OBJECT DETECTION CONVERT OBJECT NAME TO TEXT AND TEXT TO SPEECH

Shaik Rahima,Y.G.V.Viswath,Puvvada Sai Vamsi,U.Y.Vamsi Krishna,Mr.K.Bhasker
Assistant Professor,
Department of Computer Science Engineering,
Seshadri Rao Gudlavalleru Engineering College,
Gudlavalleru, Andhra Pradesh - 521286

ABSTRACT

Lately, with advancements in object recognition technology, there's been a drive to merge Speech Recognition and Object Detection into a unified model. This project aims to combine both functionalities: taking a speech command as input, processing it, and subsequently identifying a specified object within a set of images. This approach harnesses the capabilities of Google Speech Recognition and YOLO (You Only Look Once) object detection models, leveraging frameworks like Darknet and OpenCV. The outcome is the detection of the targeted object, visually represented by a rectangular box in the images.

INTRODUCTION

The training process will take place on the cloud using the Darknet framework with the assistance of a free GPU, resulting in the generation of weights and configuration files. Subsequently, the YOLO object detection model is employed. This model evaluates the confidence score for each pixel in the image. If the object's proximity to the trained image score is confirmed, a bounding box is created, indicating the detection of the desired object.To implement the Speech Recognition and Object Detection modules,

users are required to import specific packages, including SpeechRecognition and PyAudio for speech- related functionalities, and OpenCV for computer vision operations on images. Dataset training using the Darknet Framework is integral to the model's development. This approach utilizes the Google Speech Recognition model and the YOLO object detection model, a Deep Neural Networks algorithm.

Deep Learning

Deep learning stands as a branch of machine learning that instructs computers to process inputs by traversing multiple layers, enabling them to learn and make predictions or classifications. These inputs can take the form of images, text, or sound. The fundamental inspiration for deep learning lies in emulating the human brain's way of processing information, aiming to replicate its extraordinary capabilities. Maintaining the Integrity of the Specifications .

Feedforward And Feedback Networks

A feedforward network comprises input, output, and hidden layers, allowing signals to flow in a

singular direction—forward. The input data is initially processed within a layer where computations take place. Each processing element within this layer computes based on the weighted sum of its inputs. These calculated values then serve as the new inputs for the subsequent layer, continuing the feed- forward progression. This sequence proceeds across all layers until reaching the final output. Feedforward networks find frequent application in domains like data mining due to their ability to process information sequentially and derive meaningful outputs.

A feedback network, exemplified by a recurrent neural network, incorporates pathways for feedback, facilitating bidirectional signal transmission through loops. These networks allow for connections between neurons in various configurations. The presence of loops transforms this network into a nonlinear dynamic system, continually evolving until it attains a state of equilibrium. Feedback networks are commonly applied in optimization problems, where the network seeks the most optimal configuration of interconnected factors.

Weighted Sum

The inputs to a neuron originate from either features w within a training set or outputs stemming from neurons in a preceding layer. Each connection linking two neurons involves a unique synapse, carrying a specific weight. To traverse from one neuron to another, the signal must traverse the synapse, bearing the respective "toll" represented by its weight .

- Subsequently, the neuron employs an activation function on the total of the weighted inputs received from each incoming synapse. This resultant value is then transmitted to all neurons within the subsequent layer. When discussing weight updates within a network, the focus lies on adjusting these synapse-associated weights.
- A neuron's input is essentially the summation of weighted outputs from all neurons in the preceding layer. Each input is scaled by the weight linked to the synapse connecting it to the current neuron. For instance, if the previous layer comprises three inputs or neurons, every neuron within the current layer will possess three unique weights—corresponding to each synapse.
- In essence, the output of a node is determined by the activation function it employs.

Activation function

The activation function, also known as the transfer function, converts input signals into output signals by mapping them onto a specific range, such as 0 to 1 or -1 to 1. It serves as an abstraction mirroring the rate of action potential firing within a cell, essentially representing the probability of the cell firing. In its simplest form, the function is binary, resulting in either a "yes" (neuron fires) or a "no" (neuron doesn't fire) output. This binary output is typically represented as 0 or 1, akin to an on/off or yes/no scenario. However, activation functions can also produce outputs within a continuous range. For instance, suppose you're utilizing a function that maps a range between 0 and 1 to determine the likelihood of an image depicting a cat. An output of 0.9

would indicate a 90% probability that the image indeed features a cat.

