Unattended Baggage Detection Using YOLO v3

Saurav Jayasurya
Department of Electronics and Communication Engineering
Mar Baselios College of Engineering and Technology, Thiruvananthapuram, Kerala, India

Afsal Ak
Department of Electronics and Communication Engineering
Mar Baselios College of Engineering and Technology, Thiruvananthapuram, Kerala, India

Roshan Martin
Department of Electronics and Communication Engineering
Mar Baselios College of Engineering and Technology, Thiruvananthapuram, Kerala, India

Roselin Raju
Assistant Professor, *Department of Electronics and Communication Engineering*
Mar Baselios College of Engineering and Technology, Thiruvananthapuram, Kerala, India

Abstract- Our objective is to realize a model that can be implemented in larger system such as airports to increase, security and save time. We are not intending to realize this model in railway stations and other crowded areas since the background in those areas are dynamically varying in larger parameters. In multi-national airports like Japan and Singapore, the number of AI used in cleaning and elderly assistance increases, hence we use this in already existing system and implement our detection model to allow the work of the airport staff to be more convenient. The bot will detect abandoned luggage and baggage items after rush hour, and carry them away to the Lost and Found Area. Even when there is no availability of bots in cases like India, this can be efficiently used and with minimal human surveillance too. The model can detect unattended baggage from the control room, ring in the nearest authorities and alert them to the bag's location.

Keywords – Object Detection, Deep Learning, YOLOv3, Surveillance, Security.

I. Introduction

In this project, our main objective is to assimilate the realization of object detection and then step by step software design to filter people's luggage and to figure out whether they're abandoned or not. We use a wide variety of simulations and Open source packages like TensorFlow. To accessing code and the main assembly language we preferred to use in this software technology is the programming language known as Python due to its ease and efficiency. We also use the Open Source library known as OpenCV2, due to it having functions and algorithms that support real-time computer vision. The model that we're going to use is Deep Learning. More specifically, a region proposal system called You Only Look Once (YOLO) is used. In this method YOLO contains the details of data's pre-trained and it can detect these images by only looking once. Why do we use TensorFlow as our main object detection library? Its flexibility and its huge community support are the main reason we decided to make it our main API.

Since Luggage has a wide variety to objects to train in the model, despite the specific nature of the project, we still think that it's going to be much more challenging than the actual project sounds. But with TensorFlow 2, the latest version of the API, the introduction of eager execution reduces the lines of code used to realize object detection. Annotations of labels are also done at the end of the object detection phase.
The two main entities that we are going to train in this project are:

1. Luggage item
2. Human Beings

Training a program to extract facial features of a human is going to be hardest part of the two.

II. Proposed Algorithm

A. *Block Diagram*

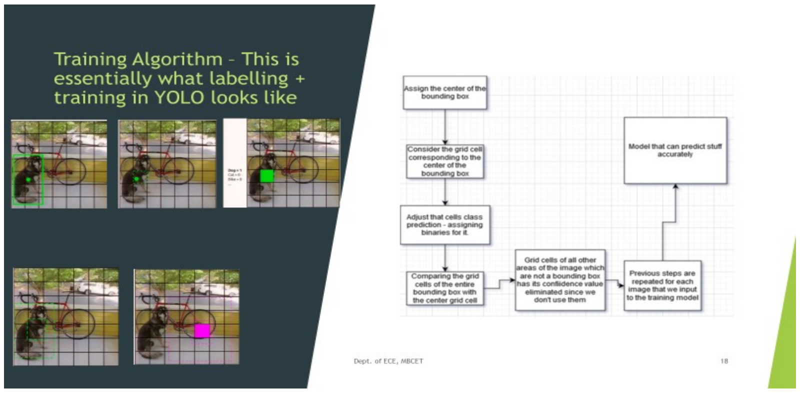

Training Algorithm - This is essentially what labelling + training in YOLO looks like
Assign the center of the bounding box
Consider the grid cell corresponding to the center of the bounding box
Adjust that cells class prediction - assigning binaries for it
Comparing the grid cells of the entire bounding box with the center grid cell
Grid cells of all other areas of the image which are not a bounding box has its confiidence value eliminated since we don't use them
Previous steps are repeated for each image that we input to the training model
Model that can predict stuff accurately
Dept. of ECE, MBCET
18

