Smart Home Security Camera With YOLO and DeepSORT

Aadharsh Aadhithya, Jayanth M, Madhav Kishore, Vishnu Radhakrishnan, and Visweswararn M

Amrita Vishwa Vidyapeetham .

(Dated: 29 June 2023)

Abstract. In this Project we present a cost-effective approach to enhance the intelligence of home security cameras using the YOLO (You Only Look Once) object detection algorithm in combination with the DeepSORT (Deep Simple Online and Realtime Tracking) algorithm. The proposed system utilizes a camera connected to a laptop, which is also mounted on a servo motor controlled by an Arduino. When a person is detected by YOLO, bounding boxes and class IDs are passed to DeepSORT for tracking. If the object's center exceeds a predefined threshold from the video's width, the servo motor rotates to align the camera with the person, effectively establishing a smart home security camera capable of tracking individuals. This article discusses the selection of YOLO based on its low latency and high frames per second (FPS) properties compared to other object detection methods. The YOLO model is trained on the CrowdHuman dataset, enabling accurate predictions of humans. Additionally, we highlight the capabilities of DeepSORT for multi-object tracking. Experimental results demonstrate the successful implementation of the proposed system, providing insights into its inference and performance using the WandB platform.

Keywords: YYOLO, DeepSORT, object detection, tracking, smart home security camera, low latency, high FPS, CrowdHuman dataset, inference, performance.

INTRODUCTION

Home security has always been a primary concern for homeowners. With advancements in technology and the rise of artificial intelligence, traditional security camera systems are being transformed into intelligent and proactive surveillance solutions. These intelligent systems utilize computer vision algorithms to detect and track objects in real-time, providing homeowners with enhanced security measures and peace of mind.The motivation behind this project is twofold. Firstly, it stems from the increasing need for effective and affordable home security solutions. Traditional security camera systems often lack the ability to actively monitor and track individuals within a home environment. By incorporating object detection and tracking algorithms, we aim to develop a system that addresses this limitation and provides homeowners with a proactive and intelligent security solution.

Secondly, the project is driven by the desire to gain a deeper understanding of the YOLO (You Only Look Once) algorithm and its practical application in real-life scenarios. YOLO is renowned for its low latency and high frames per second (FPS) properties, making it suitable for real-time object detection tasks. By working on a project that applies YOLO to a specific use case, such as home security, we can explore its capabilities, limitations, and potential for improvement. This hands-on experience allows us to gain valuable insights and practical knowledge in the field of computer vision and object detection.

The main objective of this project is two-fold. Firstly, we aim to design and implement a smart home security camera system that combines the YOLO object detection algorithm with the DeepSORT (Deep Simple Online and Realtime Tracking) tracking algorithm. This integration will enable real-time and accurate detection and tracking of individuals within a home environment, enhancing the security measures.
Secondly, we seek to gain a comprehensive understanding of the YOLO algorithm through its practical implementation. By applying YOLO to a real-life problem, we can explore its architecture, training process, and its strengths and weaknesses. This project provides an opportunity to delve into the technical intricacies of YOLO, understand its performance characteristics, and assess its suitability for real-time object detection tasks.

The significance of this project lies in its potential to revolutionize home security systems. By incorporating object detection and tracking algorithms, traditional security cameras can be transformed into intelligent and proactive surveillance systems. The ability to actively track individuals within the home environment provides homeowners with timely notifications and improved security measures, reducing the risk of potential threats. Furthermore, this project contributes to the broader field of computer vision and artificial intelligence by providing insights into the practical application of the YOLO algorithm. Through hands-on experience, we can gain a deeper understanding of its strengths and limitations, paving the way for further advancements and improvements in the field.

This paper presents a comprehensive study on the development and implementation of a smart home security camera system. The system utilizes the YOLO object detection algorithm and the DeepSORT tracking algorithm to enable real-time tracking of individuals within a home environment. The YOLO algorithm is chosen based on its low latency and high frames per second (FPS) properties, making it suitable for real-time applications. Training YOLO on the CrowdHuman dataset enables accurate predictions of human objects. The DeepSORT algorithm is integrated with YOLO to perform robust multi-object tracking, handling occlusions and maintaining accurate tracklets. The system incorporates a servo motor controlled by an Arduino, allowing the camera to dynamically adjust its orientation to track the detected objects effectively.

SINGLE STAGE VS TWO STAGE NETWORKS

Single-stage Networks

Single-stage detectors are a popular class of object detection algorithms that aim to accurately localize and classify objects in a single pass through the network. These detectors offer real-time performance with a simpler architecture compared to their two-stage counterparts. In this section, we will discuss several types of single-stage detectors and highlight their key characteristics.

YOLO (You Only Look Once): YOLO is a pioneering single-stage object detection algorithm known for its simplicity and efficiency. It divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell. YOLO achieves impressive speed and accuracy trade-offs, making it suitable for real-time applications. Variants of YOLO, such as YOLOv2, YOLOv3, and YOLOv4, have been developed, each introducing improvements in terms of accuracy and speed.

SSD (Single Shot MultiBox Detector): SSD is another widely used single-stage object detector that operates at different scales. It predicts bounding boxes and class probabilities at multiple feature maps with different resolutions, allowing it to handle objects of various sizes effectively. SSD achieves high accuracy across different object scales, but it may be slightly slower compared to YOLO due to the use of multiple feature maps.

RetinaNet: RetinaNet addresses the challenge of handling a large number of background samples in single-stage detectors. It introduces a novel focal loss that assigns higher weights to hard examples, improving the detection performance of rare objects. RetinaNet utilizes a feature pyramid network to capture multi-scale features, enabling accurate detection across different object sizes.

