RETINA: Real Time Speech Activated Assistant

Ankita Kundra
301406701

Anuj Saboo
301378525

Rishabh Jain
301300000

Abstract

This project presents an approach for enabling a human to talk to a system and instruct it to detect an object in a live video feed which aims at expanding the possibilities with Assistive Robots and Unmanned Aerial Vehicles(UAVs). The work demonstrated involves threaded programming to enable both video and speech inputs working simultaneously with the model to classify the object. The proposed approach uses state of the art YOLO(You Only Look Once) v3 model to detect objects passed as speech commands to the system. Using a wakeup command through speech we are able to enable a conversation with the system to continuously detect objects in a live video feed achieved through the webcam. We use the model trained on the COCO dataset to detect various objects in our surroundings. In addition to this, we train our own model with various objects not present in the pre-trained dataset to achieve the same objective. The application can run both models based on a parameter of model choice passed during the execution.

1 Introduction

Humans are constantly finding new ways to make life easier and supposedly better. It is undoubtedly true that technology is an important part of our daily lives. Robotics devices are being used to improve the independence and quality of life. Assistive robots have found their way from the sealed location in factories to homes and working spaces. They are able to autonomously perform different tasks to cater to a variety of services to improve the standard of living. Despite the effort involved in this ever-growing field, the progress has been relatively slow as of date. We have set forth to develop an assistant that listens to your command, detects and recognizes objects from the visual inputs such as cameras in an ordinary environment and produces a speech-based response upon successful detection of objects [1]. Vision is a primary cue in this setting and the problem of detecting and recognizing objects efficiently and accurately still remains a significant challenge when it comes to real-time settings. Apart from having lots of objects in the images, the reasons for this difficulty are to be found in issues such as their interactions and occlusions [2], thereby making object detection a challenging task. The task of speech recognition still remains an issue due to interference of noise. Relevant parameters were tuned but much more can still be done to achieve a better translation in noisy environments.

2 Approach

2.1 Dataset

For the dataset, we tried to build our dataset by scraping Google and other sources. Since the number of images was limited, we resorted to Google's Open Source image dataset Open Images V4 [3]. The Open Images V4 dataset contains 15.4M bounding-boxes for 600 categories on 1.9M images and 30.1M human-verified image-level labels for 19794 categories.

We have also used the pre-trained weights trained on the COCO dataset. It contains the object detection dataset with 80 classes, with 80,000 training images and 40,000 validation images.

We listed a few classes that could be useful in detecting objects in Robots and Drones. Assistive robots might be helpful in detecting objects like toys, vegetables, coin, mug and helmets whereas drones will be useful working in a dangerous location to perceive handguns, missiles, tanks, and weapons. Figure [1].

Figure 1: Frequency Distribution of Classes in the Dataset

2.2 Data Preprocessing

Our initial quest was to build a detector to find everyday objects like sunglasses, calculators, etc. We attempted building a custom dataset by web scraping using Beautiful Soup to access images from Google. We manually labeled the images to generate their bounding boxes. However, the results of training this model were poor and we realized that we would need a bigger dataset to accomplish the goal.

Hence, we shifted our focus to Open Images V4 dataset which contains images and bounding box annotations for these images. As a whole, the dataset is more than 500GB which was impractical to download. Hence we used a python script to download the data for the specific classes we needed to train our model from the S3 bucket. The download rate was slow but essential for our dataset build requirement.

3 Experiments

3.1 Application Architecture

The designed application supports working with weights of models trained on Coco Dataset and our own custom dataset. Through threaded programming and modification of global variables on actionable sequences, we enable the system to accept speech and video inputs and change the state of execution in the two media streams. Hence, the system expects to perform the task of object detection after the wakeup command is issued. Figure [2]

The object detection is carried out by the YOLO v3 model which has speed advantages over the other object detection algorithms and hence meets our requirement to work in a real time setting. The earlier methods such as R-CNN and it's variations used a pipeline(multiple steps) to predict the bounding box and class probabilities. However in YOLO the bounding box and class probabilities are found in a single network pass making it much faster. It was embarrassing for the publishers during a live demonstration when YOLO v2 detected the EXIT gate as a toilet. Hence, they wanted to focus on improving model accuracy even at the cost of losing some speed. YOLO v3 uses a variant of Darknet, which originally has 53 layer network