Fig. 1: Training algorithm and Block Diagram

The picture is segregated into different grid boxes with planar as well as spatial co-ordinates. When the bounding box is classed, the center of the bounding box is considered while allocating the grid cell that corresponds to that center point of the bounding box as shown in the above figure. The model assigns a class prediction value that tends to the similarities of that box relative to the other grid cells in that specific bounding box. From this we ascertain a confidence value for different grid cells in the others areas of the image which are not a bounding box, then eliminates the confidence values pertaining to them. This way the model can configure and single out certain features of an object in the image with respect to the others and thereby, making the model more adept at identifying and distinguishing them. More the number of PICTURES and data that's input to the model, better accuracy in which the model works on.

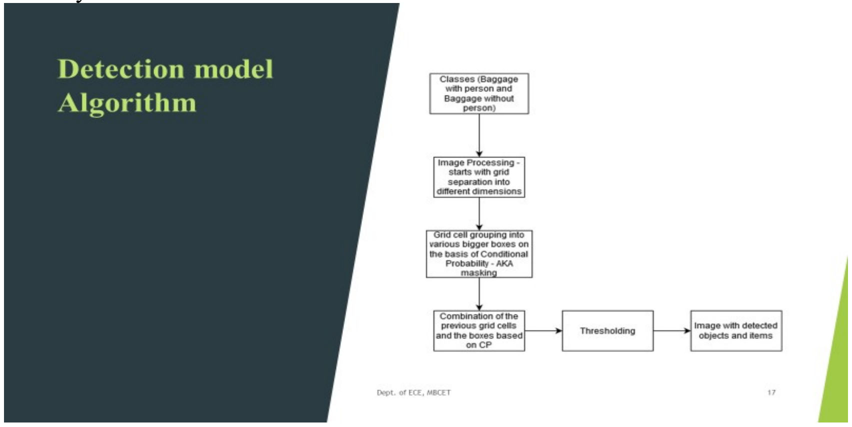

Detection model Algorithm
Classes (Baggage with person and Baggage without person)
Image Processing - starts with grid separation into different dimensions
Grid cell grouping into various bigger boxes on the basis of Conditional Probability - AKA masking
Combination of the previous grid cells and the boxes based on CP
Thresholding
Image with detected objects and items
Dept. of ECE, MBCET
17

Fig 2: Detection Model

The objects that we've based the model on are people and baggage. The video or the frame-by-frame pictures is segmented or placed under a mathematical projection that fragments the frame into grids with a specific dimension. These grids are collectively called grid cells. The grid cells are the given a certain value of CP (Conditional Probability), this parameter is then used to mask them if group them into bigger areas. The areas grouped together have a common CP value which elaborates into the detection mechanism. The areas that do not have a CP value that's specified in the trained model is eliminated while the areas that have a certain CP value already specified in the algorithm is then considered and labelled once the model figures out the CP value most commonly found in the area. This is how object detection is handled and this is the algorithm that a training model do.

B. Methodology

To predict the training parameters and determine the actions of training the remaining weights we use an Image Pre-Processing. Before we apply the distance algorithm to obtain the pre-processing of the image. Here every frame of the video is loaded from the disk, then

1. Convert the image/frame to greyscale.
2. Blur it using a Gaussian filter with a 7*7 Kernel.
3. Apply Canny Edge Detector.
4. Call cv2.findcontours.
5. Establishing and Calibrating the reference object.