Open CV

OpenCV, short for Open Source Computer Vision Library, stands as an open-source software system encompassing computer vision and machine learning capabilities. Its inception aimed to establish a unified framework for computer vision applications and expedite the integration of machine perception into various industrial products. The BSD licensing of OpenCV facilitates easy adaptation and modification by businesses.

This comprehensive library boasts 2500 optimized algorithms, comprising both conventional and cutting-edge computer vision and machine learning techniques. These algorithms serve a multitude of purposes, including facial detection and recognition, object identification, human action classification, camera movement tracking, and monitoring moving objects.

YOLO (You Only Look Once)

YOLO (You Only Look Once) represents an efficient method for object detection, employing a unique approach. It swiftly processes the entire image just once, navigating through the network to detect objects, hence its name. Renowned for its speed, YOLO v3 architecture incorporates multiple output layers that provide predictions.

Object detection in YOLO involves the creation of bounding boxes around identified objects, displaying the recognized items within the image or video stream. YOLO v3 utilizes Darknet-53, a 53-convolution architecture, for feature extraction.

These models can be extended for speech-to-text translation, enabling the identification of objects based on spoken commands. In this proposed system, users can verbally instruct the system to detect specific objects from a multitude of options displayed on the screen.

A notable feature of this system is its adaptability in accurately identifying objects, even when trained on images of varying sizes and colors, showcasing its robustness in diverse scenarios.

FLOWCHART

Figure 1. Flowchart

Object Detection : Object Detection, within the realm of Computer Vision, involves identifying semantic objects in images or videos by outlining them with bounding boxes. Once annotated, this information can be converted into voice responses, providing basic positional details of the detected objects within the person's or camera's view.

Training Data : The model undergoes training using the Common Objects in Context (COCO) dataset. You can explore the labeled images from this dataset through the provided link, offering a fascinating insight into the annotated data.

Model : The model employed is the You Only Look Once (YOLO) algorithm, utilizing a modified version of the intricate Convolutional

Neural Network architecture known as Darknet. While the system predominantly uses an improved YOLO v3 model, the explanation provided will focus on the original YOLO algorithm. Additionally, the Python cv2 package offers a method to configure Darknet based on settings specified within the yolov3.cfg file.

