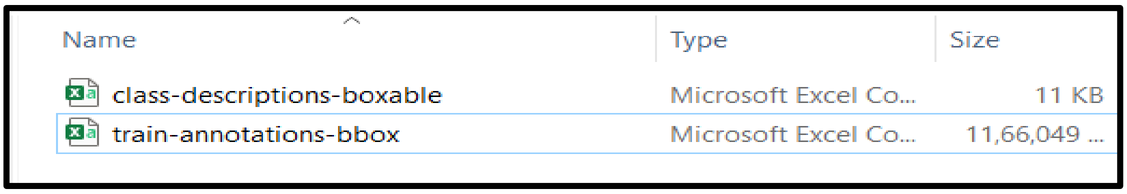
Pre-existing labels and bounding boxes that identified the objects in each image were included with these pre-annotated photographs. This removed the requirement for manual labelling, which may be a labour- and time-intensive procedure. In manual labelling, each image is reviewed by a human annotator who then manually draws bounding boxes around objects of interest and labels them with the appropriate class. It takes a lot of work to complete this process.

We expedited the process of preparing our dataset by utilizing pre-annotated photos, which guaranteed high-quality and consistent annotations. Furthermore, because the annotations were consistent and applied to every image, the usage of pre-annotated data made the training process for our object detection model more effective and dependable. Maintaining consistency is essential for training reliable and accurate models because it guarantees that the model gains knowledge from appropriately labelled input, which enhances its performance and capacity for generalization.



Name
csv_folder
Dataset

Fig. 5.1 : csv_folder and Dataset folder created

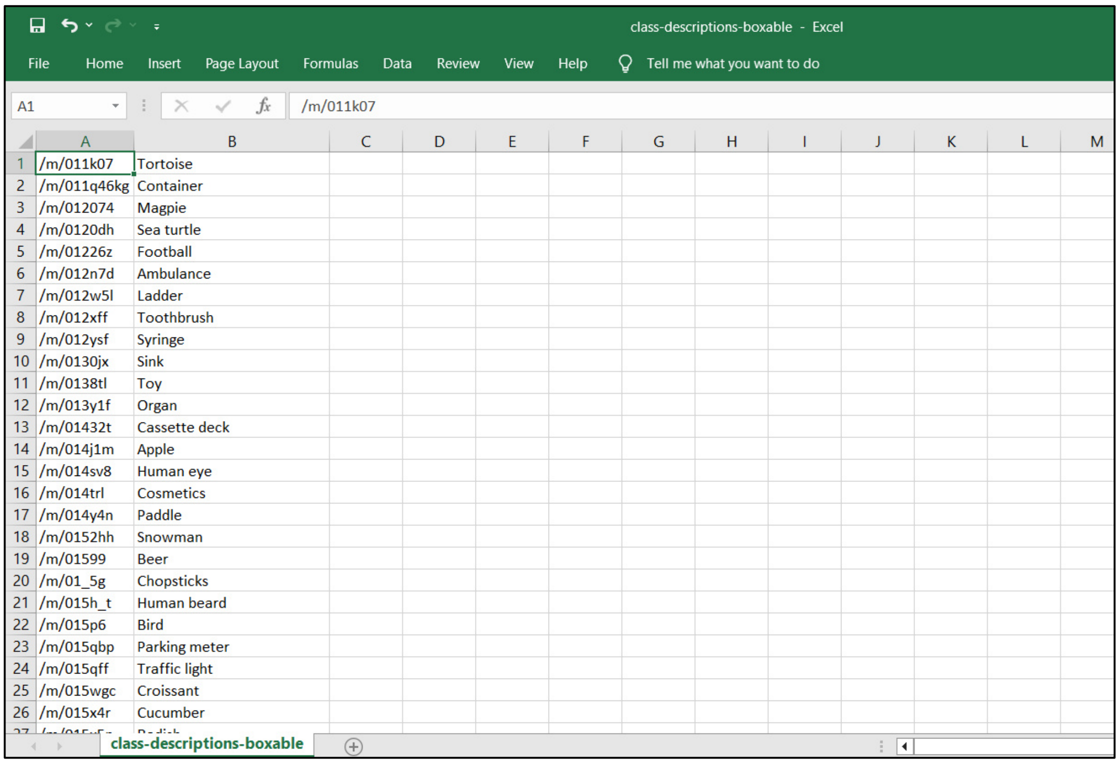


Name	Type	Size
class-descriptions-boxable	Microsoft Excel Co...	11 KB
train-annotations-bbox	Microsoft Excel Co...	11,66,049 ...