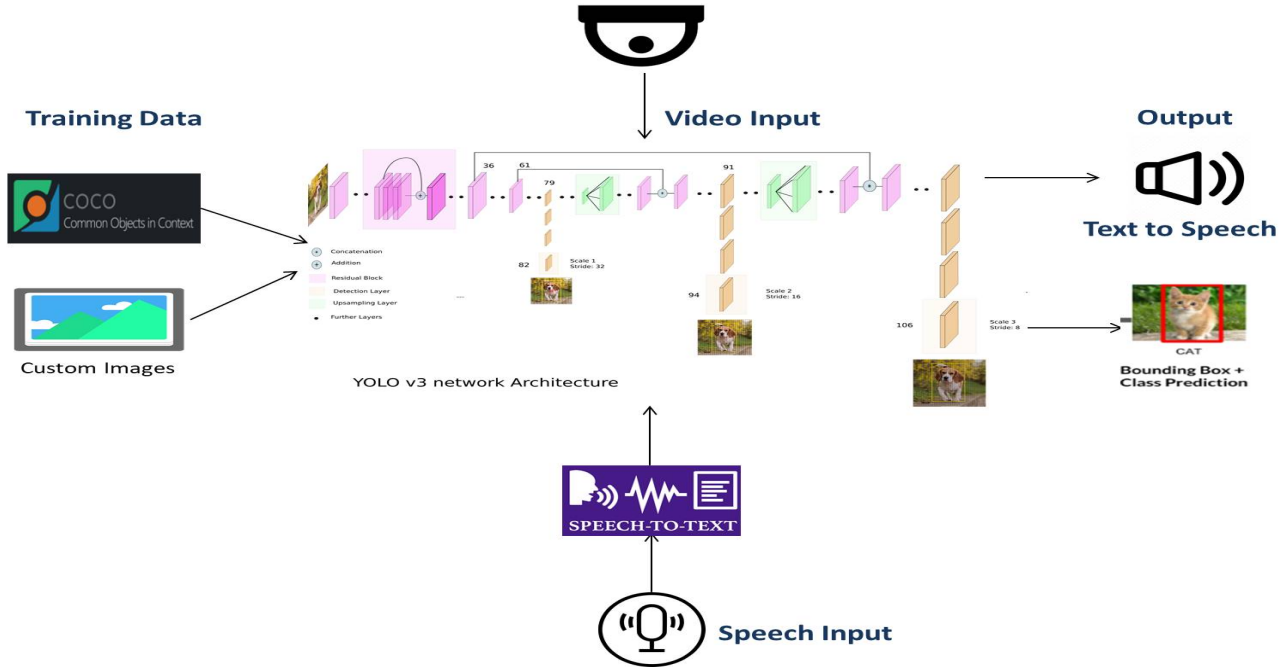

Figure 2: Application Architecture

trained on Imagenet. For the task of detection, 53 more layers are stacked onto it, giving us a 106 layer fully convolutional underlying architecture for YOLO v3 [4] for detection compared to it's former version but an increase in accuracy.

3.2 Training

We used Darknet [5], a framework to train Neural Networks to train our model on the custom dataset. In order to achieve this, we had to make numerous changes in the base yolov3-cfg file to work with our target. Since our objective is to train our model on new objects, we can gain computational advantages through **Transfer Learning** [6]. We use a pre-trained model approach of transfer learning and select the source model as darknet53.conv.74. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in the skill that they provide on related problems.

During training, we are able to plot the loss curves to decide on the criteria of how far we need to train our model. Here, we benefit from transfer learning and see our initial high loss dropping quickly. Figure [5]