Block Diagram

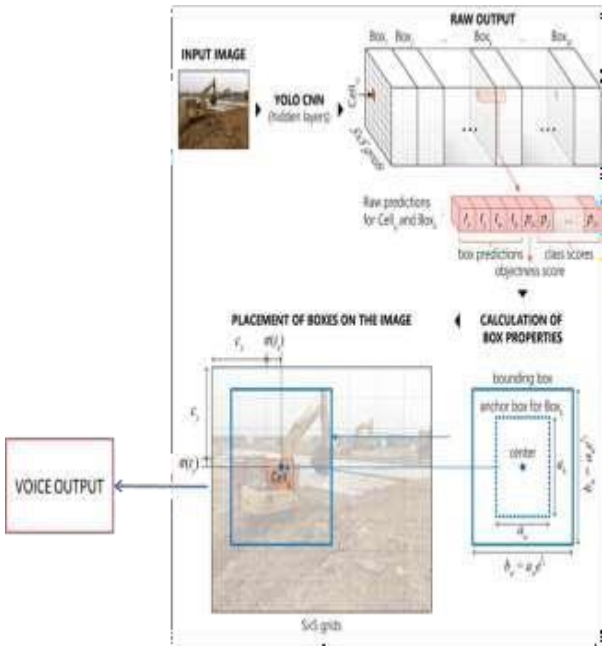


Figure 2 Block Diagram

Input Data

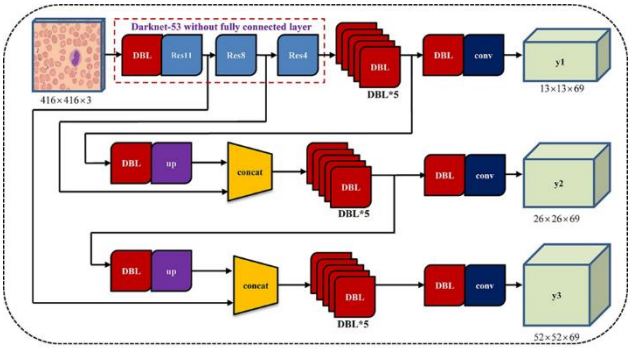
The input data for this process involves utilizing the webcam to provide images at a rate of 30 frames-per-second to the trained model. To expedite processing, the system can be configured to analyze every other frame, enhancing the overall speed of the operation.

API

For each frame, the detected objects' class prediction will be presented as a string label, such as "cat." Alongside these class predictions, the system will gather the coordinates of the objects within the image. To enhance the description, the positions—such as "top," "mid," or "bottom," and "left," "center," or "right"—will be appended to the class prediction "cat." This combined information will then be sent to the Google Text-to-Speech API using the gTTS package to generate an audio description.

OUTPUT

In addition to obtaining object coordinates, we'll extract the bounding box coordinates for each detected object within our frames. These bounding boxes will be overlaid onto the detected objects, resulting in a video playback stream



showcasing the detections. To provide voice feedback, we'll schedule a vocal response on the first frame of every second (as opposed to the continuous 30 fps rate). For instance, the feedback might include descriptors like "bottom left cat," indicating the detection of a cat in the bottom-left portion of the camera's view.

YOLO V3 Architecture

Figure 3 Architecture

Understanding the YOLO algorithm

In the past, object detection primarily relied on classification- based models employing techniques like localization, region-based

classification, or methods such as the sliding window approach. These methods selected high-scoring regions within an image for detection, which often proved to be time-consuming processes.

Contrary to earlier methods, YOLO operates on a regression- based approach. It swiftly predicts classes and bounding boxes for the entire image in a single pass of the algorithm—essentially, a single comprehensive examination of the image's pixels. This method ensures that predictions are influenced by the global context within the image, allowing for more informed detections.

DFD :

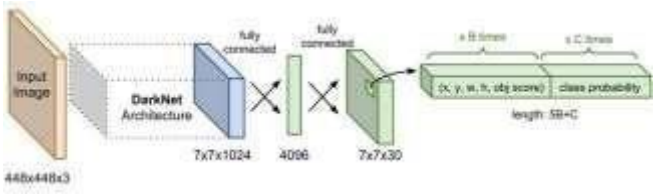


Figure 4 DFD

Training Time :

```
29  suitcase
30  frisbee
31  skis
32  snowboard
33  sports ball
34  kite
35  baseball bat
36  baseball glove
37  skateboard
```

In the context of labeling objects, 'C' represents the class index assigned to the specific object we aim to identify. For instance, if we're referring to a sports ball, its class index would be denoted as C=33. It's important to note that the model has already undergone training on the Common Objects in Context (COCO) dataset, encompassing various object classes.

For image labeling during training, specific values are assigned to objects, measured in pixels from the top-left corner (0,0) of the image. Each object's labeling includes five values formatted as C bx by bw bh. To ensure consistency, the four bounding box (bx, by, bw, bh) values are normalized to range between 0 and 1, representing proportions relative to the image dimensions (1280 x 720 pixels).

For instance, consider two labeled objects within a frame: the sports ball and a person (myself). These labels are represented as tensors—generalized vectors—fed into the model during training.

- Sports Ball: [33, 0.21, 0.58, 0.23, 0.42] (C, bx, by, bw, bh)

- Myself: [1, 0.67, 0.5, 0.5, 0.9] (C, bx, by, bw, bh)

These tensors encapsulate the object's class index along with the normalized bounding box values, facilitating the model's training process.