Two-Stage Detectors :

Two-stage detectors are another popular class of object detection algorithms that involve a two-step process: region proposal generation followed by accurate object classification and refinement. These detectors typically achieve higher accuracy but at the cost of increased computational complexity. In this section, we will discuss various types of two-stage detectors and highlight their characteristics.

Faster R-CNN (Region-based Convolutional Neural Networks): Faster R-CNN introduced the concept of region proposal networks (RPNs) for generating object proposals. It first generates a set of candidate object proposals and then refines these proposals using a region-based CNN for accurate classification and localization. Faster R-CNN achieves excellent detection accuracy but can be computationally expensive due to its two-stage architecture.

R-FCN (Region-based Fully Convolutional Networks): R-FCN improves upon Faster R-CNN by using fully convolutional networks for region-based detection. It eliminates the need for RoI pooling, making the detection process more efficient. R-FCN achieves comparable accuracy to Faster R-CNN while reducing computational complexity.

Mask R-CNN: Mask R-CNN extends Faster R-CNN by introducing an additional branch for predicting object masks. In addition to accurate object localization and classification, Mask R-CNN enables pixel-level segmentation of objects within an image. It has become the state-of-the-art method for instance segmentation, offering precise object boundaries and high-quality segmentation masks.

Cascade R-CNN: Cascade R-CNN addresses the challenge of handling objects with varying difficulty levels. It introduces a cascaded architecture that consists of multiple stages, each refining the object proposals by progressively

rejecting easy negatives. Cascade R-CNN achieves superior performance, particularly for challenging objects, at the cost of increased computation.

Each type of two-stage detector has its strengths and weaknesses, with trade-offs between accuracy and computational efficiency. These detectors have demonstrated exceptional performance in a wide range of object detection tasks, particularly when high accuracy is paramount.

BASICS OF YOLO

IOU

IOU (Intersection over Union) is a term used to describe the extent of overlap of two boxes. The greater the region of overlap, the greater the IOU. Here A and B are our object masks A intersection B indicates the area which is in the

$$IoU(A,B) = {A \cap B}/{A \cup B}$$

intersection of both of the masks, while A union B indicates the total area of the figures.

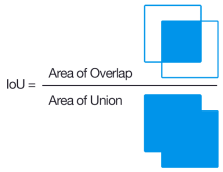
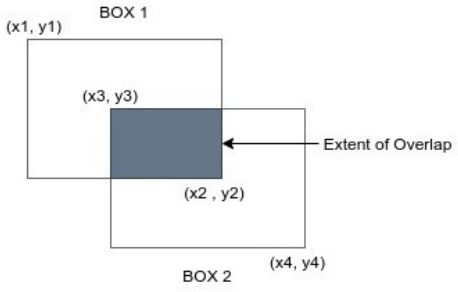


FIGURE 1. IOU

If IoU is close to 1 then we can say that our model perfectly overlaps the object.If IoU is close to 0 or 0 then we can say that our model didn't predict the object coordinates at all.
Let us assume that box 1 is represented by [x1, y1, x2, y2], and box 2 is represented by [x3, y3, x4, y4]. [x_{inter1},



y_{inter1}, x_{inter2}, y_{inter2}, to denote the coordinates of the top left, and bottom right of the intersection.To calculate the top left corner of the intersection, we compare the top left corners of each of the boxes. We can see from the examples below, that x_inter1 can be found by seeing which box has its top left corner more to the right. Similarly y_{inter1} can be found by seeing which box has its top left corner lower than the other. Mathematically they can be calculated as: x_{inter1}=$max(x_1, x_3)$ To calculate the bottom right corner of the intersection, we compare the bottom right corners of each of the boxes.

x_{inter2} can be found by seeing which box has its bottom right corner more to the left.

Similarly y_{inter2} can be found by seeing which box has its bottom right corner higher than the other. Mathematically they can be calculated as: x_{inter2} = $min(x_2, x_4)$ y_{inter2} =$min(y_2, y_4)$

$$\text{width\_inter} = (x_{\text{inter2}} - x_{\text{inter1}})$$

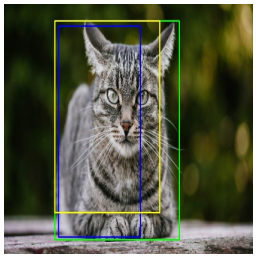
$$\text{height\_inter} = y_{\text{inter2}} - y_{\text{inter1}}$$

$$\text{area\_inter} = \text{width\_inter} \times \text{height\_inter}$$

then we calculate IOU as $IOU = area_inter/are_union$

NMS

Non max suppression is a technique used mainly in object detection that aims at selecting the best bounding box out of a set of overlapping boxes.In the following image, the aim of non max suppression would be to remove the yellow, and blue boxes, so that we are left with only the green box.



This Illustrates the process of how NMS is imiplemented.

- Define a value for Confidence-Threshold, and IOU-Threshold.
- Sort the bounding boxes in a descending order of confidence.
- Remove boxes that have a confidence < Confidence-Threshold.
- Loop over all the remaining boxes, starting first with the box that has highest confidence.
- Calculate the IOU of the current box, with every remaining box that belongs to the same class.
- If the IOU of the 2 boxes > IOU-Threshold, remove the box with a lower confidence from our list of boxes.

YOLOV5: AN OVERVIEW

YOLOv5 is an object detection model that consists of three main components: Backbone, Neck, and Head. These components work together to make dense predictions of object classes, objectness scores, and bounding boxes. In this section, we will explore each component in detail, including the mathematical equations involved.

