Utilizing spatio-temporal information to improve object detection in video data

Richard Mathews II
Oden Institute

Abstract

In this paper, I will discuss the importance of spatio-temporal information when performing object detection on video or time-varying image data, demonstrate how to handle video data, propose potential data structures for training recurrent neural networks, and present an algorithm that can improve object detection in any time-sequence visual data by considering short-term spatio-temporal information. I used video data taken from the dashboards of vehicles driving in urban streets. The object detection model I used is a Keras implementation of state-of-the-art RetinaNet by Facebook AI Research. The goal of this project was to investigate the incorporation of previous timestep information in increasing object detection in video data. The resulting algorithm is built for a RetinaNet model pretrained on object detection for an urban street setting but can be applied to time-varying imaging.

1. Introduction

Most of the progress in computer vision has centered around object detection and semantic segmentation in images. For image classification, popular networks have been ResNet [1], VGG Network [2], and GoogleNet [3]. We have seen strong image segmentation architectures such as FCN [4], SegNet [5], UNet [6], and PSPNet [7].

When it has come to video data, the most common approach has been to deploy fast object detection algorithms on each frame of the video, such as YOLO [8] and RetinaNet [9]. While this approach is effective, there is certainly room for improvement. By performing fast object detection frame-by-frame, all of the previous timestep information is lost, and each timestep is just a brand-new image to the object detection algorithm. In recent years, there has been increased focus on utilizing spatio-temporal information in computer vision tasks for video. FlowNet [10] showed that convolutional neural networks are capable of estimating optical flow, the perceived pixel-by-pixel motion in images. SegFuse [11], a semantic video segmentation competition hosted by MIT, presented the challenge of building a dynamic perception model using spatio-temporal information.

There can be many approaches to solving the problem of dynamic object detection in time-varying visual data. [12] proposed convolutional recurrent neural network architectures to perform feature aggregation across time for electrocardiogram classification. Implementing temporal features into the training of a model is certainly a solid approach, and one being taken.

I approached this problem by considering the results of an object detection algorithm in previous timesteps as information to assist the algorithm's predictions of the current timestep. I propose two concepts: 1) YOLBO (You Only Look Back Once), an algorithm that utilizes object detection results from the previous timestep, and 2) an ensemble of an object detection algorithm and LSTMs trained on object movement. For 1), I present the concept, code, implementation, and results, and for 2), I present the concept and a data structure for training the LSTM model on object movement.

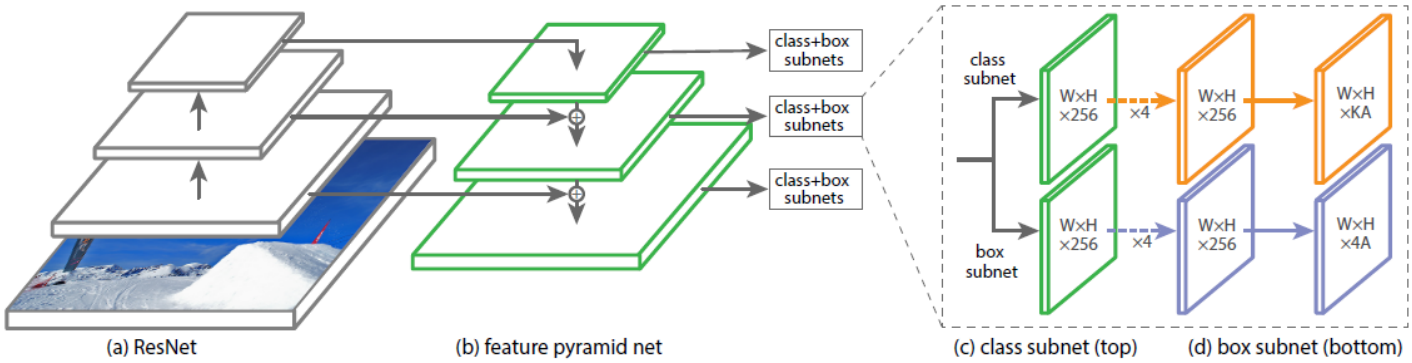

Figure 1 – The components of RetinaNet are a) ResNet backbone, b) Feature Pyramid Net on top of the ResNet backbone, c) classification subnet for each FPN level, d) box regression subnet for each FPN level.

