Mask RCNN and DeepSORT

Ishani Mhatre
M.S.E. ESE
University of Pennsylvania
Philadelphia, USA

Ruchi Gupte
M.S.E. Robotics
University of Pennsylvania
Philadelphia, USA

***Abstract*—Object detection is an important Computer Vision technique which leverages Deep learning to produce meaningful results. Faster R-CNNs(Region Based Convolutional Neural Network) is one such technique which solves the problem of object detection with 3 neural networks. A deep fully convolutional network proposes regions and the Fast R-CNN detector uses the proposed regions. After detecting an object we implemented Mask R-CNN [1] to output a binary mask predicting object instance. This given pipeline achieves object detection along with Instance segmentation and has the added benefit of modularity, where each given block can be improved further to improve overall results. We further extended our project by implementing multi object tracking with DeepSort which is an extension of SORT(Simple Real-Time Object Tracking) algorithm. Our implementation effectively reduced the number of identity switches around 75% of the time while tracking moving objects.**

Code: https://github.com/Ruchi-Gupte/CIS680_Final_Project
Novelty: https://www.youtube.com/watch?v=qWqGCyW5Qfo
Video: https://www.youtube.com/watch?v=besmD-9EXSM

I. Introduction

Instance Segmentation is an important step towards training neural networks to not only ascertain the difference between various classes but to also identify different instances of objects within the classes. This is the logical next step from semantic segmentation which considers all objects of a class as one entity, and allows us to identify more nuanced characteristics in a given scene and identify the shape and structure of the object and not just it's general location. Fast R-CNNs(Region Based Convolutional Neural Network) is one such technique which solves the problem of object detection in 2 steps: region proposal step and the classification step and requires 3 neural networks. After detecting an object, Mask R-CNN [1] adds a branch(Fully Convolutional Network on top of a CNN based feature map) to Faster R-CNN and outputs a binary mask that says whether or not a given pixel is part of an object. This is a detect-then-segment structure where a mask is detected post detection.

DeepSORT[7] is a extension of SORT[8](Simple Real-Time Object Tracking) algorithm which tracks objects mainly based on the velocity and motion of the object using a Kalman Filter assigned to each detected ID with an appearance descriptor through a pre-trained association metric which is trained using neural networks. Through this extension, objects could be tracked through longer periods of occlusions, effectively reducing the number of identity switches around 65% of the time.

II. Related Work

Fast-RCNN [2][4] which stands for Fast Region-Based Convolutional Neural Network, is a object detection based model that uses the concept of region-based CNN and it extracts features using Region of Interest Pooling. This is the backbone of Mask RCNN as the objects detected by Fast RCNN are then further used to perform instance segmentation.

Other papers which deal with instance segmentation are DeepMask [3] and SOLO which employ different approaches to the instance segmentation problem. While Mask RCNN first detects bounding boxes and then segments the instance mask in each bounding box, SOLO's approach learns an affinity relation by pushing away pixels belonging to different instances and pulling close pixels in the same instance. DeepMask on the other hand creates a class-agnostic mask and confidence of the patch containing a centered object (without class category).

Kalman filters have been used in recent research to improve the tracking of objects over time. SORT, proposed by Bewley et al.[8], uses a Kalman filter to predict the states of objects and the Hungarian algorithm to associate these predictions with new object detections. Wojke et al.[7] later improved upon this approach with Deep-SORT, which adds a step that uses convolutional neural network-based appearance features for object association. The data association algorithm in Deep-SORT combines the similarity of object appearance features with the Mahalanobis distance between object states, and, for unmatched states, also uses SORT's data association method.

III. Dataset

We will be testing and training Mask RCNN the COCO Dataset and DeepSort Deep Descriptor Network on the Market1501 Dataset.