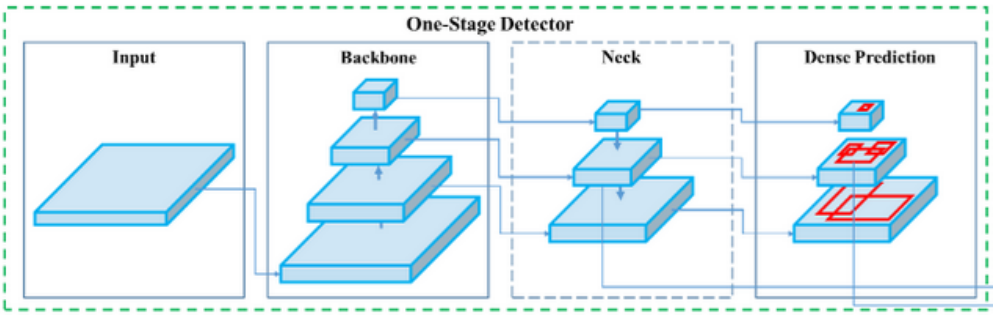


FIGURE 2. YOLOv5 Architecture

Model Backbone

The fundamental structure of YOLOv5 relies on a modified form of the Darknet architecture known as CSP-Darknet53. This framework serves the purpose of extracting high-level features from the input image denoted as X
in
$mathbbR^{HtimesWtimesC}$, where H, W, and C represent the image's height, width, and number of channels, respectively.

CSP-Darknet53 comprises multiple convolutional layers, which can be expressed mathematically as:

$$Y = \text{Conv}_1(\text{Conv2}(\ldots \text{Conv}N-1(\text{Conv}_N(X))\ldots)), \tag{1}$$

Here, Conv_i signifies the i-th convolutional layer, and N indicates the total count of convolutional layers within the backbone.

Each convolutional layer applies a set of adjustable filters to the input feature map, followed by a non-linear activation function like ReLU. The resulting feature map Y contains significant semantic information pertaining to the input image.

The backbone of YOLOv5 is an already trained network employed to extract comprehensive feature representations from input images. This process helps reduce the image's spatial resolution while enhancing its feature resolution. In YOLOv5, the CSP-Darknet53 network is utilized as the backbone. CSP-Darknet53 expands upon the Darknet53 network utilized in YOLOv3 by incorporating the Cross Stage Partial (CSP) network strategy.

The CSP network strategy addresses the issue of redundant gradients that can arise from employing dense and residual blocks within deep networks. It preserves the beneficial aspects of DenseNet's feature reuse characteristics while curtailing the gradient flow to mitigate the redundancy in gradient information. This strategic approach contributes to improved training efficiency and overall model performance.

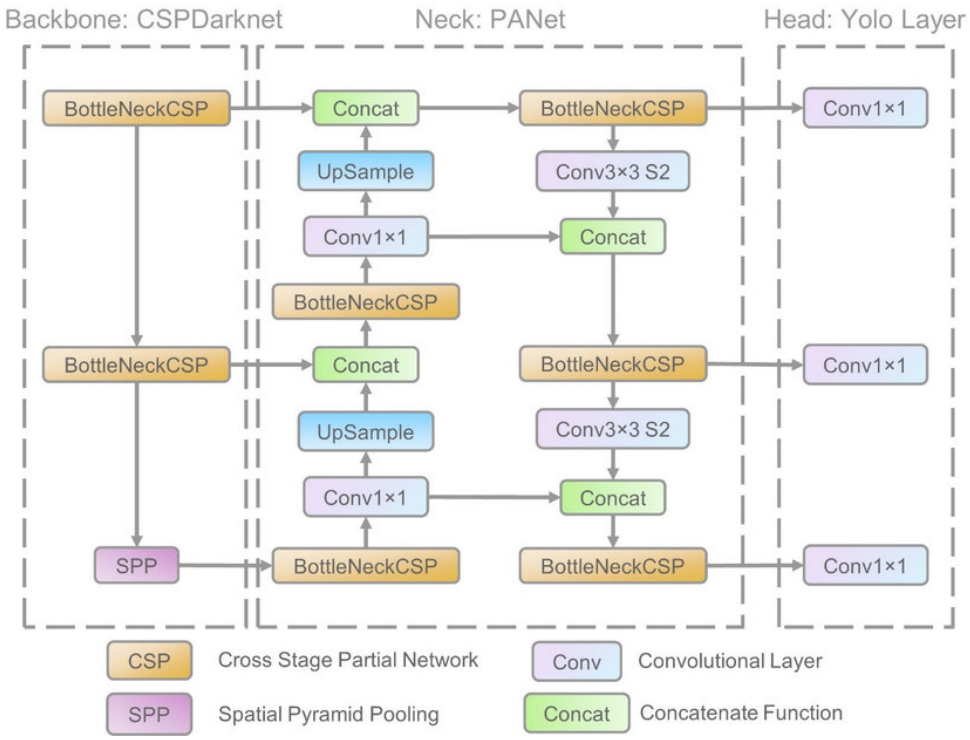


FIGURE 3. YOLOv5 Architecture

Model Neck

The neck component of YOLOv5 plays a crucial role in consolidating features from different scales and improving their spatial representation. It incorporates two key components: Spatial Pyramid Pooling with Focus (SPPF) and Cross Stage Partial Network (CSP) with Path Aggregation Network (PAN).

The SPPF module captures features at multiple scales by applying pooling operations with various kernel sizes to the input feature map denoted as Y from the backbone. Mathematically, the SPPF module can be represented as:

$$Z = \text{SPPF}(Y), \tag{2}$$

where Z represents the resulting feature map from the SPPF module.

The CSP-PAN module combines low-resolution and high-resolution feature maps to create a comprehensive representation. It consists of a sequence of convolutional blocks with skip connections that merge features from different scales. The mathematical representation of the CSP-PAN module is as follows:

$$V = \text{CSP-PAN}(Z, Y), \tag{3}$$

where V represents the resulting feature map from the CSP-PAN module.

The neck component in YOLOv5 extracts feature pyramids, allowing the model to effectively handle objects of varying sizes and scales. YOLOv5 introduces two significant changes to the model's neck compared to previous versions.