Prediction / Detection Time

During the prediction phase, we feed 1280 x 720 frames from our camera into YOLO. YOLO employs automatic resizing, adjusting the frame to 416 x 234 and fitting it within a standardized 416 x 416 network. This resizing is achieved by padding any excess space with zeros. YOLO then segments each image into S x S cells, each measuring 32 x 32 pixels (resulting in a reduction factor of 32). In this context, this division creates a grid of 13 x 13 cells, derived from the original image dimensions of 416/32 = 13 x 13 cells.

In a grid divided into cells, consider a scenario using an 8x8 cell layout for illustration purposes. If the center of an object (represented by the red dot) lies within a particular grid cell (depicted by the dark-green box), that specific cell assumes responsibility for detecting that object. Within each bounding box, there exist five values: bx, by, bw, bh, and BC. The first four values—bx, by, bw, bh—convey the position of the box within the grid.

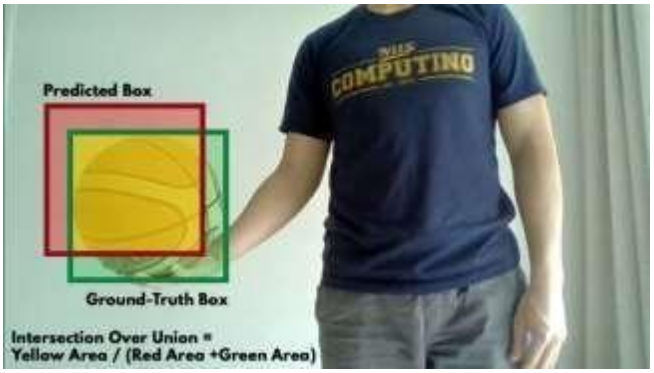
The fifth value, BC (box confidence score), is computed as the

$$\frac{bx-cx}{cx}, \frac{by-cy}{cy}, \frac{bw}{W}, \frac{bh}{H}$$

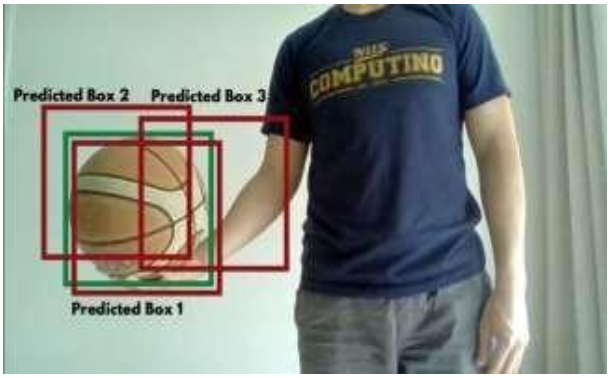
$$\frac{270-155}{155}, \frac{420-360}{360}, \frac{300}{1280}, \frac{300}{720}$$

product of two factors: Pr(Object existing in box) and IOU (Intersection Over Union).

1. Pr(Object existing in box) represents the likelihood of an object being present within the box.
2. IOU (Intersection Over Union) measures the accuracy of the prediction by evaluating the intersection of the predicted and ground truth bounding boxes divided by their union.



There are B bounding boxes predicted in each grid cell



In YOLO v3, each cell predicts B=3 bounding boxes for a single object within that cell. Additionally, for each grid cell, there exist C conditional class probabilities.

Considering the Common Objects in Context (COCO) dataset, there are 80 conditional class probabilities denoted as Pr(Class i | Object) per cell. These probabilities signify the likelihood that the predicted object belongs to Class i, given that there's an object present in the cell.

For instance:

- Class 1 (person): 0.01 □
- Class 2 (bicycle): 0.004
- ...
- Class 33 (sports ball): 0.9
- ...
- Class 80 (toothbrush): 0.02

In the scenario provided, if Class 33 (sports ball) demonstrates the highest probability among these classes within a cell, it will be considered as the prediction for the object detected in that cell, indicating that the object is likely a sports ball.

To sum up the above

There are S x S cells and in each of these cells there are 2 things: 1) B bounding boxes each with 5 values (bx, by, bw, bh, BC), 2) C conditional class probabilities. The predictions are encoded as a S x S x (5 * B + C) tensor.