Then we use the Distance Determining Algorithm, in this method the distances of can be identified by knowing to important properties of the object, that is the dimension of that object (in terms of millimeters, inches, etc.) and the object should be easily identifiable from the image. From this to find the distance we use the following steps,

1. Find the centroid of the bounding boxes.
2. Establish the two midpoint of the contour boxes, for reference, a and b.
3. Using the variables a and b, apply the Euclidian distance equation.
4. The result is we get the PPM.

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}.$$

After the LabelImg, annotating and pre-processing the collected data’s given to the system, we successfully created a system that can detect, label and identify if a baggage is found unattended. Through physical data including live CCTV footage we can obtain the respected data with the help of fully automated AI which work’s on the basic of creating a bounding boxes around the object that has to be detected and the if these baggage’s including with human presence. If this baggage’s were found unattended, that means no human presence were found near this baggage’s will be found undetected and a data sheet will have created automatically at the data centre and informs the authorities that a baggage is found undetected at the preferred location. With this technology if a person lost his or her baggage can also be found easily through checking the footage with accurate precision

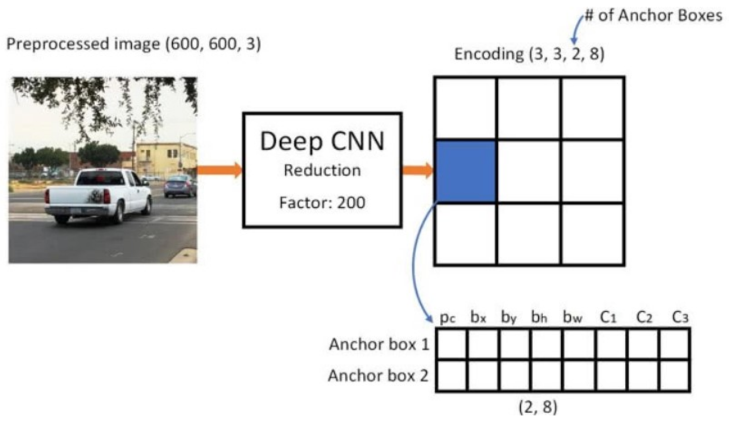


Fig 3: Yolo Architecture of detection

In the above figure shown is the basic detection method used in YOLO detection, where the concept of breakdown the images to grid cells is unique as compared to other object localization solutions. In reality, one image can be cut to 19 *19 grid cells, but for this explanation process we are going to divide the image into 3*3 grid.

From the above image we can detect the objects based on these grids which bounds the objects. If the object to be detected comes under a single grid it's easy to detect by forming a bounding boxes inside that box. If the object is larger than the grid cell and its bounding box intersects with several grid cell, YOLO algorithm takes the middle point of the bounding box and associates it to the grid cell containing it.

But it is not that much accurate to detect images when it comes to multiple images, so to overcome this difficulty we found a solution for this by using Anchor Box. Anchor box makes it possible for the YOLO algorithm to detect multiple objects in single grid cell. The idea of anchor box is used so that to add one more "dimension" to the output labels by the pre-defining a number of anchor boxes. Another reason for choosing a variety of anchor box shapes is to allow the model to specialize better. That is some of the objects are trained to detect a wider object like bags, cars, another output will be trained to detect a tall and skinny objects like a human and so on the shapes.

Another method that is used here is knowns as Non-Max Suppression. This is a commonly used algorithm used for cleaning up when multiple boxes are predicted for the same object. According to the grid box the objects are detected, but for a single object multiple boxes will be present and from these boxes choosing the right box depends on the pc value if it's more than 0.6. The process is does in 3 steps as follows:

1. Discard all the boxes with pc less or equal to 0.6.
2. Pick the box with the largest pc output as a prediction.
3. Discard any remaining box with IoU (Intersection over Union) greater than or equal to 0.5.