Firstly, it utilizes a modified version of Spatial Pyramid Pooling (SPP), which aggregates information from the inputs and produces an output of fixed length. This technique significantly expands the receptive field and focuses on the most relevant contextual features, while maintaining network speed. The SPPF module mentioned earlier is a part of this SPP implementation.

Secondly, the Path Aggregation Network (PANet) incorporates the BottleNeckCSP module into its architecture. PANet, which is a feature pyramid network, improves information flow and enhances localization accuracy in tasks such as mask prediction.

In summary, the neck component in YOLOv5 leverages the SPPF module and the CSP-PAN module to extract feature pyramids, providing the model with the ability to handle objects of various sizes and scales effectively. The modified SPP implementation expands the receptive field, while the BottleNeckCSP integration in PANet improves information flow and localization accuracy.

Model Head

The model head in YOLOv5 performs the final stage operations. It applies anchor boxes on feature maps and generates the final output, including object classes, objectness scores, and bounding boxes. YOLOv5 uses the same head as YOLOv3 and YOLOv4, which consists of three convolutional layers.

The target coordinates for the bounding boxes in YOLOv5 are computed using a different set of equations compared to previous versions. These equations introduce changes in how the bounding box coordinates are calculated, leading to improved performance. The specific equations used in YOLOv5 are shown in Figure 4.

$$\begin{aligned}
&\text{(a)} \quad x = \sigma(t_x) + c_x \\
&\qquad\quad y = \sigma(t_y) + c_y \\
&\qquad\quad w = p_w e^{t_w} \\
&\qquad\quad h = p_h e^{t_h}
\end{aligned}
\qquad
\begin{aligned}
&\text{(b)} \quad x = \sigma(t_x) + \text{sign}(t_x)(\frac{w}{2} - 0.5) + c_x \\
&\qquad\quad y = \sigma(t_y) + \text{sign}(t_y)(\frac{h}{2} - 0.5) + c_y \\
&\qquad\quad w = p_w e^{t_w} \\
&\qquad\quad h = p_h e^{t_h}
\end{aligned} \tag{4}$$

FIGURE 4. Equations used to compute the target bounding boxes. (a) Equations used in previous versions (YOLOv2, YOLOv3). (b) Equations used in YOLOv5.

The equations depend on various parameters such as t_x and t_y (predicted offsets), c_x and c_y (cell offsets), p_w and p_h (anchor box dimensions), w and h (predicted bounding box dimensions), and σ (sigmoid function).

Activation Function

Choosing an appropriate activation function is crucial for deep learning models. In YOLOv5, the authors have used the SiLU (Sigmoid Linear Unit) and Sigmoid activation functions.

SiLU, also known as the swish activation function, has been applied to the convolutional operations in the hidden layers. It is defined as:

$$\text{SiLU}(x) = x \cdot \sigma(x) \tag{5}$$

where σ represents the sigmoid function.

On the other hand, the Sigmoid activation function has been used with the convolutional operations in the output layer. It is defined as:

$$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{6}$$

The choice of these activation functions helps introduce non-linearity and make the model more expressive.

Loss Function

YOLOv5 returns three outputs: object classes, bounding boxes, and objectness scores. To compute the loss, YOLOv5 uses the BCE (Binary Cross Entropy) loss for classes and objectness, and the CIoU (Complete Intersection over Union) loss for localization.

The final loss is computed as follows:

$$\mathscr{L} = \lambda_{\text{cls}} \cdot \mathscr{L}_{\text{cls}} + \lambda_{\text{obj}} \cdot \mathscr{L}_{\text{obj}} + \lambda_{\text{loc}} \cdot \mathscr{L}_{\text{loc}} \tag{7}$$

where λ_{cls}, λ_{obj}, and λ_{loc} are weighting factors, and $\mathscr{L}_{\text{cls}}$, $\mathscr{L}_{\text{obj}}$, and $\mathscr{L}_{\text{loc}}$ are the individual loss terms for classes, objectness, and localization, respectively.

The BCE loss is given by:

$$\mathscr{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N} [y_i \cdot \log(\hat{y}_i) + (1-y_i) \cdot \log(1-\hat{y}_i)] \tag{8}$$

where N is the total number of elements, y_i is the ground truth, and $\hat{y}_i$ is the predicted value.

The CIoU loss is a modified version of the Intersection over Union (IoU) loss that takes into account the distance between the predicted and ground truth bounding boxes. The CIoU loss encourages better localization accuracy and is defined as:

$$\mathscr{L}_{\text{CIoU}} = 1 - \text{IoU} + \text{CIoU}_{\text{DIoU}} + \text{CIoU}_{\text{GIoU}} \tag{9}$$

where IoU is the Intersection over Union, and $\text{CIoU}_{\text{DIoU}}$ and $\text{CIoU}_{\text{GIoU}}$ are additional terms that penalize the distance between the predicted and ground truth boxes.

OTHER IMPROVEMENTS

Apart from the main components and mathematical formulations discussed above, YOLOv5 incorporates several other improvements. These enhancements contribute to the overall performance and usability of the model.

The Focus Layer

The Focus Layer is a key component introduced in YOLOv5 that replaces the initial layers of the network. It plays a crucial role in reducing the computational complexity of the model while improving its efficiency and speed. Let's delve into the details of the Focus Layer and explore what happens within this component.

The main purpose of the Focus Layer is to capture and process feature information from the input image efficiently. It achieves this by reducing the resolution of the input feature map while maintaining its spatial information. By doing so, the Focus Layer reduces the number of parameters and computations required in subsequent layers.

The operation of the Focus Layer can be summarized as follows:

Input Feature Map: The Focus Layer takes the initial feature map as its input. This feature map typically has a high resolution, containing detailed information about the input image.