A. COCO dataset(Mask RCNN)

1) **Images:** 3265 images with one to multiple objects in each image. Shape of each image is 3x300x400.
2) **Labels:** Labels of shape (3265,) which is an array of arrays with all labels in each image

Fig. 1. COCO Dataset Samples

3) **Bounding Boxes:** Shape of bounding boxes is (3265,) where each bounding box is [(no. of objects), 4] in all float32
4) **Masks:** n masks for all images in dataset each with shape 300x400 with one to multiple objects in each image.
5) **Preprocessing:** The dataset images are normalized to [0,1], rescaled to 800x1066, then normalised with mean: [0.485,0.456,0.406], std:[0.229,0.224,0.225] and finally padded to give a final size of 800x1088.

B. *Market1501 Dataset(DeepSort)*

Fig. 2. Market1501 Dataset Samples

1) **IDs:** 32,668 bounding boxes and from stereo video sequences recorded in Marketplace enviornments. Provides robust and easy to formulate Object Tracking Dataset.
2) **Labels:** 178 unique Identities of people along with additional identifiers
3) **Structure:** 0001_c1s1_001051_00.jpg where 0001 is the ID, c1s1 states Camera number 1 and Sequence number 1, 1051 is the total number of frames and 00 is the bounding box.
4) **Preprocessing:** This dataset is modified to a folder based approach where each ID is assigned its own folder containing all its instances, which can be stored as a Pytorch ImageDataset object.

IV. Methodology

A. ***Region Proposal Network(RPN):***

Region Proposal Networks (RPNs) perform a basic yet computationally inexpensive initial assessment of where the bounding boxes of the objects should be as "attention mechanisms" for the object identification job. They were initially proposed as a solution to the problem of costly greedy algorithms like Selective Search. It created new opportunities for end-to-end object detection activities. They sort the initial anchor boxes into object and background categories as well as fine-tune the locations for the object-containing boxes. The instance segmentation heads are further tightened and refined to classify into appropriate categories. The image in Fig 3 shows Region Proposal Network(RPN): Backbone CNN + 2 heads (Regressor head - Bounding box, Classifier head – Object class).

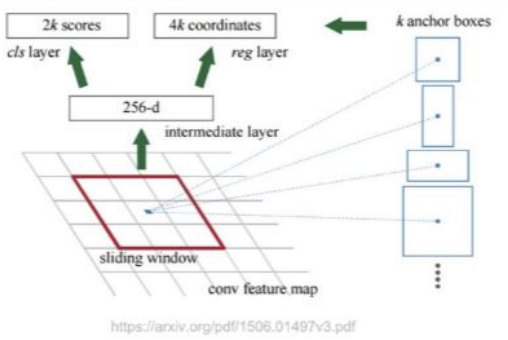


Fig. 3. Detailed RPN

We can see the general structure of the RPN in Figure 4 and the batch of images is passed to the ResNet50 FPN Backbone and then to through the intermediary layer in Figure 2. The output of these will then be forwarded to the RPN heads for classification and regression. The classification head gives a C channel output where C is the number of classes in the data, the regression head gives a 4*C channel output which related to the 3 sets of bounding box coordinates of x,y,w,h wrt to their anchors.

Layers	Hyperparameters
Convolution 1	Kernel Size = (5,5,16), BatchNorm, ReLU, Padding=Same
Max Pooling 1	Kernel Size = (2,2), Stride = 2, Padding=0
Convolution 2	Kernel Size = (5,5,32), BatchNorm, ReLU, Padding=Same
Max Pooling 2	Kernel Size = (2,2), Stride = 2, Padding=0
Convolution 3	Kernel Size = (5,5,64), BatchNorm, ReLU, Padding=Same
Max Pooling 3	Kernel Size = (2,2), Stride = 2, Padding=0
Convolution 4	Kernel Size = (5,5,128), BatchNorm, ReLU, Padding=Same
Max Pooling 4	Kernel Size = (2,2), Stride = 2, Padding=0
Convolution 5	Kernel Size = (5,5,256), BatchNorm, ReLU, Padding=Same

Fig. 4. RPN Backbone

Anchor Box Creation: For each of the 5 FPN levels a specific set of anchor boxes are assigned to each cell of that level. The below table shows the Scale and Stride parameters for the anchor box creation, where each level additionally has 3 sets of aspect ratios of 0.5,1,2 and that combined with the FPN levels assigned scale, give a different set of 3 anchor boxes for each level. An example of this set is given in Figure 3. where the square anchor is from a 0.5 aspect ratio and the horizontal and vertical rectangles are from the 1,2 aspect ratios.