Non-Max Suppression

In YOLO v3, each grid cell generates B duplicate detections for predicting the object. Non-Maximum Suppression (NMS) functions to eliminate redundant detections with low box confidence scores (BC), ensuring a more accurate and concise prediction.

A simple implementation of NMS involves the following steps:

- Begin with the bounding box that exhibits the highest BC among the detections.
- Eliminate any remaining bounding boxes that overlap this chosen box by more than a specified threshold (e.g., 0.5).
- Repeat steps 1 and 2 until no further bounding boxes remain.

This process iteratively selects the most confident bounding box and removes others that significantly overlap with it, ensuring the final predictions represent unique and accurate detections without redundant or closely overlapping bounding boxes.

Loss Function

1. During training, the adjustment of weights in our model aimed to minimize the loss function. Although the function may appear intricate, breaking it down reveals its intuitive nature.
2. The objective revolves around optimizing the model by adjusting its parameters (weights) to minimize the loss function. This function encapsulates the disparity between predicted

values and ground truth annotations. The process involves iteratively fine-tuning these weights to diminish this disparity, aiming to achieve more accurate predictions.

3 In essence, the complexity of the function lies in its comprehensive assessment of prediction errors, guiding the model towards optimal adjustments in its weights, thereby minimizing the difference between predicted and actual values during training.

Multi-scale Predictions

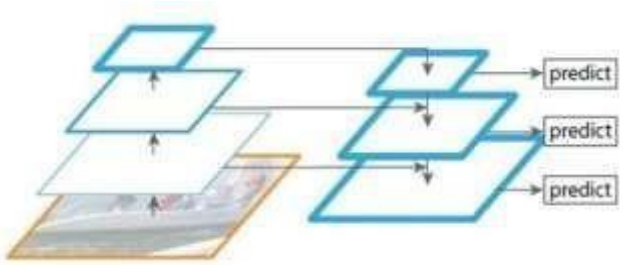
The output of Darknet results in a grid size of 13 x 13 x 1024. Essentially, this implies that the grid consists of 13 x 13 cells, each containing a representation of 32 x 32 pixels.

This structure allows for the possibility of each cell serving as the center for a bounding box. However, due to the limitations of the grid size and pixel representation, this setup tends to excel in detecting larger objects rather than smaller ones. The grid's resolution can influence its ability to effectively detect and localize objects based on their size within this representation.

To address the limitation of detecting smaller-sized objects within the YOLO framework, the concept of multi-scale prediction is introduced. This involves generating predictions using multiple grid sizes alongside the standard 13 x 13 grid.

Before applying the last stride-2 convolution, the size would have been 26 x 26 x 512. Similarly, before the second-to-last stride- 2 convolution, the size would have been 52 x 52 x 256. These additional grid sizes aim to enable the detection of smaller objects by reducing the number of pixels contained within each cell.

To achieve this objective, networks like Feature Pyramid Networks (FPNs) are employed. FPNs specialize in enhancing object detection capabilities across various scales by leveraging multi-scale features from different layers within a network. They facilitate the extraction and integration of features at multiple resolutions, enabling the detection of objects at different sizes more effectively.



Combining Boxes from Various Scales:

Let's perform a quick calculation. Suppose we're working with an image size of 416 x 416 pixels and computing 3 bounding boxes within three different grid sizes: 52 x 52 cells, 26 x 26 cells, and 13 x 13 cells. In this scenario, each cell can serve as the central point for one of the bounding boxes, allowing for a maximum of 3 bounding boxes per cell at each grid scale. The total number of bounding boxes across these multi-scale grids would amount to 10,647. However, it's crucial to note that this represents the maximum potential number of bounding boxes. Many of these boxes

would be significantly small, approaching minimal sizes or even close to zero in terms of coverage within the image.