Channel Shuffling: The input feature map is divided into four parts, and the channels within each part are grouped together. This grouping ensures that each part captures a different region of the input image.

Downsample: Each of the four parts is downsampled using a stride of 2. This downsampling reduces the spatial resolution of the feature map while increasing its channel (feature) resolution. The downsampling process helps to extract rich and compact representations of the input image.

Concatenation: The downsampled feature maps from the four parts are concatenated along the channel dimension. This concatenation results in a final feature map that has a reduced spatial resolution compared to the initial feature map.

The main advantage of the Focus Layer is that it significantly reduces the computational burden of subsequent layers without sacrificing much information. By performing downsampling at the beginning of the network, the Focus Layer effectively reduces the spatial dimensionality of the input, making subsequent convolutional operations more efficient.

Furthermore, the Focus Layer helps improve the model's ability to capture multi-scale information. By dividing the input feature map into four parts and downsampling them independently, the Focus Layer enables the model to capture features at different scales and resolutions. This multi-scale information is crucial for accurately detecting objects of various sizes and maintaining spatial context.

Eliminating Grid Sensitivity

Grid sensitivity refers to a limitation that was present in previous versions of YOLO (You Only Look Once) object detection models. These earlier versions, such as YOLOv2 and YOLOv3, faced challenges when detecting objects located at the corners of an image. This issue arose due to the constraints imposed by the equations used for predicting bounding boxes.

In YOLOv5, specific modifications were introduced to overcome this grid sensitivity problem and enhance the model's performance in detecting objects near the image edges. The adjustments made in YOLOv5 are as follows:

Expanding the Range of Center Point Offset: In previous versions, the center point offset was limited to the range of (0-1). However, YOLOv5 expands this range to (-0.5, 1.5). By allowing the center point offset to extend beyond the image boundaries, YOLOv5 enables the model to more easily detect objects near the image edges. This expanded range ensures that objects located at the corners of an image can be accurately detected.

Constraining Height and Width Scaling Ratios: Previous equations used for predicting bounding boxes had unbounded height and width scaling ratios. This lack of constraints could lead to training instabilities. In YOLOv5, these scaling ratios are constrained, resulting in improved stability during training. By constraining the scaling ratios, YOLOv5 mitigates potential issues and ensures more reliable and consistent predictions of bounding boxes.

These adjustments in YOLOv5 effectively address the grid sensitivity problem encountered in earlier versions. By expanding the range of the center point offset and constraining the scaling ratios, YOLOv5 enhances the model's ability to accurately detect objects at the image corners and near the edges. This improvement contributes to the overall effectiveness and robustness of YOLOv5 in object detection tasks.

Running Environment

While previous versions of YOLO were implemented on the Darknet framework written in C, YOLOv5 is implemented in PyTorch. This implementation choice provides greater flexibility in controlling the encoded operations and allows for easier experimentation and customization.

In summary, YOLOv5 is a powerful object detection model that incorporates various improvements over previous versions. Its backbone, neck, and head components work together to extract feature representations, generate feature pyramids, and perform the final prediction tasks. The activation functions, loss functions, and other enhancements contribute to the model's performance and speed. By understanding the mathematical equations and concepts behind YOLOv5, we can appreciate its capabilities and potential applications in real-time object detection scenarios.

SINGLE OBJECT TRACKER AND MULTI OBJECT TRACKER

Object tracking is a fundamental task in computer vision that involves predicting the positions of objects throughout a video by utilizing their spatial and temporal features. Tracking can be divided into two main categories: single object tracking and multi-object tracking. In this section, we will explore these two types of trackers and provide examples for each.

Single Object Tracker

Single object trackers are designed to track a single target object even when there are multiple objects present in the frame. These trackers typically initialize the object's location in the first frame and then track it throughout the sequence of frames. Single object trackers are known for their speed and efficiency.

Traditional single object trackers are often built using traditional computer vision techniques such as correlation filters. Examples of traditional single object trackers include CSRT (Channel and Spatial Reliability Tracker) and KCF (Kernelized Correlation Filters). However, with the advancements in deep learning, deep learning-based trackers have shown superior performance compared to traditional trackers. For instance, SiamRPN (Siamese Region Proposal Network) and GOTURN (Generic Object Tracking Using Regression Networks) are examples of deep learning-based single object trackers.

Multi Object Tracker

Multi object trackers, also known as Multiple Object Trackers (MOTs), are capable of tracking multiple objects simultaneously. These trackers are trained on large amounts of data and can handle tracking objects of different classes in real-time while maintaining high accuracy. MOTs are particularly useful in scenarios where multiple objects need to be tracked and identified.

DeepSORT (Deep Learning for MOT with Segmentation and Tracking) is a notable example of a multi-object tracking algorithm. DeepSORT is an extension of the SORT (Simple Online and Realtime Tracking) algorithm that incorporates deep learning techniques. It combines motion and appearance descriptors to reduce identity switches and improve tracking efficiency. Other examples of powerful multi-object tracking algorithms include JDE (Joint Detection and Embedding) and CenterTrack.

Examples of Tracking Applications

Now that we have discussed single object trackers and multi-object trackers, let's explore some real-world applications of object tracking:

Traffic Monitoring

Object tracking can be utilized for traffic monitoring systems. By tracking vehicles on the road, it becomes possible to analyze traffic patterns, detect traffic violations, and even implement systems like Automatic License Plate Recognition (ALPR).

Sports Tracking

Tracking can be employed in sports to track various elements such as the ball or players. Ball tracking can help determine fouls, track scorers in a game, or provide valuable insights for sports analysis. Player tracking enables monitoring individual players' movements, positioning, and interactions on the field.