Fig. 5.2 : class-descriptions-boxable.csv and train-annotations-bbox.csv files created in csv_folder

After executing the above steps, 2 files were created in csv_folder i.e. class-descriptions-boxable.csv and train-annotations-bbox.csv as displayed in figure 5.1 and 5.2.

The class-descriptions-boxable.csv displayed in figure 5.3 contains the name of all the classes with their corresponding 'LabelName' and the validation-annotations-bbox.csv file contains one bounding box (bbox for short) coordinates for one image, and it also has this bbox's LabelName and current image's ID from the validation set of OIDv4.


class-descriptions-boxable - Excel
File
Home
Insert
Page Layout
Formulas
Data
Review
View
Help
Tell me what you want to do
A1
/m/011k07
class-descriptions-boxable

	A	B
1	/m/011k07	Tortoise
2	/m/011q46kg	Container
3	/m/012074	Magpie
4	/m/0120dh	Sea turtle
5	/m/01226z	Football
6	/m/012n7d	Ambulance
7	/m/012w5l	Ladder
8	/m/012xff	Toothbrush
9	/m/012ysf	Syringe
10	/m/0130jx	Sink
11	/m/0138tl	Toy
12	/m/013y1f	Organ
13	/m/01432t	Cassette deck
14	/m/014j1m	Apple
15	/m/014sv8	Human eye
16	/m/014trl	Cosmetics
17	/m/014y4n	Paddle
18	/m/0152hh	Snowman
19	/m/01599	Beer
20	/m/01_5g	Chopsticks
21	/m/015h_t	Human beard
22	/m/015p6	Bird
23	/m/015qbp	Parking meter
24	/m/015qff	Traffic light
25	/m/015wgc	Croissant
26	/m/015x4r	Cucumber

Fig. 5.3 : class-descriptions-boxable.csv

5.3 People Classes

For the people classes, images were downloaded from Google and processed further. The classes included Bill Gates, Elon Musk, Marilyn Monroe, Leonardo DiCaprio, and Will Smith. For each classes the steps such as auto-orienting the pixel data striped the EXIF orientation metadata. The images were resized to 640x640 pixels with stretching to fit the dimensions. The Images were augmented by applying three 90-degree rotations, along with a random rotation of -15 and +15 degrees horizontally and vertically. To expose the model to varied scenario. For bill gates 61 images were pre-processed and augmented to 130 images; For the Elon musk class 151 images were augmented using 63 original images; with an initial set of 63 images of Marilyn Monroe the dataset was expanded to 144 images after augmentation; After pre-processing and augmentation of 62 images of Leonardo DiCaprio the dataset stretched to 148 images; For the final class of people images a set of 62 images of will smith were pre-processed and augmented to expand the dataset to 148 images.

We augmented the dataset of images featuring people. Implementing image augmentation techniques expanded the dataset and helped reduce overfitting. Cleaning and augmenting image data can significantly improve the model's performance.

Each of these steps ensured that the dataset was robust and suitable for training object detection models.

Labelling of Images

The images were labelled by a user-friendly platform- **Roboflow.** This powerful platform helped in efficiently labelling the images in order to get precise and consistent annotations across all images. The advanced features such as automatic annotation suggestions greatly assisted in fastening the labelling process with great efficacy. Each object with in the image was carefully annoted to match the resulting labels making it fit for training high-performance object detection models.

Also, to ensure transparency and facilitate easy access to the dataset the links to all images were maintained in an excel file, serving as a comprehensive reference, containing the URLs of each image.

This organized approach not only aids in tracking the origin of each image but also makes it easier for other researchers to verify and utilize the dataset. By providing this level of detail and accessibility, we ensure that our dataset preparation process is both transparent and reproducible, supporting the integrity and utility of our research.

Table 5.2 : Before & After Pre-processing People Image Dataset

CLASS	No. of instances Before Pre-processing	No. of instances After Pre-processing and Augmentation
BillGates	61	130
ElonMusk	63	151
MarilynMonroe	63	144
LeonardoDicaprio	62	148
WillSmith	62	148

This table 5.2 illustrates the increase in the number of instances for each class of people after applying the pre-processing and augmentation steps to the initial set of images. The augmentation process significantly enhanced the dataset, making it more suitable for training robust object detection models.