Level:	1	2	3	4	5
Scale	32	64	128	256	512
Grid Size	200,272	100,136	50,68	25,34	13,17
Stride	4	8	16	32	64

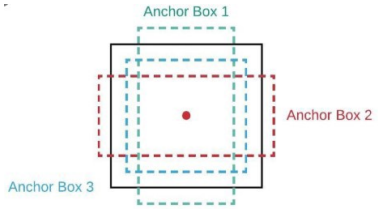


Fig. 5. Anchor Box Set

Ground Truth Creation Creation: Once Anchor Boxes are created and assigned to each grid cell of the FPN, when given a batch of ground truth images we need to assign them to the correct cells to ensure the errors are propagated effectively for training. This way the network learns to assign a loss to the grid cell and correctly identify which class that grid cell should belong to and when these need to be ignored. The predictions from the network are trained to give a encoded bounding box coordinate which are wrt the anchor box assigned to that cell, and thus when creating the ground truth the assigned bounding boxes need to be encoded.

$$t_x = (x - x_a)/w_a$$
$$t_y = (y - y_a)/h_a$$
$$t_w = log(w/w_a)$$
$$t_h = log(h/h_a)$$

Are equations for encoding a given bounding box with coordinated x,y,w,h and anchor box coordinates x_a, y_a, w_a, h_a. When plotting the results the respective decoding can be applied to re obtain the original bounding box coordinates. This is why one must always keep a track of the predictions and their assigned anchor box.

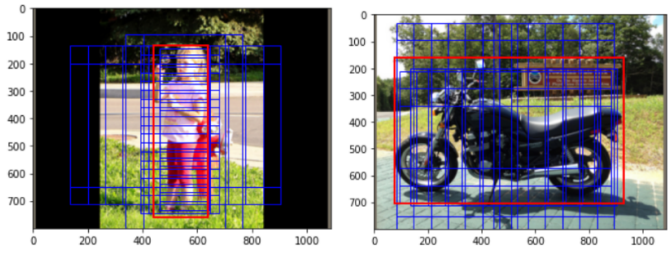


Fig. 6. Ground Truth Assignment

Some of the results of the ground truth assignment can be seen in Figure 6. This shows the anchor boxes assigned for a certain set of image showing how different aspect ratio and scales are present, each active at different levels of the FPN.

Losses: The RPN Network used two types of losses: classifier loss and regressor loss. The classifier's loss is simply the binary cross entropy between the predicted objectness p and the ground truth objectness p. For the regressor's loss we use the Smooth L1 loss between the sets of bounding boxes related to each class. These losses are then equalised to ensure that both are trained properly.

Sample Minibatch: One known problem of anchor based methods are that they suffer from a class imbalance problem, this is because the vast majority of the anchor boxes are assigned as background classes while the only a small percentage relate to detected objects. To tackle this the RPN network employs mini-sampling of the negative and positive anchors. From a given set of detections and ground truths, a set of n/2 samples are selected from the background class and n/2 of the positive class in the ground truth. Their respective predictions are then extracted and the losses are trained on this small minibatch. If there are less than n/2 positive predictions then all positive predictions are taken and n - len(positive) samples are taken from the background class. This ensure that the training is not skewed towards the background classes which can cause the model to not predict any classes so as to minimise loss on background classes rather that positive anchors.