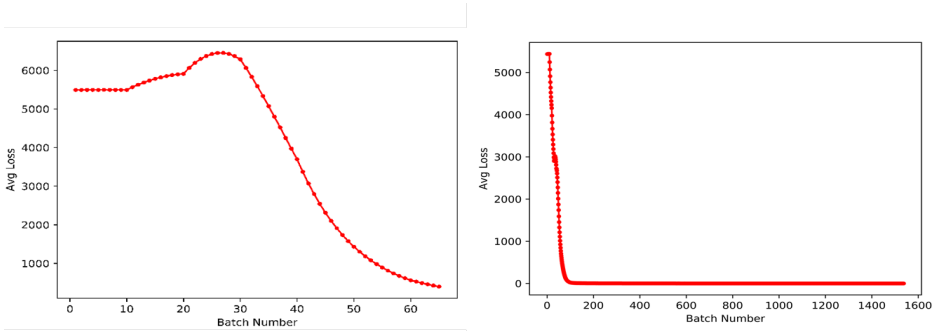

Figure 3: Average Loss over Iterations

3.3 Model Evaluation

We used the performance metric of IoU (Intersection over Union) [7] which is the measure of the overlap of the two regions. In our case we compare the ground truth bounding box available in the Open Images V4 dataset against the prediction bounding box generated by our model.

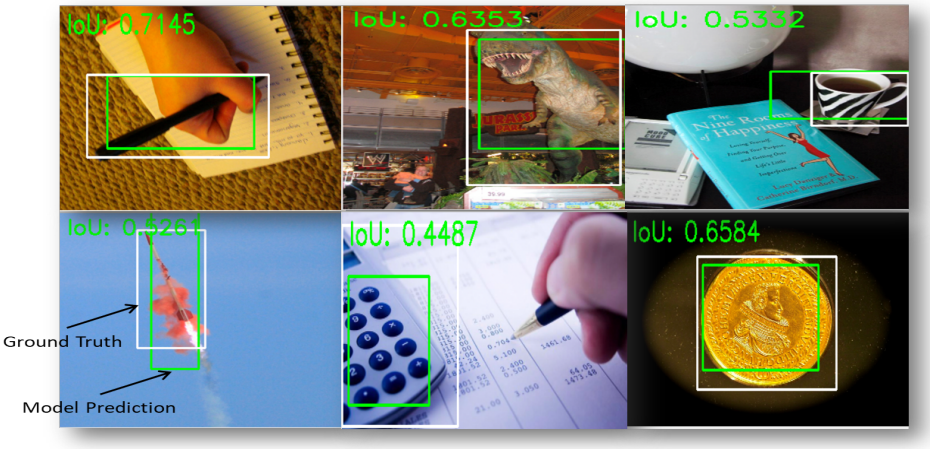

Figure 4: IOU Comparison between Ground Truth and Prediction

We also calculated mAP(Mean Average Precision) for a subset of test set available in the Open Images V4 dataset. The results show the average precision on the test set along with individual class level results.

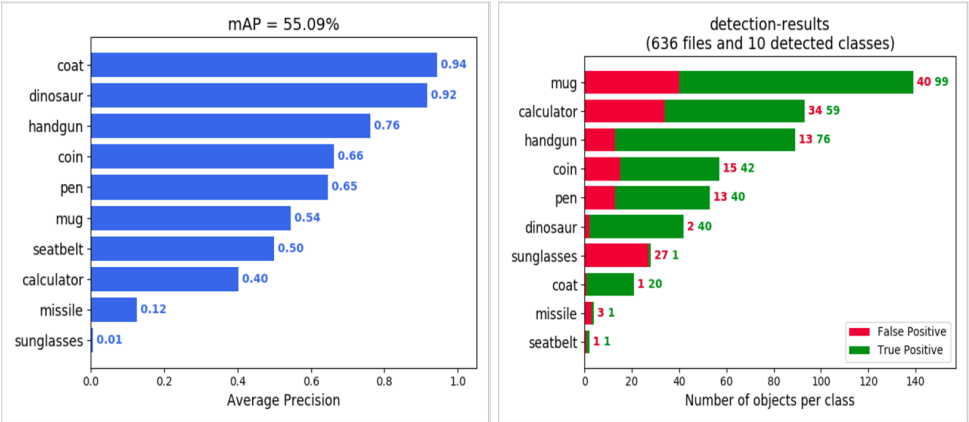

Figure 5: Average Mean Precision and Detection Results per Class

DataSet	mAP
COCO	57.9
Custom Data	55.09

3.4 Custom Configuration

Our mission to detect train the model on a dataset of 10 classes required certain parameter changes in the YOLO v3 cfg file. More specifically, we had to change the number of classes to 10 and accordingly adjust the filters in the convolutional layers as per the below equation

$$Filter = B * (5 + C)$$

Here, B = 3 and C = number of classes(10) gave us the value of 42 for the filters.

