ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY of WASHINGTON

Multiple Object Tracking

E E 443 Machine Learning for Signal Processing Applications

Authors:
Ara Vardanyan, Stefan Arsov

Submitted to:
Dr. Jenq-Neng Hwang

June 2022

1 Introduction

Multiple Object Tracking (MOT) is a computer vision task that aims to analyze videos in order to identify and track objects belonging to one or more categories, such as pedestrians, cars and inanimate objects without any prior knowledge about the appearance and number of targets. MOT algorithms associate a target id to each object detected, classify the object, infer a bounding box, and use a tracking algorithm to predict trajectories of the object. Multiple Object Tracking extends computer vision to many important applications such as autonomous cars, robot navigation, medical imaging, security surveillance, crowd behavior analysis, and more.

Our task in this project was to design and implement a Multiple Object Tracking system to track cars and pedestrians. We were provided with a subset of the KITTI dataset which only considers the classes 'Car', 'Pedestrian', and 'DontCare'. Our implementation consisted of the YOLOv5s detection architecture and model pre-trained on the COCO dataset, a Siamese neural network to obtain the embedding features of cars required for DeepSort, and DeepSort (Simple Online and Realtime Tracking with a Deep Association Metric).

In this report, we will discuss our experience with Multiple Object Tracking as pertaining to our methodology and implementation, challenges faced, results achieved, and future work to be done on the project.

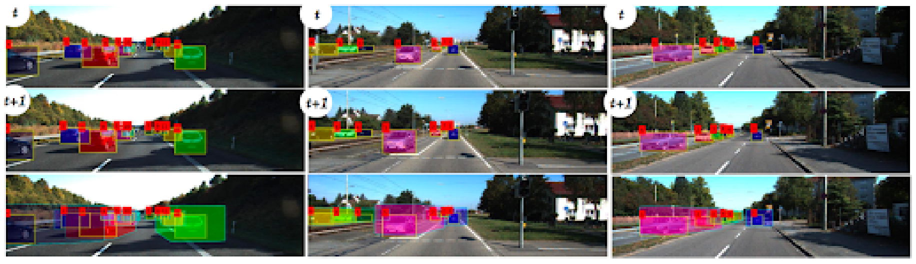


Figure 1: An illustration of the output of an MOT algorithm on KITTI dataset. Each output bounding box has a number (track id) that identifies a specific object in the video.

2 Methodology

2.1 Detection with YOLO5vs

The first step necessary for tracking objects is detecting them and classifying what type of object they are. For this objective, We chose to use a model built on the YOLOv5s architecture and pre-trained on the COCO dataset. We then trained this model's weights even further on our sub-KITTI dataset. YOLOv5 is the leading architecture in realtime object detection due to its speed, higher average precision than other architectures, and easy deployment making it an obvious choice for object detection and classification needs.

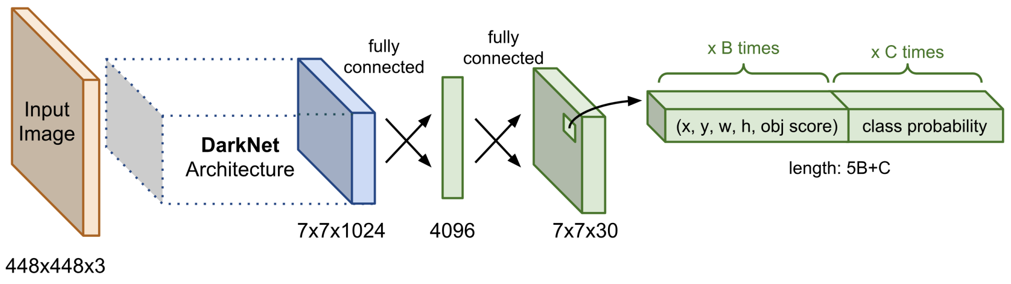


Figure 2: YOLOv5 network architecture and outputs

The YOLO algorithm works by dividing the image into N grids, each having an equal dimensional region of SxS. Each of these N grids is responsible for the detection and localization of the object it contains. These grids then predict B bounding box coordinates relative to their cell coordinates along with the object label and probability of the object being present in the cell.