Multi-Camera Surveillance

In multi-camera surveillance scenarios, object tracking can play a vital role. The core idea is re-identification, where a person or an object is tracked across different cameras. By maintaining consistent IDs for tracked objects, it becomes possible to re-identify objects that appear in different camera views. This can be beneficial for tasks like intrusion detection and security monitoring.

These examples showcase the versatility and importance of object tracking in various domains. Whether it is monitoring traffic, analyzing sports events, or enhancing surveillance systems, object tracking provides valuable insights and enables numerous applications.

TRACKING BY DETECTION AND WITHOUT DETECTION

Tracking by Detection

The type of tracking algorithm where the object detector detects the objects in the frames and then perform data association across frames to generate trajectories hence tracking the object. These types of algorithms help in tracking multiple objects and tracking new objects introduced in the frame. Most importantly, they help track objects even if the object detection fails.

Tracking without Detection

The type of tracking algorithm where the coordinates of the object are manually initialized and then the object is tracked in further frames. This type is mostly used in traditional computer vision algorithms as discussed earlier.

INTRODUCTION TO DEEPSORT

DeepSORT is a computer vision tracking algorithm for tracking objects while assigning an ID to each object. DeepSORT is an extension of the SORT (Simple Online Realtime Tracking) algorithm. DeepSORT introduces deep learning into the SORT algorithm by adding an appearance descriptor to reduce identity switches, hence making tracking more efficient. To understand DeepSORT, let's first see how the SORT algorithm works.

Simple Online Realtime Tracking (SORT)

SORT is an approach to object tracking where rudimentary approaches like Kalman filters and Hungarian algorithms are used to track objects and claim to be better than many online trackers. SORT is made up of four key components which are as follows:

- **Detection:** This is the first step in the tracking module. In this step, an object detector detects the objects in the frame that are to be tracked. These detections are then passed on to the next step. Detectors like FrRCNN, YOLO, and more are most frequently used.
- **Estimation:** In this step, we propagate the detections from the current frame to the next, which is estimating the position of the target in the next frame using a constant velocity model. When a detection is associated with a target, the detected bounding box is used to update the target state where the velocity components are optimally solved via the Kalman filter framework.The Kalman filter is a framework that estimates an object's state by combining current measurements with previous estimates. It predicts the future state based on known dynamics, allowing accurate position estimation even without new measurements, which is useful during occlusions or when objects are temporarily out of view. Additionally, the filter fuses measurements from multiple bounding boxes, improving accuracy and robustness by combining information from different bounding boxes .
- **Data Association:** We now have the target bounding box and the detected bounding box. So, a cost matrix is computed as the intersection-over-union (IOU) distance between each detection and all predicted bounding

boxes from the existing targets. The assignment is solved optimally using the Hungarian algorithm. If the IOU of detection and target is less than a certain threshold value called IOUmin, then that assignment is rejected. This technique solves the occlusion problem and helps maintain the IDs.

- **Creation and Deletion of Track Identities:** This module is responsible for the creation and deletion of IDs. Unique identities are created and destroyed according to the IOUmin. If the overlap of detection and target is less than IOUmin, then it signifies the untracked object. Tracks are terminated if they are not detected for TLost frames; you can specify what the amount of frames should be for TLost. Should an object reappear, tracking will implicitly resume under a new identity.

The objects can be successfully tracked using SORT algorithms, beating many state-of-the-art algorithms. The detector gives us detections, Kalman filters give us tracks, and the Hungarian algorithm performs data association. So, why do we even need DeepSORT? Let's look at it in the next section.

DeepSORT

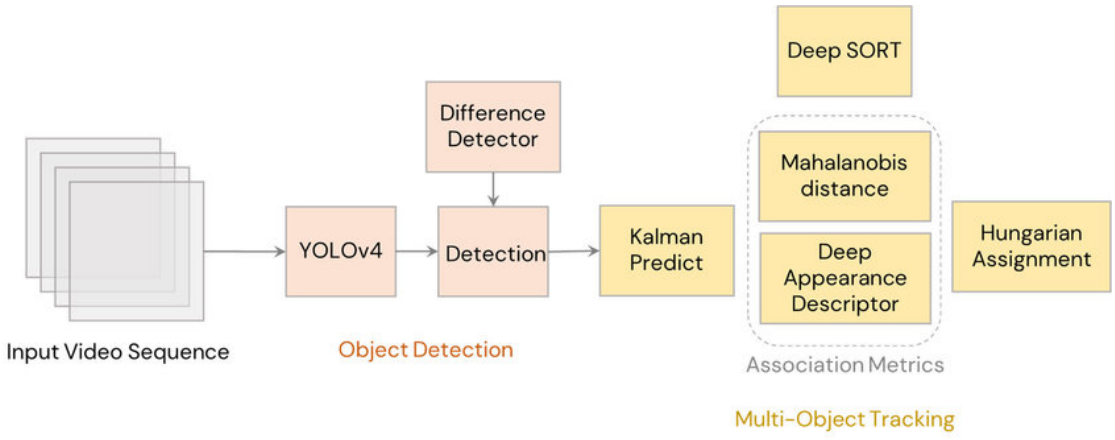


FIGURE 5. DeepSORT architecture

SORT performs very well in terms of tracking precision and accuracy. But SORT returns tracks with a high number of ID switches and fails in the case of occlusion. This is because of the association matrix used. DeepSORT uses a better association metric that combines both motion and appearance descriptors. DeepSORT can be defined as the tracking algorithm which tracks objects not only based on the velocity and motion of the object but also the appearance of the object. To train the deep association metric model in the DeepSORT, cosine metric learning approach is used. According to DeepSORT's paper, "The cosine distance considers appearance information that is particularly useful to recover identities after long-term occlusions when motion is less discriminative." That means cosine distance is a metric that helps the model recover identities in the case of long-term occlusion and when motion estimation also fails. Using these simple things can make the tracker even more powerful and accurate.