3.5 Speech

With humans reaching higher levels of robotic feature design with Sophia(humanoid robot), it is important for us to integrate the speech component in our application to transform the mundane machine feel into an interactive conversation. This would guarantee a more comfortable environment to work in the settings of restaurants where Robot waiters can perform the basic tasks and interact with clients.

3.5.1 Speech Processing

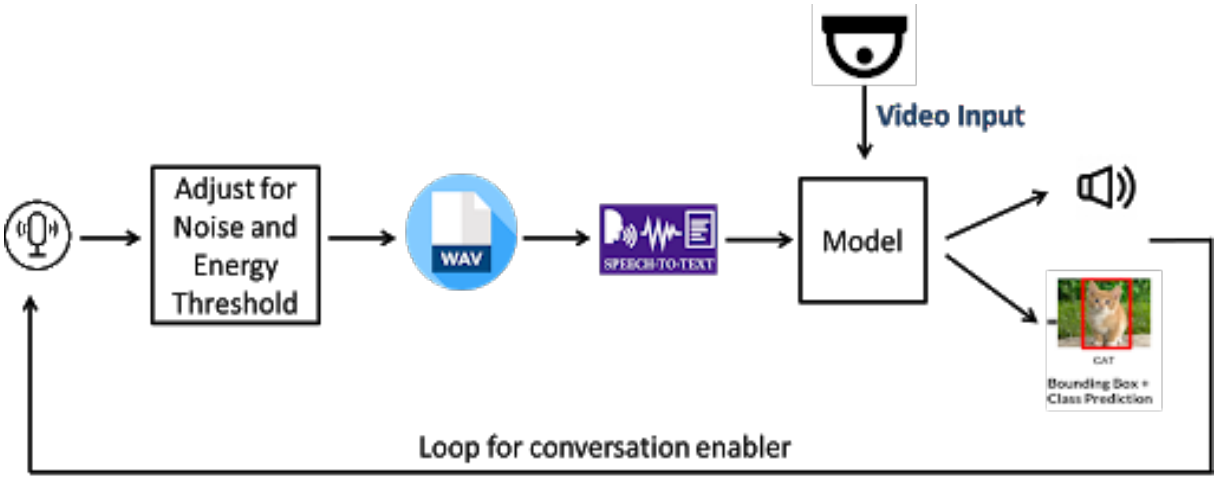

Figure 6: Workflow for Speech Handling

To make this possible, the system accepts Speech Input through the attached microphone [8]. To ensure continuous listening and processing, we program the system to generate and process sound from a WAV file in order to interpret and perform the desired action through speech(wakeup or detect object). The underlying workflow to enable this functioning is described below:

3.5.2 Speech Feedback

A conversation cannot be one way and hence it is important for the system to provide a speech feedback. This is done using pyttsx3 package in python which translates text into speech and provides tunable parameters for the desired playback output. Hence, whenever the system does successfully detect the commanded object in the surrounding, it would provide a speech feedback of the object found. Next, the user can continue to ask the system to find further objects and hence a conversation like below is achieved.

4 Results

Figure [8] shows the test data predicted on the model trained on COCO and our custom dataset. The prediction were filtered through the non maximum suppression and shows the associated label and confidence score on the input frame.

5 Conclusion and Future Work

We are successfully able to run the application for models on Pre-trained COCO dataset and our own custom trained model. The conversation experience built around audio processing is a