ROI Align: RPN produces proposal boxes for each image

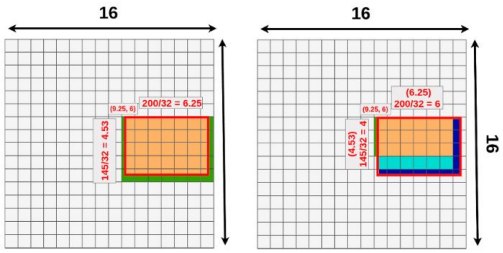


Fig. 7. Comparing RoIAlign(left) and RoIPooling(right))

which contains the coordinates of the upper left and lower right corner of the proposal. This has found to improves mask accuracy by relative 10% to 50%, showing bigger gains under stricter localization metrics.We use only a few proposals with higher confidence score and pass it on to the Box head and Mask head. This decreases the number of computations which needs to be done in the later layers and to ensure training is done on higher scored bounding box proposals. Because the Region Proposal Network's outputs might not always be integers, the associated predicted boxes might not line up exactly with the image. So we used ROI Align function provided by the Pytorch 'torchvision.ops' library and made changes for it to work for multiple scales. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. The image below shows comparison between ROI Align and ROI Pooling.

As shown in Figure 7. the green shaded region on the left indicates the additional data used for pooling. The blue region on the right image shows the data lost when ROI ALign was not used.

B. *Box Head:*

A Box Head refines and classifies the boxes produced by the RPN network. The Box Head processes each proposal independently but in our implementation we stacked them together in a single batch and performed computations in parallel.

Intermediate Layer: We have defined a straightforward 2-layer MLP with ReLU activation functions for the intermediate layer. The flattened feature vector created after the ROI Align is the input of the intermediate layer and the output of both layers are 1024.

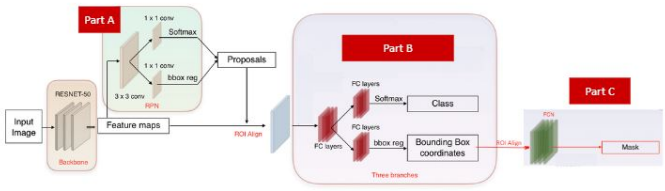


Fig. 8. Workflow of Mask RCNN

From the above Mask RCNN Workflow figure it can be seen that the output of this intermediate layer is send as input to the Classifier and Regressor branch of Box head.

Classifier: Classifier is implemented with a fully connected layer followed by Softmax activation function to output the class probabilities for all C class. We used cross entropy loss between the predicted class probability vector(output of softmax layer) and the ground truth class to calculate classifier loss.

Regressor: Similar to classier, we have implemented a fully connected layer but the output size is 4*C where C is the number of classes. This gives us the upper left and lower right corner of the predicted proposals bounding boxes. For mini batch sampling, we used the same methodology as explained in the RPN section but with a 3:1 ratio for no-background ground truth vs proposals with background ground truth. Smooth L1 loss is used to calculate loss which compares encoded coordinates of the ground truth bounding box with the specific prediction of the regressor that corresponds to the ground truth class for all non background proposals.

C. *Mask Head:*

For Mask R-CNN, the Faster RCNN is extended to include a Mask Head which is responsible for mask generation of all the detected bounding boxes in the Faster RCNN network. Thus it is simple to train and adds only a small overhead to Faster R-CNN. In Mask R-CNN, we re align the proposals from Faster RCNN using Region of Interest (RoI) method with a P of 14. This branch works in parallel with the existing branch for classification and bounding box regression.

The mask branch is a small FCN applied to each RoI, predicting a segmentation mask in a pixel-topixel manner. The architecture can be seen in Figure 7. Thus for Each Region of Interest, Mask R-CNN also outputs a binary mask for each RoI. This is in contrast to most recent systems, where classification depends on mask predictions. FPN uses a top-down architecture with lateral connections to build an in-network feature pyramid from a single-scale input. Faster R-CNN with an FPN backbone extracts RoI features from different levels of the feature pyramid according to their scale, but otherwise the rest of the approach is similar to vanilla ResNet. Using a ResNet-FPN backbone for feature extraction with Mask RCNN gives excellent gains in both accuracy and speed.