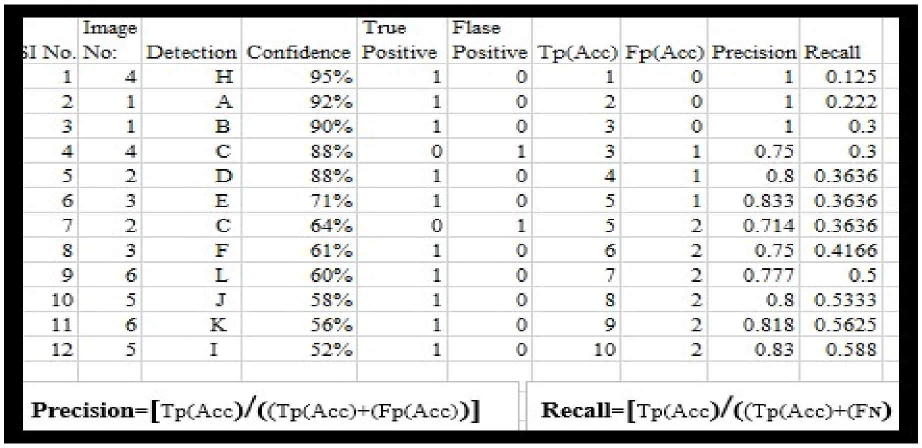
Thus we get the final cleaned up prediction.

III. Experiment and Result

The main aim of this project is to create a completely integrated AI driven surveillance in public places which scans for nefarious baggage's that are likely left unattended, detects them and sends the signal to the nearest authorities. Once integrated, there will be no requirement for human surveillance and the only instance where human inputs will be mandatory is when the system already detects the target bag and tags their location.

While processing the live video footages an image is taken as each frames from the video and then convert these frames to grey scale so that the basic detection occurs. While these detection occurs we obtained some basic data's that connects the detection accuracy, precision and so on.

First, we wrote a code that we used before the training stage to disintegrate frames of a random sample video and introduce ground truth boxes, which after training generates the actual detection contours along with their confidence values. For that we used 6 images and compiled a table using the information we obtained. We recompiled the table in order of confidence. Then calculates the precision and recall in each detected object of every frames. Thus taking only the highest precision value of re-occurring recall value to obey the laws of interpolation.



SI No.	Image No:	Detection	Confidence	True Positive	Flase Positive	Tp(Acc)	Fp(Acc)	Precision	Recall
1	4	H	95%	1	0	1	0	1	0.125
2	1	A	92%	1	0	2	0	1	0.222
3	1	B	90%	1	0	3	0	1	0.3
4	4	C	88%	0	1	3	1	0.75	0.3
5	2	D	88%	1	0	4	1	0.8	0.3636
6	3	E	71%	1	0	5	1	0.833	0.3636
7	2	C	64%	0	1	5	2	0.714	0.3636
8	3	F	61%	1	0	6	2	0.75	0.4166
9	6	L	60%	1	0	7	2	0.777	0.5
10	5	J	58%	1	0	8	2	0.8	0.5333
11	6	K	56%	1	0	9	2	0.818	0.5625
12	5	I	52%	1	0	10	2	0.83	0.588

Precision=[Tp(Acc)/((Tp(Acc)+(Fp(Acc))] **Recall=**[Tp(Acc)/((Tp(Acc)+(FN)

Table 1: Precision Recall Table

From the above table we can plot the Precision-Recall graph. Precision is the ability of a model to identify only the relevant objects. It is the percentage of correct positive predictions and Recall is the ability of a model to find all the relevant cases. It is the percentage of true positive detected among all relevant ground truths.
Like first we get True Positive value and False Positive value by using image to anchor segregation which are binaries. If the corresponding image has correctly detected object, it's TP (True Positive) and if it doesn't detect it correctly but still shows the bounding box, it's FP (False Positive) and if the bounding box does not appear or if the image has NO detected objects, then it's FN (False Negative). The values are either 0 or 1, 0 for NO and 1 for YES. Then we do the calculation we can see that in the table. Using the equation, we calculate the precision and recall for every image. Then once we find the precision and recall values, we plot the precision recall graph shown below.