Table 5.3 : Class instances and their no. of annotation

CLASS	CLASS INSTANCES	NO. OF ANNOTATION
Knife	200	258
Handgun	200	254
Bottle	200	509
Axe	115	148
Hammer	114	139
BillGates	130	130
ElonMusk	151	149
MarilynMonroe	144	144
LeonardoDicaprio	148	148
WillSmith	148	148

This table 5.3 details the number of instances and corresponding annotations for each class in the dataset. The variations in the number of annotations reflect the complexity and distinctiveness of each class, contributing to a comprehensive dataset for object detection model training.

Table 5.4 : Dataset Summary

	With Pre-processing and Augmentation
Total Images	1550
Classes	10
Unannotated	0
Training Set	1085(70%)
Validation Set	310(20%)
Testing Set	155(10%)
Annotation	2032(.3 per image (average))

This table 5.4 provides a summary of the dataset used for training, validation, and testing. The dataset consists of 1550 images across 10 classes, all of which have been annotated. The distribution of images ensures a balanced split for training (70%), validation (20%), and testing (10%) purposes. On average, each image has 0.3 annotations, leading to a total of 2032 annotations.

This dataset, was further trained on the three versions of YOLO (You Only Look Once) object detection model: YOLOv5, YOLOv7, and YOLOv8. Each model underwent due training to use the pre-processed and augmented dataset to ensure consistency and robustness in the training process.

Training:

- **YOLOv5**: firstly, YOLOv5, known for its balance between speed and accuracy was trained.
- **YOLOv7**: As an improved version YOLOv7 incorporates advanced techniques to enhance detection accuracy and speed.
- **YOLOv8**: The Dataset was trained using the latest iteration YOLOv8, to leverage on latest advancements.

5.4 Evaluation Metrics

To assess the performance of our model using the created dataset, we computed Precision, Recall, and Average Precision (AP). These metrics help in evaluating the classification accuracy, detection ability, and overall effectiveness of the model. The following formulas were used to calculate these metrics:

Precision (P)

Precision measures the accuracy of the classification by determining the ratio of correctly identified positive instances to the total number of positive instances identified. It indicates how many of the identified instances are actually relevant. The formula for Precision is:

$$Precision = \frac{True\ Positive}{(True\ Positive + True\ Negative)}$$

Recall (R)

Recall measures the ability of the model to correctly identify all relevant instances within the dataset. It is the ratio of correctly identified positive instances to the total number of actual positive instances. The formula for Recall is:

$$Recall = \frac{True\ Positive}{(True\ Positive + False\ Negative)}$$

F1-Score

The F1-Score provides a balance between Precision and Recall by considering both metrics in its calculation. It is the harmonic mean of Precision and Recall, making it a more accurate measure than accuracy alone, especially in cases of imbalanced datasets. The formula for the F1-Score is:

$$F1 - score = \left(\frac{Recall^{-1} + Precision^{-1}}{2}\right)^{-1}$$

where:

- True Positives (TP) are positive samples classified correctly.
- False Positives (FP) are negative samples classified incorrectly.
- False Negatives (FN) are positive samples classified incorrectly.

Average Precision (AP) and Mean Average Precision (MAP)

Average Precision (AP) provides the precision of the model across different recall values. It is a weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight. Mean Average Precision (MAP) is the average of the AP values for each class, providing a single performance measure for object detection models.

Frames Per Second (FPS)

Frames Per Second (FPS) is a crucial speed performance metric that indicates the number of images the model can process per second. Higher FPS values are desirable as they reflect the model's efficiency and speed in real-time applications.

To evaluate the performance of our model, we computed several key metrics: Precision (P), Recall (R), F1-Score, Average Precision (AP), Mean Average Precision (MAP), and Frames Per Second (FPS). Precision measures the accuracy of positive predictions, while Recall assesses the model's ability to identify all positive instances. The F1-Score provides a balanced measure between Precision and Recall. AP evaluates the precision across different recall thresholds, and MAP summarizes this performance across all classes. FPS indicates the model's processing speed in images per second. Together, these metrics offer a comprehensive assessment of the model's accuracy, detection capability, and efficiency.