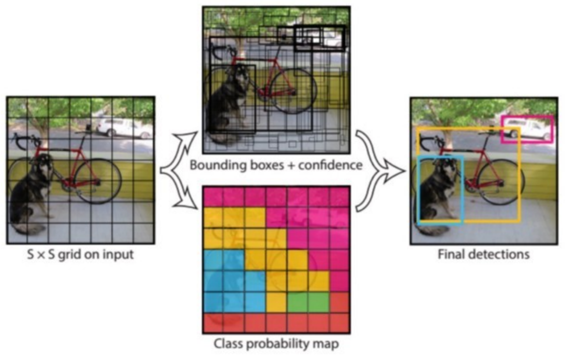


Figure 3: YOLOv5 Visualization

YOLOv5 uses a stochastic gradient descent as the default loss function based on objectness score, class probability score, and bounding box regression score. Below we can see the results achieved by our YOLOv5 model pre-trained on the COCO dataset and custom trained on our sub-KITTI dataset. Note the high precision score but low recall score. This means our model does a good job at classification and creating bounding boxes when it detects an object, however, the low recall denotes it does not produce as many results as expected. These results were not expected and the goal for future iterations of this project would be to increase recall in our YOLOv5s model.

2.1.1 YOLOv5s results

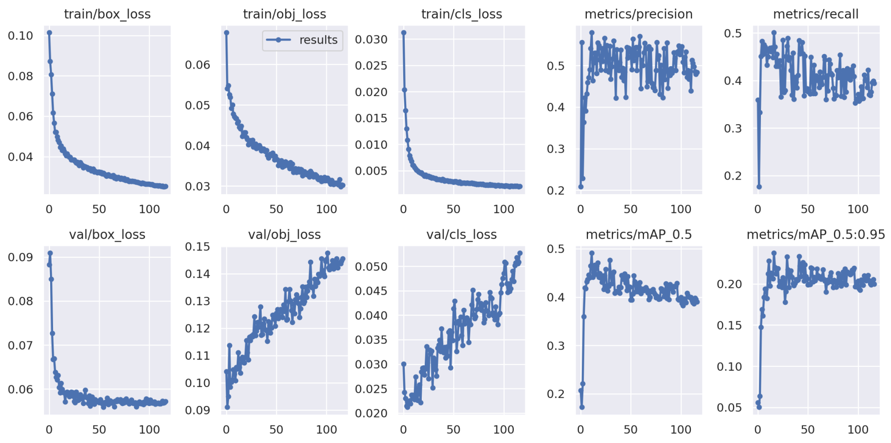


Figure 4: YOLOv5s Metrics on Validation Set

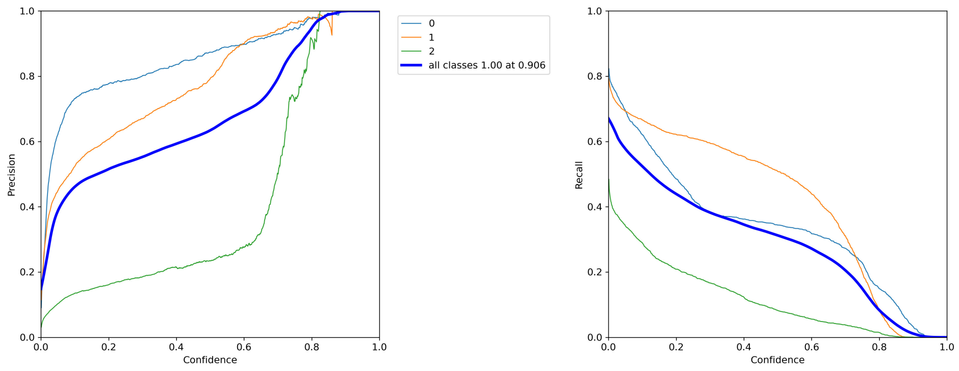


Figure 5: Precision / Confidence and Recall / Confidence - Model Metrics

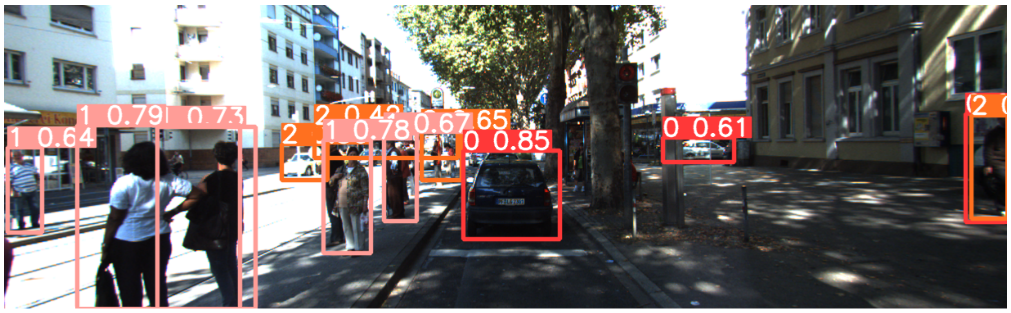


Figure 6: Image ran through YOLOv5s model, 0: Car, 1: Pedestrian, 2: Dont-Care

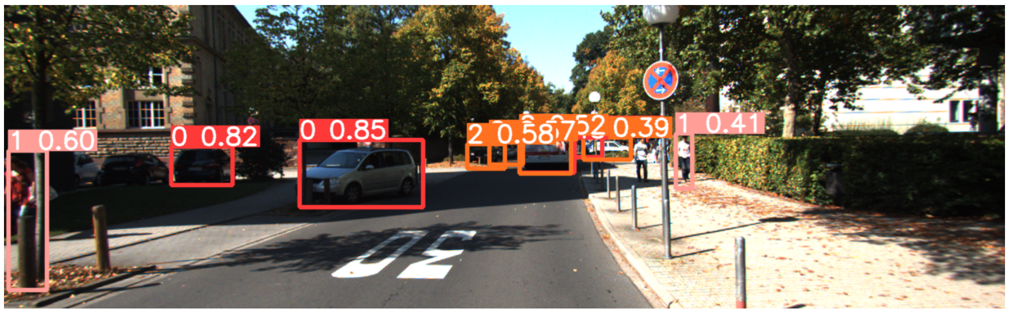


Figure 7: Image ran through YOLOv5s model, 0: Car, 1: Pedestrian, 2: Dont-Care

2.2 Feature Extraction with Siamese Neural Network

Our next step before tracking objects was obtaining the embedding features of cars necessary for DeepSort. DeepSort uses these features to help identify if a currently detected object has been detected previously. This helps combat ID switching, an important metric for judging MOT accuracy. We decided to use the