V. Results

A. *RPN Network Outputs*

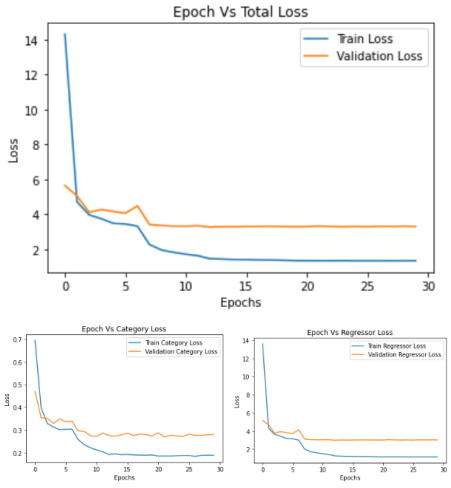


Fig. 9. RPN Network Train and test Curves

Figure 10 shows the results for top 25 results from the RPN model. While this is only used for vizualisation, for the input to the BoxHead network, a parameter preNMS is set to 2000 and postNMS number is set to 1000. The RPN network seems

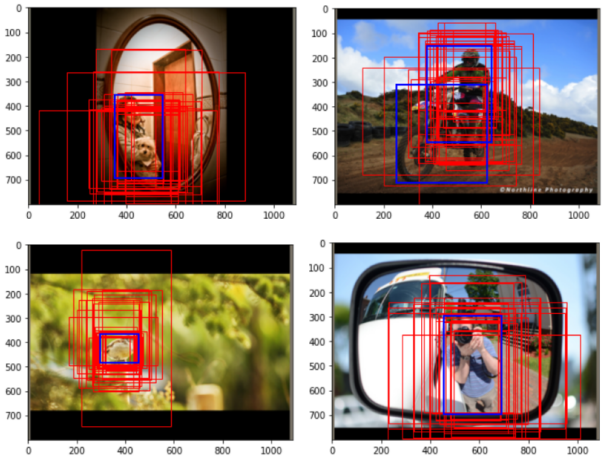


Fig. 10. Top 25 Proposals from RPN Network)

to generate a good list of proposals and can thus be used effectively in the BoxHead and MaskHead network.

B. BoxHead Network Outputs

The RPN Network was used to generate proposals and sent to Boxhead where the model was trained again using the RPN Network and ROI Parameter of 9. Figure 14 shows the Train and Test curves for Boaxhead

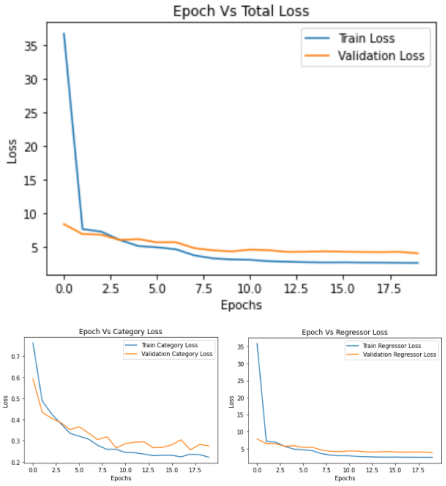


Fig. 11. BoxHead Network Train and Test Curves

The resulting images from boxhead were shown in Figure 15. which shows the results after NMS with the top 20 results after post processing. While this is only used for vizualisation, for the input to the MaskHead network, a parameter preNMS is set to 200 and postNMS number is set to 100.

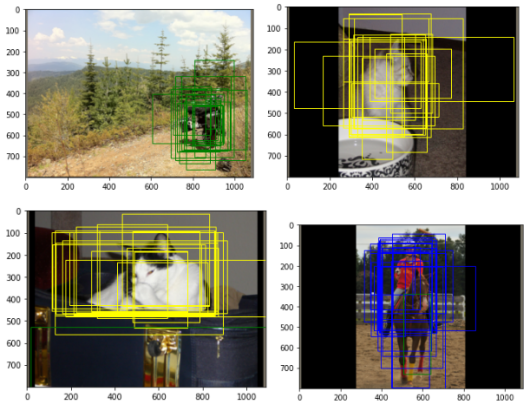


Fig. 12. BoxHead Network Train and Test Curves

C. MaskHead Outputs

The final results of the network were good and were able to correctly classify between classes and multiple instances as well. Figure 16. shows the train and test losses for Mask Head over 17 Epochs.