RESULTS

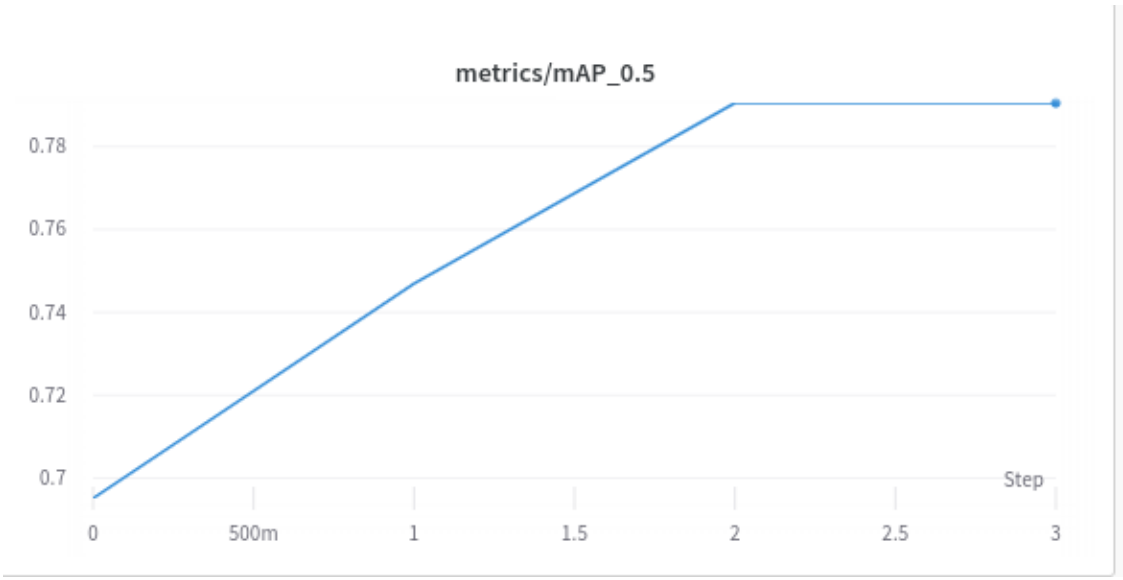


FIGURE 6. This figure shows the MAP acquired while FInetuning the model

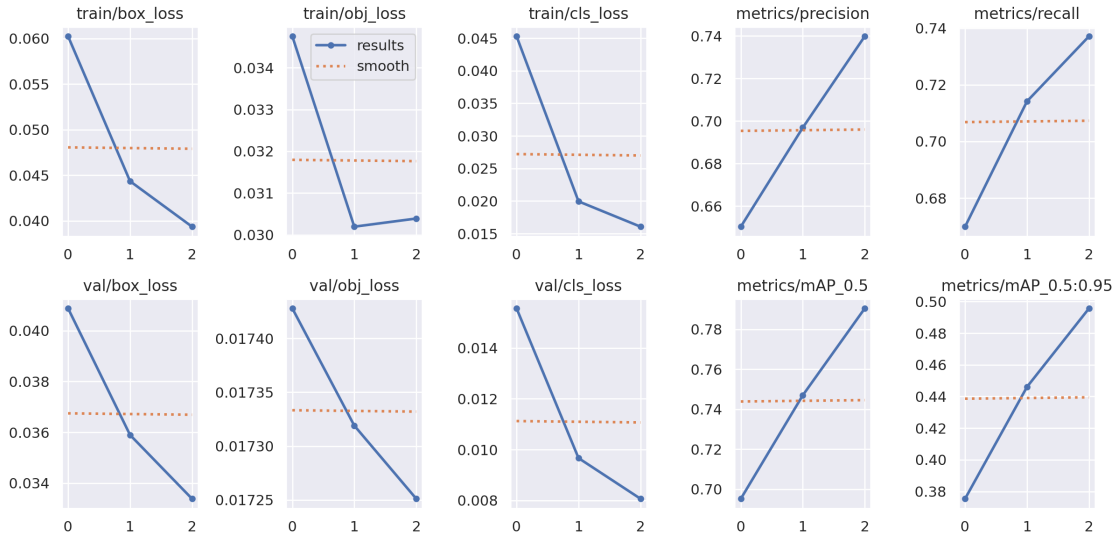


FIGURE 7. This figure shows box loss,objective loss, class loss , precision, MAP for both train and validation.