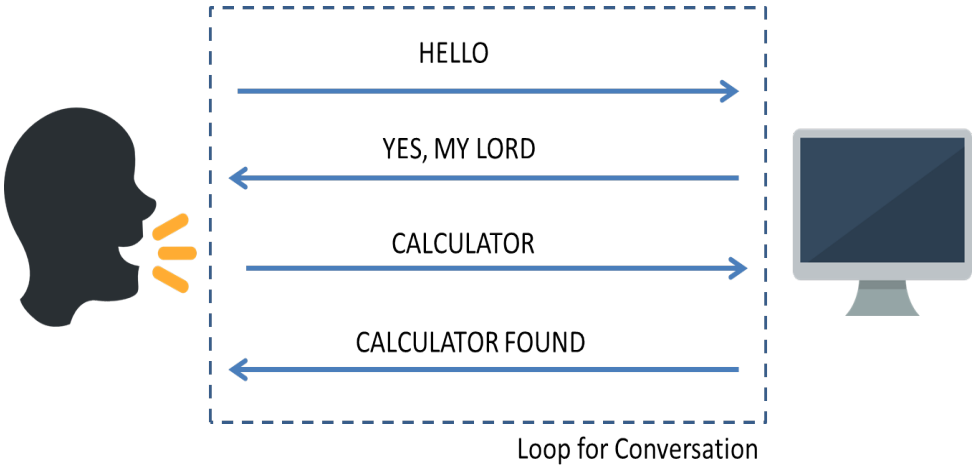

Figure 7: Conversation Model

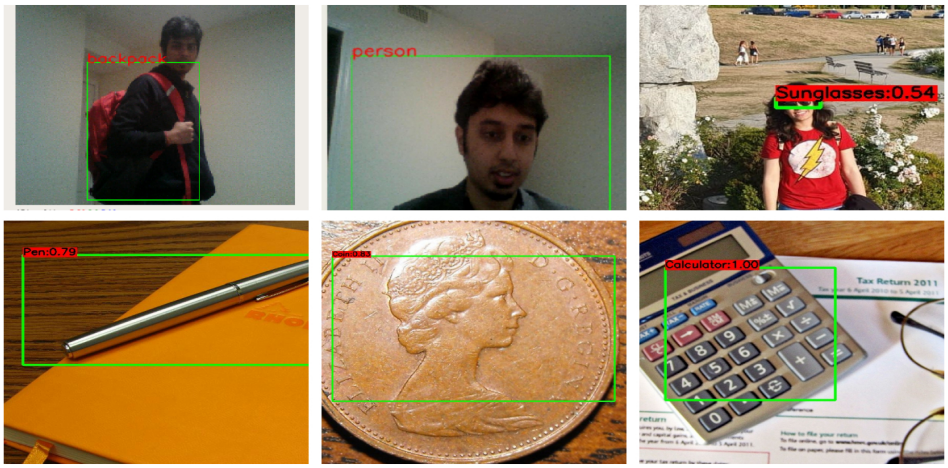

Figure 8: Test data prediction

smooth experience for user and adds relevance for building on top of the application to support Service Robots for human assistance.

The future work is gathered around:

- Larger custom dataset to train our own model to increase classification accuracy
- Designing an android app to use the phone's microphone and camera to run the application
- Tune parameters related to speech processing to improve human to machine conversation in noisy environments

6 Contribution

Tasks	Member
Web Scraping Dataset	Ankita, Rishabh
Bounding Box Labeling	Anuj
Darknet Configuration	Ankita, Anuj
Open Images V4 Dataset Preparation	Ankita, Anuj
YOLO Configuration	Rishabh
Training on Custom Dataset	Ankita, Anuj, Rishabh
Error Metrics IOU and MAP	Rishabh
Threaded Programming	Ankita, Rishabh
Video Input using OpenCV	Anuj
Speech Input Translation	Ankita
Wav File and Audio Processing	Rishabh
Speech Feedback	Anuj
Running model on COCO dataset	Ankita, Anuj
Poster	Ankita, Anuj, Rishabh
Report	Ankita, Anuj, Rishabh

References

[1] Zhong-Qiu Zhao, Peng Zheng, Shou tao Xu, and Xindong Wu. Object detection with deep learning: A review, 2018.

[2] Angel P. del Pobil Ester Martinez-Martins. Object detection and recognition for assistive robots. *IEEE Robotics Automation Magazine*, 2016.

[3] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale, 2018.

[4] Ali Farhadi Joseph Redmon. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[5] Joseph Redmon. Darknet: Open source neural networks in c. `http://pjreddie.com/darknet/`, 2013-2016.

[6] Joseph Redmon. A gentle introduction to transfer learning for deep learning. `https://machinelearningmastery.com/transfer-learning-for-deep-learning/`, 2017.

[7] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression, 02 2019.

[8] Anthony Zhang. Speech recognition pypi. `https://pypi.org/project/SpeechRecognition/`, 2017.