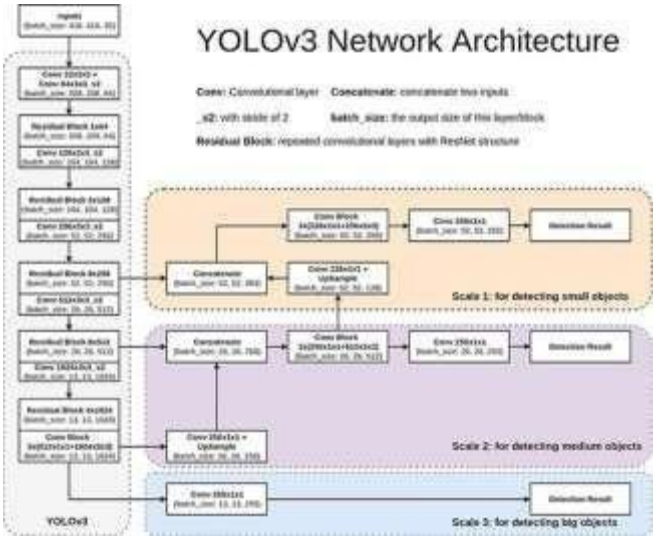
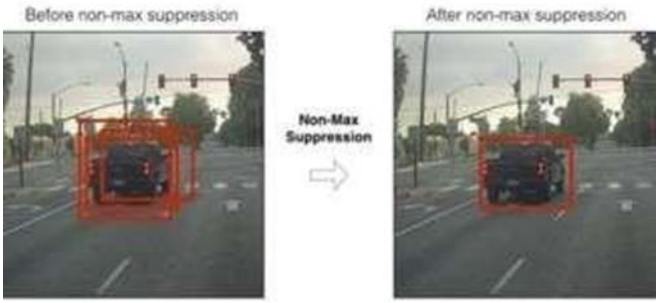


Figure 5 Network Architecture

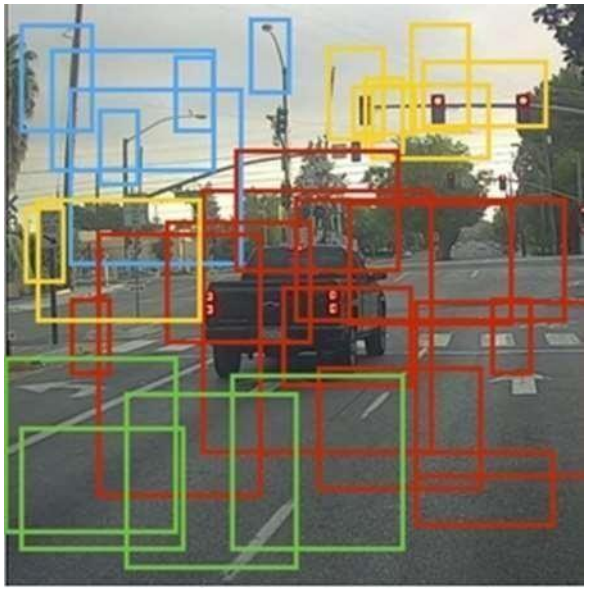
Certainly! To break down the process: Starting from the DarkNet 53 architecture, the goal is to transform the 13 x 13 x 1024 output to 13 x 13 x 255 in order to predict bounding boxes within a 13 x 13 cell grid. Instead of simply using a 255-channel 1 x 1 convolution (as previously done in YOLOv1), a more sophisticated approach is adopted in YOLOv3.

- Utilizing a combination of convolutions - specifically, 3 sets of convolutions comprising 512 kernels with a 1 x 1 convolution, followed by 1024 kernels of 3 x 3 convolutions - helps improve performance. This sequence aims to gather information not only from the current cell but also from the surrounding cells to predict the bounding box effectively within the 13 x 13 grid.
- This process results in the generation of 255 kernels using a final 1 x 1 convolution, achieving the desired output of 13 x 13 x 255 for predicting bounding boxes at the 13 x 13 grid level.
- To cater to the idea of Feature Pyramid Networks (FPNs) and detect objects at varying scales (26 x 26 and 52 x 52 cell grids), a similar process is replicated. The output from the DarkNet 53 architecture is passed through a 1 x 1 convolution to transform the 13 x 13 x 1024 output to 13 x 13 x 256. This is followed by upsampling to convert it to 26 x 26 x 256.
- Concurrently, information from the previous residual block, i.e., the 26 x 26 x 512 output, is also available. Concatenating both outputs results in a combined 26 x 26 x 768 feature map.
- Similarly, the process is extended to predict bounding boxes at the 52 x 52 grid level, following a comparable methodology to the 13 x 13 and 26 x 26 cases.