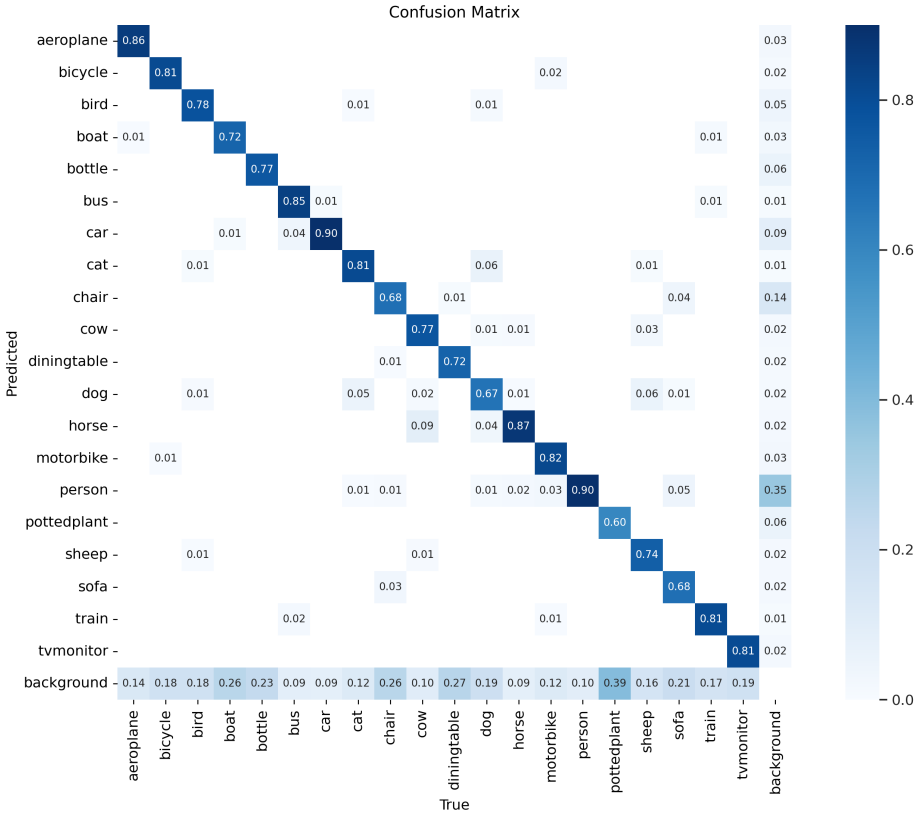


FIGURE 8. This figure shows the confusion matrix

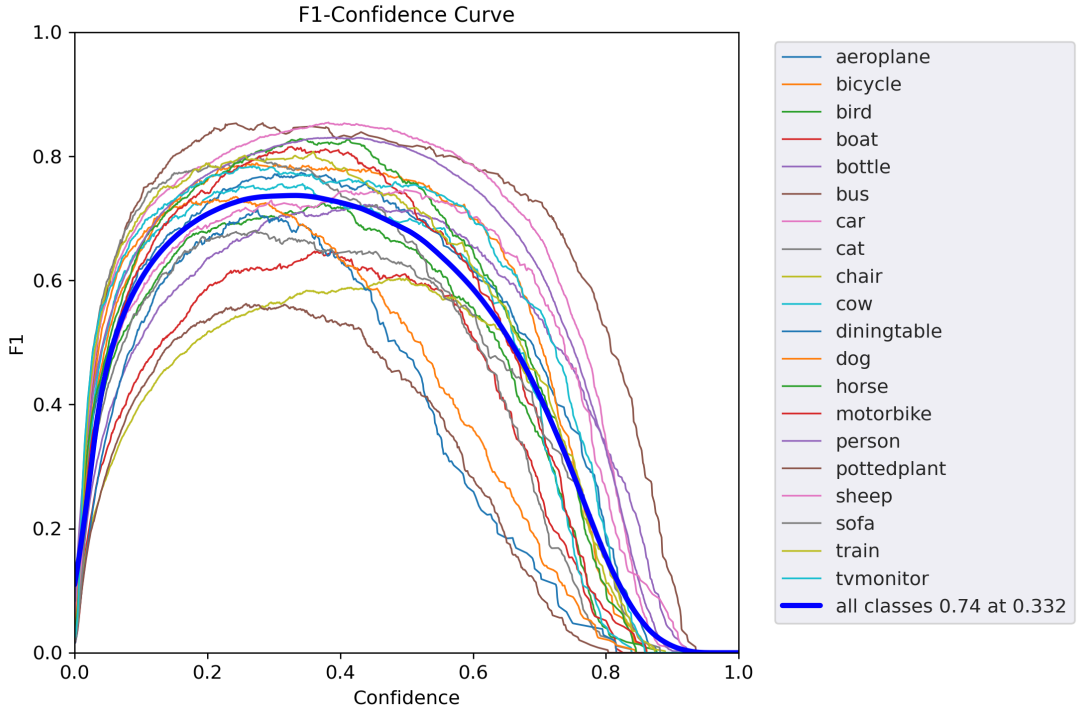


FIGURE 9. This figure shows the F1- confidence curve

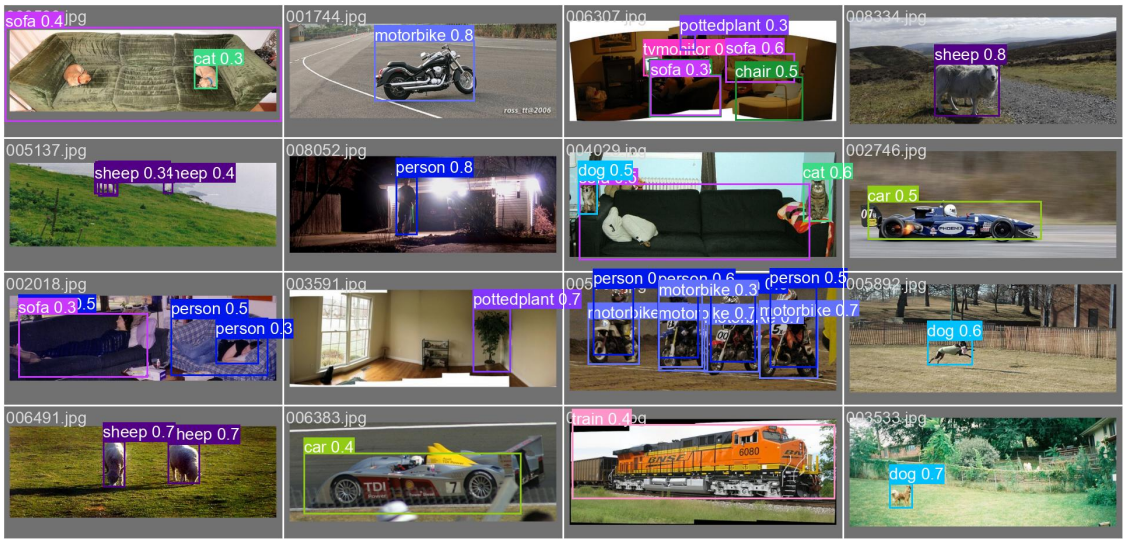


FIGURE 10. This figure shows how our model performs in validation