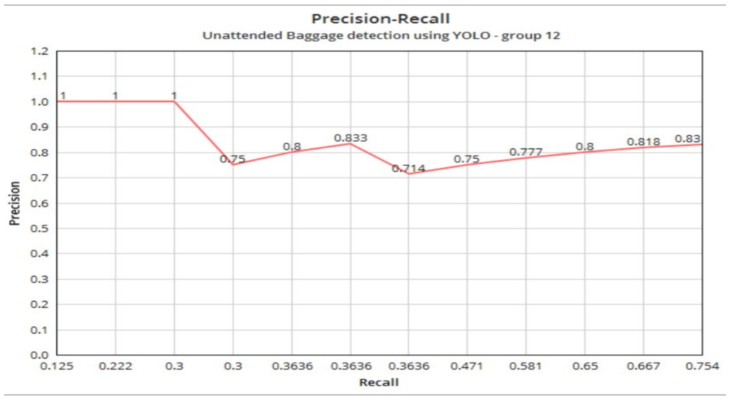


Fig 4: 10-Point Interpolated D Graph

The Precision * Recall curve is a good way to evaluate the performance of an object detector as the confidence is changed by plotting a curve for each object class. An object detector of a particular class is considered good if it's precision stays high as recall increase, which means that if you vary the confidence threshold, the precision and recall will still be high. Currently, the interpolation performed by PASCAL VOC challenge uses all data points, rather than interpolating only 10 equally spaced points as stated in their paper. As we want to reproduce their default implementation, our default code follows their most recent application.

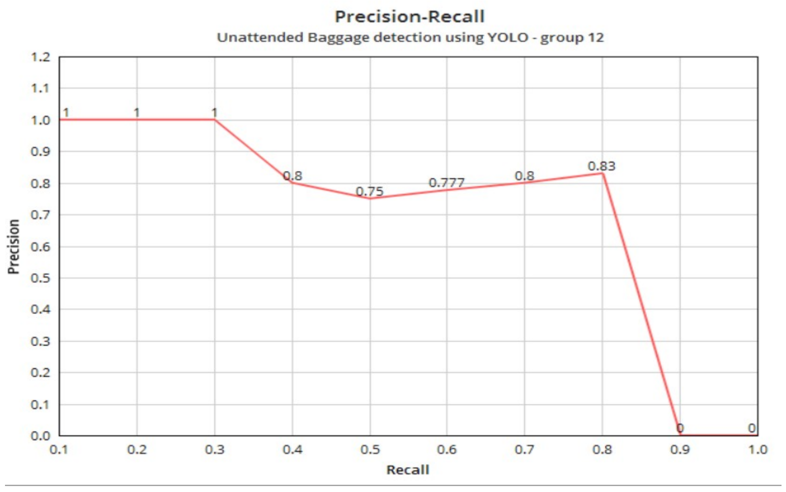
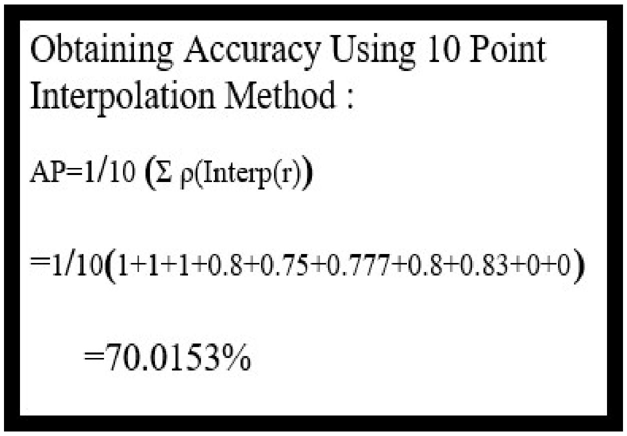


Fig 5: Precision Recall Graph

Re-occurring recall points gets eliminated and the precision value that has the maximum confidence is taken. All recall values are estimated to the next decimal point. The 10-Point interpolation tries to summarize the shape of the Precision * Recall curve by averaging the precision at a set of eleven equally spaced recall levels. Instead of using the precision observed at each point, the AP is obtained by interpolating the precision only at 10 levels r taking the maximum precision whose recall value is greater than r. Thus we can obtain the accuracy.