This approach aims to gather information from different scales and levels of detail within the network, enhancing the model's ability to detect objects across varying sizes and scales.



Filtering:

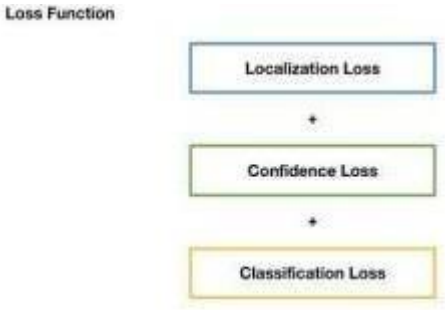


For each bounding box generated, calculations are performed against 'n' anchor values, resulting in 80 probabilities per box. By performing these calculations, 80 probabilities are computed for each anchor.

Among these probabilities, the maximum value is selected.

If this maximum probability is less than 0.5, the box is considered insignificant or useless, and therefore discarded. This process of filtering helps eliminate irrelevant or less confident predictions.

Previously, the model generated over 10,000+ boxes, but not all of these boxes contain valuable information. By applying this filtering mechanism and discarding boxes with probabilities below the threshold of 0.5, the goal is to retain only the most relevant and confident predictions, disregarding the ones considered less useful.



1. Difference between the ground-truth box vs the predicted boundary box.

2. Difference between 100% confidence that an object is in the box vs the box confidence.

3. Difference between C actual class probabilities (0, …, 1, …, 0) vs C predicted class probabilities (0.01, …, 0.8, …, 0.02)

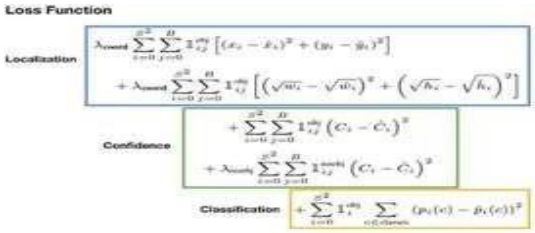
Loss Function

On the other hand, the parameter λnoobj is typically set to 0.5 to diminish the influence of the confidence loss. This adjustment is made

because boxes without objects possess confidence scores BC=0. By reducing the weight assigned to the confidence loss, the model becomes more stable during training and exhibits improved convergence, focusing less on cases where the model predicts absence (BC=0) of objects within boxes.

Voice Feedback

We can leverage the bx and by values, which are relative to



Basically uses sum of squared-differences for each component.
Most symbols are pretty self-explanatory.

$\mathbb{1}_{ij}^{obj}$ = 1 if the j th boundary box in cell i is responsible for detecting the object, otherwise 0

The parameter λcoord is commonly set to 5 to amplify the significance of the localization loss, thereby emphasizing the precise localization of objects during training

the width (W) and height (H) of the image, to ascertain the position of detected objects. With this information, we can construct a text string that describes the object's position.

Pydub and ffmpeg libraries can be employed for various audio manipulations such as adjusting volume, changing format, merging, or splitting audio files, allowing for customization and enhancement of the generated speech audio files.

System Design

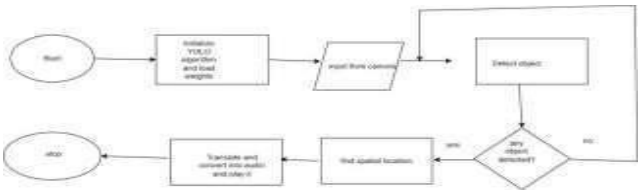
The YOLO object detection system with voice feedback involves several key components:

YOLO Object Detection Algorithm : YOLO, an advanced object detection algorithm utilizing a convolutional neural network (CNN), identifies objects in real-time video streams. It predicts bounding boxes and class probabilities for detected objects in images.

.

User Interface : The user interface facilitates interaction with the system and serves as the platform for receiving feedback. This interface could manifest as a web-based dashboard, a mobile application, or a specialized hardware device, offering users a means to interact with and comprehend the object detection findings.