Figure 17. shows some of the test results from MaskHead showing that the model was successfully able to learn trends and Instances on the Test Dataset.

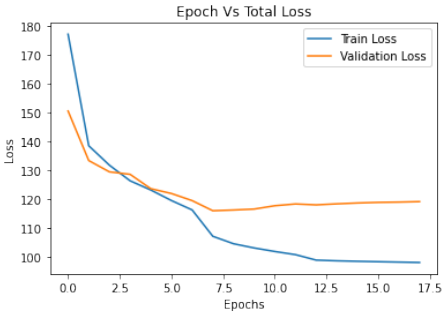


Fig. 13. MaskHead Network Train and Test Curves

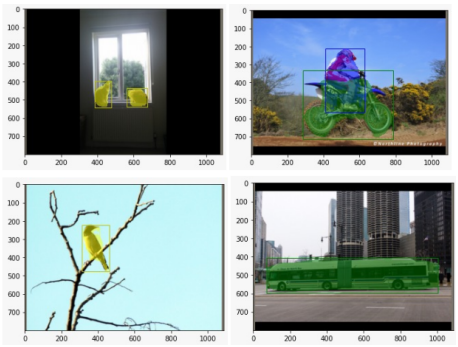


Fig. 14. MaskHead Network Test Results

VI. Novelty: DeepSORT

We wanted to add a another component to our project and worked on MultiObject Tracking with DeepSort.

*A. **Deep Sort Methodology***

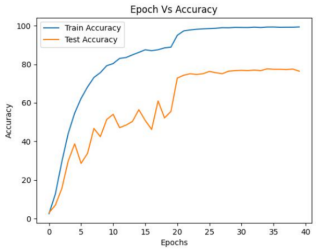
1) Object Detection: The Object Detection will be performed using a pre-trained YOLOv5 network. The detected class was limited to Cars. Initially we attempted to integrate Faster RCNN to the Object Detection Module but the calculation time and results on low lighting areas were relatively poor and hindered the DeepSort Architecture. This is why, to better showcase our work we used a temporary YOLOv5 model which can be switched out for a newly better train faster RCNN, which would be done as a potential Future Work.

2) Kalman Filter: The Kalman Filter is a crucial component of the DeepSORT Architecture. The state is described by 8 variables $(u, v, a, h, u', v', a', h')$ where (u,v) are centres of the bounding boxes, a is the aspect ratio and h, the height of the image.

3) Deep Appearance Descriptor: The descriptor builds a classifier over the dataset, leaving a dense layer producing a single feature vector for classification. This feature vector becomes our "appearance descriptor" of the object.

*B. **Results***

We were able to successfully run the Deepsort object tracking in real time on video sequences from the Speed Challenge dataset. Example dashboard can be seen in the image below.



(a) Deepsort Accuracy

Name	Patch Size/Stride	Output Size
Conv 1	$3 \times 3/1$	$32 \times 128 \times 64$
Conv 2	$3 \times 3/1$	$32 \times 128 \times 64$
Max Pool 3	$3 \times 3/2$	$32 \times 64 \times 32$
Residual 4	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 5	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 6	$3 \times 3/2$	$64 \times 32 \times 16$
Residual 7	$3 \times 3/1$	$64 \times 32 \times 16$
Residual 8	$3 \times 3/2$	$128 \times 16 \times 8$
Residual 9	$3 \times 3/1$	$128 \times 16 \times 8$
Dense 10		128
Batch and ℓ_2 normalization		128