2. Object Detection Algorithm

There are several state-of-the-art object detection algorithms that have shown impressive average precision for small, medium, and large objects. I took YOLO and RetinaNet under consideration because of their impressive mAP scores, and ultimately decided to use RetinaNet for this project, but a YOLO implementation could potentially be in the works.

2.1 RetinaNet Overview

The reason one-stage detectors have fallen short of two-stage detectors is because of foreground-background class imbalance. The advantage of the one-stage approach is its speed and simplicity, but until RetinaNet, the accuracy of these detectors has been inferior to two-stage detectors, such as R-CNN [13]. Facebook AI Research devised a novel loss function, Focal Loss, to resolve the class imbalance problem. To test the effectiveness of Focal Loss, they designed RetinaNet, which has been shown to match the speed of other one-stage detectors while beating the best two-stage detectors in accuracy.

2.2 Components of RetinaNet

RetinaNet consists of a backbone, two subnetworks, and a Feature Pyramid Network [14] on top of the backbone (Figure 1). The ResNet backbone generates a feature map across an image using convolutional feature extraction, and the FPN constructs a multi-scale feature pyramid for detecting objects at different scales. For each FPN level, there is a fully convolutional network for predicting the probability of each object class at each spatial position and a fully convolutional network for bounding box regression.

2.3 RetinaNet Implementation

For this project, I used a Keras implementation of RetinaNet [15] in the Python language. From the same source, I used model weights pretrained on the COCO dataset with a ResNet-50 backbone. There are 80 object classes for the model to detect, with some of the more common ones for an urban street driving setting being 'car', 'person', 'truck', 'bicycle', and 'traffic light.'

For this research paper, I investigated how RetinaNet detections from previous frames could be used to accurately detect objects in the current frame that did not meet the 0.5 threshold due to lighting, angle, interference, or other anomalies.

3. Data

The dataset used for this project was Berkeley DeepDrive [16], which is a data repository of 100,000 HD video sequences taken from the dashboard cameras of vehicles driving in urban streets. This dataset has been widely used by researchers in the autonomous vehicle space for perception tasks such as road object detection, segmentation, driveable area, and lane markings.

4. YOLBO (You Only Look Back Once)

Inspired by the You Only Look Once (YOLO) object detection algorithm, the YOLBO algorithm only takes information from the previous timestep into account, hence the name. The advantages of YOLBO over recurrent neural nets are: 1) no training required, 2) simplicity (no parameter tuning, simple implementation), 3) very fast (good for live object detection with cameras), 4) can work in conjunction with any pretrained state-of-the-art object detection algorithm on any time-varying image data where spatial information is of importance. The disadvantages of YOLBO compared to RNNs are: 1) performance is limited by the quality of the object detection algorithm, 2) does not capture deeper patterns in video sequences, 3) can extend the duration of wrong detections.

4.1 Concept

The core idea behind YOLBO is if RetinaNet is unsure about a detection in the current frame but was confident about a similar detection in the previous frame, then the detection is most likely valid. A typical video is 30 frames per second, so the timestep of each frame is 1/30 seconds. The fact that objects are not going to move very far in a 1/30 second timestep is an important spatio-temporal concept, one that is leveraged to make accurate detections using the detections scoring less than 0.5 from RetinaNet. Figure 2 shows a real-world demonstration of this in action.

In Figure 2, we start with an annotated frame in a video sequence at timestep t, and 3 frames later, there are 4 more detections in the YOLBO frame than that of the standard frame. These are not additional detections, but rather, they are poor RetinaNet detections that did not meet the 0.5 threshold. What YOLBO has done across this t→t+3 time period is looked back at the confident detections from the previous frame and compared them to the detections in the current frame determined by RetinaNet to be unlikely. If the object class is the same and the spatial location is similar, YOLBO rescores the detection with a high degree of confidence (score > 0.5). Essentially, if RetinaNet thinks there might be a car at pixel location *(x, y)*, and YOLBO informs RetinaNet that 1/30 seconds ago there was definitely a car near pixel location *(x, y)*, then RetinaNet decides there is definitely a car there.