Obtaining Accuracy Using 10 Point Interpolation Method :

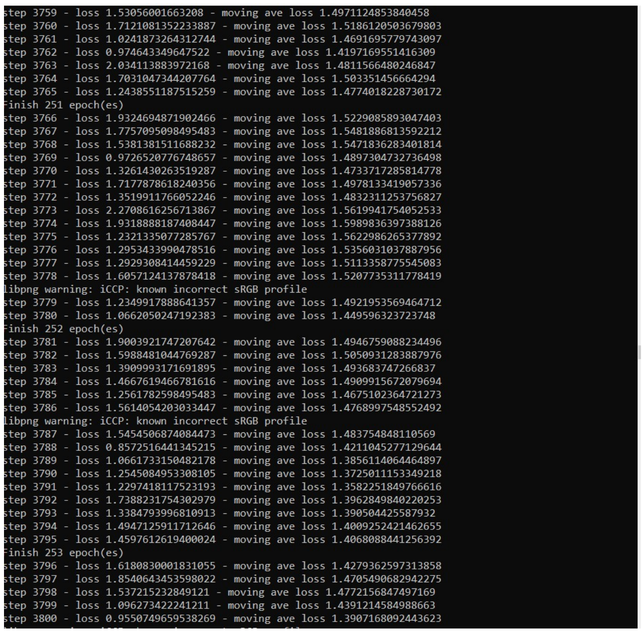
AP=1/10 (Σ ρ(Interp(r)))

=1/10(1+1+1+0.8+0.75+0.777+0.8+0.83+0+0)

=70.0153%

Fig 6: Precision Recall Graph

While training the collected data’s we came at a point where loss occurs per each iteration. These loss is based on each epoch of the compiling. We can see that it occurs from the 3759 step in our project, that is shown below.



```
step 3759 - loss 1.53056001663208 - moving ave loss 1.4971124853840458
step 3760 - loss 1.7121081352233887 - moving ave loss 1.5186120503679803
step 3761 - loss 1.0241873264312744 - moving ave loss 1.4691695779743097
step 3762 - loss 0.974643349647522 - moving ave loss 1.4197169551416309
step 3763 - loss 2.034113883972168 - moving ave loss 1.4811566480246847
step 3764 - loss 1.7031047344207764 - moving ave loss 1.503351456664294
step 3765 - loss 1.2438551187515259 - moving ave loss 1.4774018228730172
Finish 251 epoch(es)
step 3766 - loss 1.9324694871902466 - moving ave loss 1.5229085893047403
step 3767 - loss 1.7757095098495483 - moving ave loss 1.5481886813592212
step 3768 - loss 1.5381381511688232 - moving ave loss 1.5471836283401814
step 3769 - loss 0.9726520776748657 - moving ave loss 1.4897304732736498
step 3770 - loss 1.3261430263519287 - moving ave loss 1.4733717285814778
step 3771 - loss 1.7177878618240356 - moving ave loss 1.4978133419057336
step 3772 - loss 1.3519911766052246 - moving ave loss 1.4832311253756827
step 3773 - loss 2.2708616256713867 - moving ave loss 1.5619941754052533
step 3774 - loss 1.9318888187408447 - moving ave loss 1.5989836397388126
step 3775 - loss 1.2321335077285767 - moving ave loss 1.5622986265377892
step 3776 - loss 1.2953433990478516 - moving ave loss 1.5356031037887956
step 3777 - loss 1.2929308414459229 - moving ave loss 1.5113358775545083
step 3778 - loss 1.6057124137878418 - moving ave loss 1.5207735311778419
libpng warning: iCCP: known incorrect sRGB profile
step 3779 - loss 1.2349917888641357 - moving ave loss 1.4921953569464712
step 3780 - loss 1.0662050247192383 - moving ave loss 1.449596323723748
Finish 252 epoch(es)
step 3781 - loss 1.9003921747207642 - moving ave loss 1.4946759088234496
step 3782 - loss 1.5988481044769287 - moving ave loss 1.5050931283887976
step 3783 - loss 1.3909993171691895 - moving ave loss 1.493683747266837
step 3784 - loss 1.4667619466781616 - moving ave loss 1.4909915672079694
step 3785 - loss 1.2561782598495483 - moving ave loss 1.4675102364721273
step 3786 - loss 1.5614054203033447 - moving ave loss 1.4768997548552492
libpng warning: iCCP: known incorrect sRGB profile
step 3787 - loss 1.5454506874084473 - moving ave loss 1.483754848110569
step 3788 - loss 0.8572516441345215 - moving ave loss 1.4211045277129644
step 3789 - loss 1.0661733150482178 - moving ave loss 1.3856114064464897
step 3790 - loss 1.2545084953308105 - moving ave loss 1.3725011153349218
step 3791 - loss 1.2297418117523193 - moving ave loss 1.3582251849766616
step 3792 - loss 1.7388231754302979 - moving ave loss 1.3962849840220253
step 3793 - loss 1.3384793996810913 - moving ave loss 1.390504425587932
step 3794 - loss 1.4947125911712646 - moving ave loss 1.4009252421462655
step 3795 - loss 1.4597612619400024 - moving ave loss 1.4068088441256392
Finish 253 epoch(es)
step 3796 - loss 1.6180830001831055 - moving ave loss 1.4279362597313858
step 3797 - loss 1.8540643453598022 - moving ave loss 1.4705490682942275
step 3798 - loss 1.537215232849121 - moving ave loss 1.4772156847497169
step 3799 - loss 1.096273422241211 - moving ave loss 1.4391214584988663
step 3800 - loss 0.9550749659538269 - moving ave loss 1.3907168092443623
```