Siamese Neural Network (SNN) for extracting embedding features of cars. SNN requires a small data set to train, perfect for our project. SNN uses a triplet loss function which compares a reference input (anchor) to a matching input (called positive) and a non-matching input (called negative).

SNN allows for an object to be passed into the model and then the model discriminates whether the object is the same object or not with very high accuracy by creating and comparing domain specific features. In our case, whether or not the object passed in matches the features expected with the same object.

We trained our SNN by passing in cropped images of the same cars but from different angles and perspectives into the training model to build the discriminating properties within the model. The goal would be that no matter the perspective the car is seen from, it should all be discriminated to 'Yes this is the same car'. This plays a role in matching an object in the frame to one previously already identified.

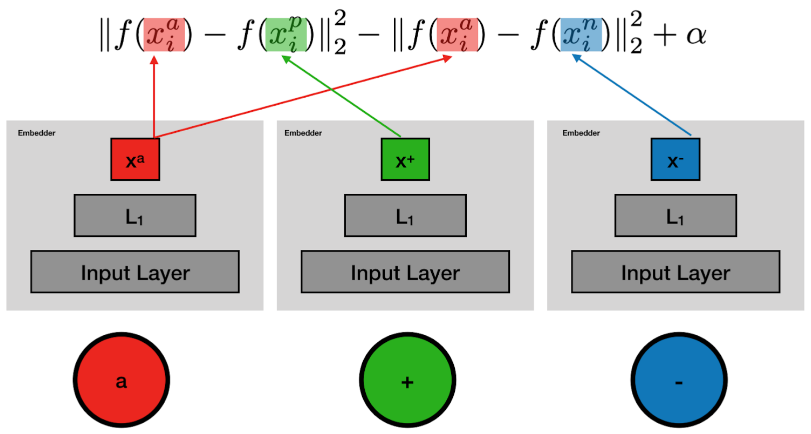
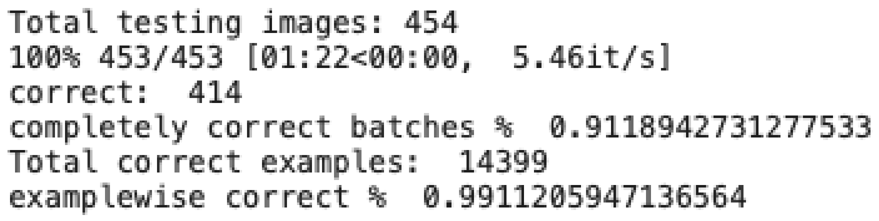


Figure 8: Triplet Loss

2.2.1 Siamese Neural Network Model Results



```
Total testing images: 454
100% 453/453 [01:22<00:00,  5.46it/s]
correct:  414
completely correct batches %  0.9118942731277533
Total correct examples:  14399
examplewise correct %  0.9911205947136564
```

Figure 9: Validation data results on trained Siamese model

As seen from the results, the discrimination accuracy from the Siamese model is very high, ultimately being very effective for our use case.