(b) Deepsort Architecture

Fig. 15. Deep Descriptor Accuracy(Left), Deep Sort Architecture

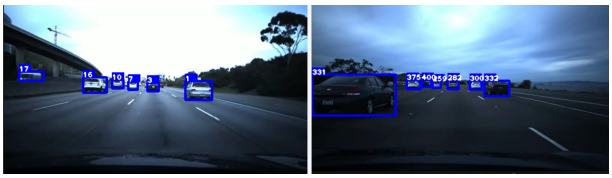


Fig. 16. Example Dashboards

We observed that the dataset performed well on reidentification after occlusions. We can see the working of the DeepSORT algorithm as we track the car with ID 204. The ID does not change even after some occlusions due to other vehicles with ID 217 which crossed the frame of the dashcam. A similar case can be seen on a moving car with ID 623 which is occluded by an overtaking motorcycle(not set to be detected by YOLO). This shows that DeepSort is able to reassign and keep track of objects despite long occlusions.

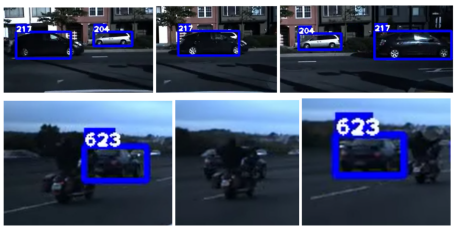


Fig. 17. MaskHead Network Test Results

We also analysed the porcessing time of DeepSort to better understand its potential avenues for improvement. As we can see in figure 18. YOLO takes a fairly similar amount of time to generate bounding boxes across different frames of the video as expected since YOLO is a single shot detector. DeepSORT takes more time to run as the number of objects detected by YOLO increases. This can be attributed to the fact that there are multiple initializations of the Kalman filter simultaneously being updated for each new car which came in the video frame along with prior existing ones.

VII. Conclusion and Future Work

After running the Mask RCNN code we were able to get good results for all instance classes and the resulting masks were able to learn the overall structure of the objects well. From out given approach we were able to realise the benefits

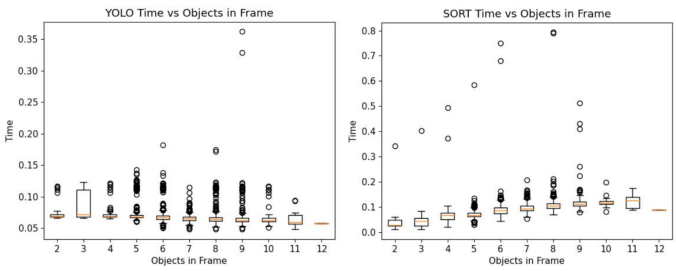


Fig. 18. YOLO and DeepSORT Time Analysis

of modularity. If a better optimisation is found, it is easy to swap out a module and get better results.

While we were unsuccessful with integrating DeepSort with FasterRCNN due to time and efficiency requirements but since we are confident in our implementation of DeepSort itself we would want to pursue making Faster RCNN better tuned for lower lighting and faster processing. This could help us get a better understanding of the process of code optimisation and would be a great use of the modularity of our code which allows us to swap out modules for better ones.

VIII. References

1) He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask R-CNN. In: ICCV. (2017):
2) Girshick, Ross and Donahue, Jeff and Darrell, Trevor and Malik, Jitendra, Rich feature hierarchies for accurate object detection and semantic segmentation, 10.48550/ARXIV.1311.2524, arXiv 2013.
3) Learning to Segment Object Candidates via Recursive Neural Networks, Chen, Tianshui and Lin, Liang and Wu, Xian and Xiao, Nong and Luo, Xiaonan, 10.48550/ARXIV.1612.01057, arXiv 2016
4) Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: NIPS. (2015) Vol 28.
5) Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, Bernt Schiele: The Cityscapes Dataset for Semantic Urban Scene Understanding, 10.48550/ARXIV.1604.01685, arXiv 2016.
6) Gkioxari, Georgia and Malik, Jitendra and Johnson, Justin, Mesh R-CNN, arXiv 2019.
7) Wojke, N.; Bewley, A.; Simple Online and Realtime Tracking with a Deep Association Metric. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Beijing, China, September 2017
8) Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; Upcroft, B. Simple online and realtime tracking. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016