Fig 7: Position of Moving Average Loss.

Moving Average Loss is the data that we get from the training of custom data, at the final steps, like the last 10 steps, we average the 10 values and get the moving average. Thus we obtained a graph showing the moving average loss that is based on no. of steps per loss.

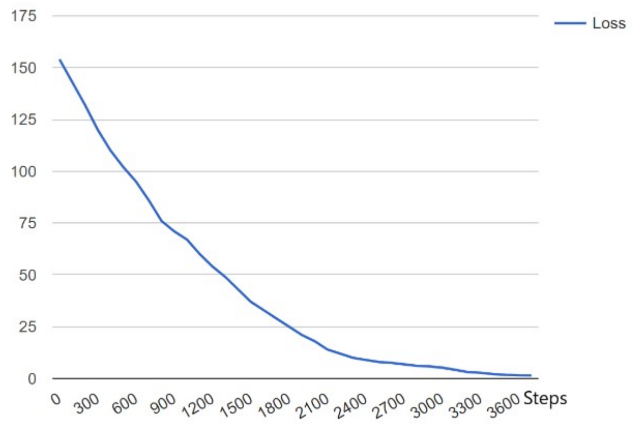


Fig 8: Moving Average Loss.

While Convergence Loss is the SAME value, the only difference is the graphs, both graphs are different accordingly. Moving average loss graph inputs the entire 3800 steps we trained and plots the loss to the corresponding step and this makes that graph. Convergence loss graph is basically the last 10 steps plotted on the graph as well as their loss. The values both convergence loss and moving average loss is the same.