2.3 Tracking with DeepSORT

After detection, classification, and feature extraction we can finally complete our objective of tracking objects. We chose to use DeepSORT for this task due to its great abilities in dealing with occlusion which allow DeepSORT to be the best multi-object tracking machine learning model. DeepSORT deals with occlusion using a Kalman filter among other things, a filter used to estimate states based on linear dynamical systems in state space format. It assumes a constant velocity model and Gaussian distribution and an object's location is estimated by its model of motion. The more occlusion there is, the more weight is given to the motion model's prediction. The less occlusion there is, the more weight is given to sensor data. As stated earlier, DeepSORT creates a track id for each object detected and uses the Siamese network feature extractor for track id association. This means when a previously seen object is occluded and reappears from the occlusion, they will ideally maintain their previously assigned track id. DeepSORT also uses the the Hungarian algorithm (also known as Kuhn-Munkres) for association. The Hungarian algorithm associates an object from one frame to another based on scores associated with intersection over union, shape, and convolution cost. Eventually, after an object has been occluded for enough frames, the object is assumed to no longer exist and the track id along with all state information about the object is deleted to preserve memory and speed for greater performance. Despite the Kalman filter and the Hungarian algorithm, our model can still fail to do its job. Thus, DeepSORT strips off the final classification layer of the model and instead uses the final dense layer as a single feature vector to serve as the appearance descriptor of an object. This combined with a simple distance metric

$$D = \lambda * D_k + (1 - \lambda) * D_a$$

where

$$D_k$$

is the Mahalanobis distance

$$D_a$$

is the cosine distance between the appearance feature vectors and

$$\lambda$$

is the weighting factor, allows for elegant and efficient object tracking.

2.4 Final Tracking Video Results

- First video output from the testing data on the modified KITTI-set: DeepSORT Output-1.
- Second video output from the testing data on the modified KITTI-set: DeepSORT Output-2.

3 Conclusion

Our implementation of Multiple Object Tracking did function very well after many hours of work. We are proud to have a fully working YOLOv5s model, and a Siamese Feature Extractor for the objective. We also have good results to show for it. Unfortunately, while our DeepSORT worked well and produced outstanding video results, we could not obtain accuracy metrics for our model. This result gives us many key takeaways from this multi-faceted project. As our first large scale machine learning research project, we had no point of reference to begin from and as such, made some crucial mistakes in our implementation that affected our working processes. Our implementation involved using open-source code and this is where things got messy. We were not very organized and did not come into the project with a great plan and outline to support our implementation. We had a 'figure it out as you go' approach which seems to have been inevitable considering this was our first exposure to a project of this gravity. This approach, while not optimal, did teach us a lot about the technical details of Machine Learning implementation. However, it will not be sustainable for future research.

We plan to go into future research of this type with organized directories and a detailed plan. Another challenge we faced specific to this project was data management. YOLOv5 expected a different data format than we were provided. As such, we had to write many scripts for data formatting to comply with the requirements. Luckily, we were able to do so successfully after many hours of confusion and we improved our skills in data management and curation throughout this challenge. Another side effect of the details of implementation that tripped us up was less research into the topic and no novel ideas to experiment with. Ideally, we would have proposed and experimented with novel ideas in efforts to improve final MOTA metrics. Again, we were not able to due to the nuances of the implementation. Luckily, we have come out of this research project as better engineers with more machine learning experience. We are ready to tackle projects of equal and greater caliber. For that, we would like to declare this research project a great learning experience and a great success!

4 References

1. Benhur, Sean. “A Friendly Introduction to Siamese Networks.” Built In, https://builtin.com/machine-learning/siamese-network
2. Kanjee, Ritesh. “DeepSORT - Deep Learning Applied to Object Tracking.” Medium, Augmented Startups, 9 May 2022, https://medium.com/augmented-startups/deepsort-deep-learning-applied-to-object-tracking-924f59f99104.
3. Koch, Gregory, et al. Siamese Neural Networks for One-Shot Image Recognition, https://www.cs.cmu.edu/ rsalakhu/.
4. Maiya, Shishira R. “DeepSORT: Deep Learning to Track Custom Objects in a Video.” AI amp; Machine Learning Blog, AI amp; Machine Learning Blog, 24 Apr. 2020, https://nanonets.com/blog/object-tracking-deepsort/.
5. Solawetz, Jacob. “How to Train yolov5 on a Custom Dataset.” Roboflow Blog, Roboflow Blog, 17 May 2022, https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/.
6. Weng, Lilian. “Object Detection Part 4: Fast Detection Models.” Lil’Log (Alt + H), 27 Dec. 2018, https://lilianweng.github.io/posts/2018-12-27-object-recognition-part-4/.
7. Wojke, Nicolai, et al. “Simple Online and Realtime Tracking with a Deep Association Metric - Arxiv.” Arxiv, Arxiv, https://arxiv.org/pdf/1703.07402.pdf.
8. https://github.com/adambielski/siamese-triplet
9. https://github.com/ls-da3m0ns/Multiple-Object-Tracking
10. https://github.com/abhyantrika/nanonets_object_tracking