Integration layer : The integration layer serves as the bridge connecting the YOLO object detection During operation, the YOLO object detection algorithm scrutinizes the video input stream, identifying and detecting objects within



the scene. This analysis enables the system to Object detection projects, particularly those utilizing advanced algorithms like YOLO (You Only Look Once), offer a wide range of practical applications across various industries. These projects revolutionize traditional processes by enabling automated, real-time detection and tracking of objects in images or video streams. generate information about the objects present in

the visual data. seamless communication between these components.

algorithm, the video input stream, and the voice feedback module. Its implementation can leverage different technologies like Python, Node.js, or Java, facilitating

CONCLUSION:

Various algorithms like CNN, Fast R-CNN, and RCNN exist for object detection and recognition. YOLOv3 was selected for its speed and accuracy, enabling real-time object detection and precise object localization. Notably, the proposed system doesn't rely on internet connectivity for text-to-sound conversion, utilizing Play sound to play pre-existing audio files from a local sounds dataset. This approach reduces dependency on external resources. During testing on a PC, the detection time was notably shorter compared to other techniques, affirming its efficiency. Future iterations could involve porting the system onto Raspberry Pi to create smart eyeglasses tailored for assisting visually impaired individuals in detecting and recognizing objects within their surroundings. Expanding the system's capabilities, potential enhancements include integrating age and gender prediction techniques. Additionally, incorporating face recognition technology would empower the system to identify individuals. These augmentations could significantly enhance the system's utility and provide more comprehensive assistance in various scenarios.

REFERENCES:

1. V. Kharchenko and I. Chyrka, “Detection of ground planes using the YOLO neural network,” Int. Meeting. Maths. Method
2. Electromagnetic method. Theory, MMET, vol. 2018-July, pages 294–297, 2018, doi: 1109/MMET.2018.8460392.
3. Z. Cheng, J. Lv, A. Wu, thiab N. Qu, “YOLOv3 object detection algorithm with feature pyramid maintenance in remote sensing images,” Sensors Mater., vol. 32. No. 12 Ib., p. 4537, 2020, doi: 10.18494/sam.2020.3130.
4. H. Jabnon, F. Benzarti and H. Amiri, “Exploration and experience in videos for the blind,” Int. Meeting. Intel. system. December.
5. application form. ISDA, vol. 2016-June, p. 363–367, 2016, doi:10.1109/ISDA.2015.7489256.
6 M. Shah and R. Kapdi, “Object detection using deep neural networks,” Proc. 2017 Int. Conf. Intell.

7 S.Geethapriya, N. Duraimurugan and S.P. Chokkalingam, “Inflight object detection using Yolo,” Int. J. Engineering. envelope. Technology, vol. No. 3 Special Issue, p. 578-581, 2019.

8 S. Geethapriya, N. Duraimurugan and S.P. Chokkalingam, “In-flight object detection using Yolo,” Int. J. project. letter. Technology, vol. 8. Special Report No. 3, p. 17 578-581, 2019.

9 J Redmon, S Divvala, R Girshick et al., “You Only Look Once: Unified,” Real-Time Object Detection[J], p. 779-788, 2015.

10 J Redmon and A Farhadi, YOLO9000: Better, faster and more powerful [J], 2016.

11 W Liu, D Anguelov, D Erhan et al., SSD: Single-Shot Multi-Box Detector[J], p. 21-37, 2015.

12 K He, X Zhang, S Ren et al., “Depth of Image Recognition Residual Learning[C],” Computer Vision and Pattern Recognition, p. 770-778, 2016.

13 P Felzenszwalb, D Mcallester and D Ramanan, A method for isolating multiple deformable part models[J], vol. 8, p. 1-8, 2008.

14 S. N. Srivatsa, G. Sreevathsa, G. Vinay and P. Elaiyaraja,

15 “Exploratory Research Using Interactive Learning with OpenCV and Python,” International Research Journal of Engineering andTechnology (IRJET), Volume 8, Issue 1, Pages 227-230, 2021.