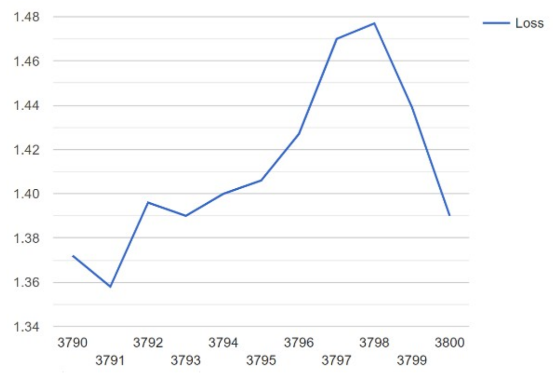


Fig 9: Convergence Loss

From all the above research and analysis of our project, the results we obtained from our custom training model were good and some of those data's are like, Mean Moving average loss = 1.525, Learning rate is 1e-05, the Epochs were 253, the no. of Iterations took 3800 steps, the Accuracy of this detection system becomes almost 70.0513% which is much higher than ant detection system available now.

This model can be used in areas where large number of people come and go, the person that get tagged with the baggage gets saved in the database. So in the database there exists the model of the bag and the person that gets tagged along with. If the same person does not appear with the bag for more than a minute, the bag is identified as unattended.

Nefarious organization and individuals have been using bags and suitcases to disguise their bombs so they can create havoc in public places. They usually leave the baggage in a crowd and then leave the area. Our method is crude but will definitely minimize the risk of such a thing happening in the public places even in area of high population density. Integrated with a system in public places especially airports and if provided a small investment, the dynamic accuracy of the model will increase exponentially.

IV.CONCLUSION

We aim to make it easier for local authorities to detect suspicious activities and deter malicious plots that may put a lot of lives at risk. Since our technology has the advantage of convenience as well affordability, implementation of a system like this isn't too hard. We've already got a large network of CCTV camera systems that's routed in essential and public locations like railway stations, airports, hospitals etc. We can tap into that existing system and configure our technology in such a way that compatibility is boosted to the highest extent since latency is key. Additionally, our technology can also be used in less intense situations like theft and situations where people misplace their luggage. These are very common scenarios, especially in places like the airport where there exist huge and dynamic influx of people, making it rather easy for luggage to get lost or stolen. Our project also aims to make the already existing technology of object detection exponentially more accurate since we narrow down the items we train. Despite the way it sounds, it still is a challenging mountain to climb since luggage items come in a plethora of models. Each model is to be treated as a class, and each class gets to have a variety of images that needs to be trained to get peak accuracy.

This model is fully AI-driven and hence there is no requirement of human surveillance. The only instance where human assistance is required is when the model detects an unattended bag, where the nearest authorities are alerted to its location.

REFERENCES

[1] Zhong-Qiu Zhao, Shou-tao Xu and Xindong-Wu "**Object Detection with Deep Learning: A Review",** IEEE Trans. NNL systems.

[2] K. K. Sung and T. Poggio, **"Example-based learning for view-based Human face detection,"** IEEE Trans. Pattern Anal. Mach. Intell., vol. 20, no. 1, pp. 39–51, 2002.

[3] Wojek, P. Dollar, B. Schiele, and P. Perona, **"Pedestrian detection: An evaluation of the state of the art,"** IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 4, p. 743, 2012.

[4] Krizhevsky, I. Sutskever, and G. E. Hinton, "**ImageNet classification with deep convolutional neural networks,"** in NIPS, 2012.

[5] Sun, P. Wu, and S. C. Hoi, **"Face detection using deep learning: An Improved Faster R- CNN approach,"** arXiv: 1701.08289, 2017.

[6] Arka Prava Jana, Abhiraj Biswas, Mohana, **"YOLO based Detection and Classification of Objects in video records**", 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT-2018), MAY 18th & 19th 2018.

[7] Zhihao Chen, Redouane Khemmar, Benoit Decoux, Amphani Atahouet, Jean-Yves Ertaud, **“Real Time Object Detection, Tracking, and Distance and Motion Estimation based on Deep Learning: Application to Smart Mobility”**, 978-1-7281-5546-3/19/$31.00 ©2019 IEEE.

[8] Li Tan, Xu Dong, Yuxi Ma, Chong Yu, **“A Multiple Object Tracking Algorithm Based on YOLO Detection”**, 2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI 2018).