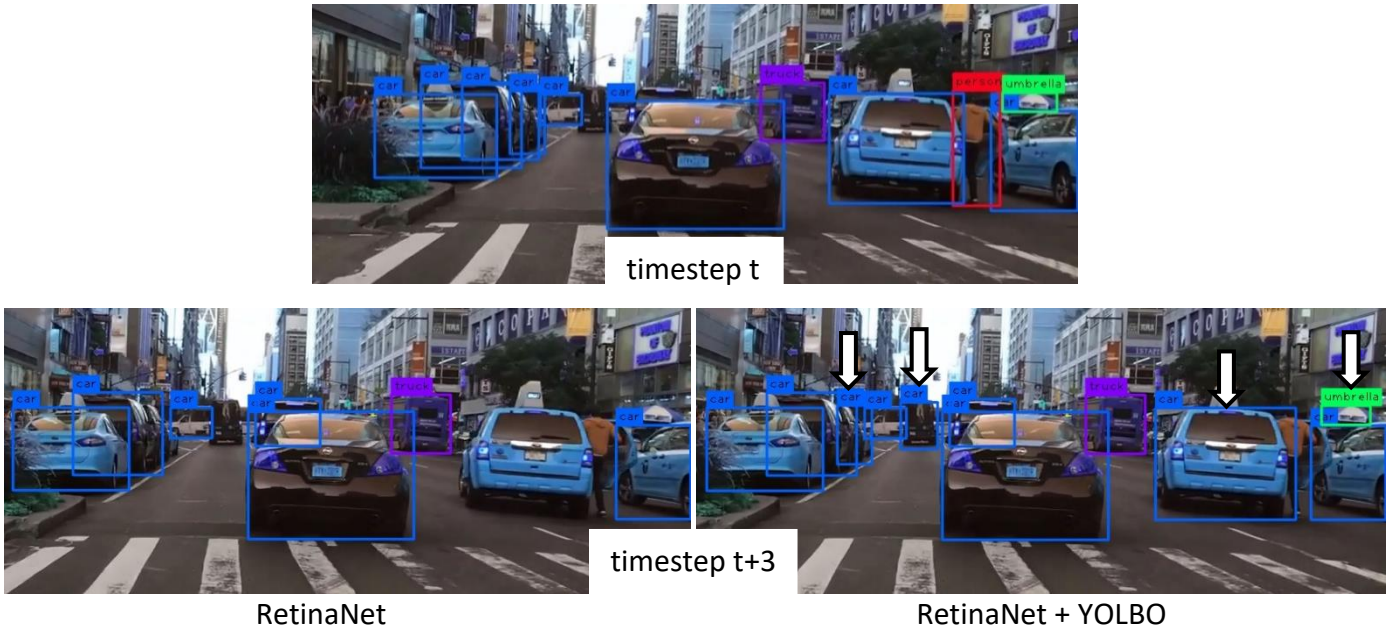

Figure 2 – Demonstration of the increase in object detection by implementing the YOLBO algorithm on top of RetinaNet.

4.2 Algorithm

For every frame, RetinaNet makes a significant number of detections. Setting the threshold for what constitutes a valid detection involves a tradeoff between the total number of detections and the accuracy. Lowering the threshold may result in more valid detections, but also more invalid detections. The score deemed to be the most effective threshold is 0.5. YOLBO is able to effectively identify which of the many detections scored less than 0.5 are actually valid by utilizing spatio-temporal information.

Detection Matrix: The key data structure used in the YOLBO algorithm to store and reference spatio-temporal information is the detection matrix. Given an image, or in the case of video data, a frame, RetinaNet returns three numpy arrays: bounding boxes, scores, and labels. The index of these arrays represents the detection, so for the i^{th} detection, the pixel coordinates of the bounding box are the i^{th} index of the boxes array, the score is the i^{th} index of the scores array, and the label is the i^{th} index of the labels array. These outputs serve as inputs to the YOLBO algorithm, which constructs the detection matrix, a 3-dimensional matrix of dimensions n x h x w, where n is the total number of object classes, h is the height of the video frames in pixel units, and w is the width of the video frames in pixel units. For my data, the detection matrix has a size of 80 x 720 x 1280, since there are 80 object classes that RetinaNet-ResNet50 will detect and the resolution of the Berkeley DeepDrive videos are 720 x 1280. For each detection, the score is mapped to the spatial location of the bounding box center on the object class layer corresponding to the label of the detection (Figure 3). Once all the detections have been mapped to a spatial class layer, the layers are compiled into a 3-dimensional matrix, which contains information on all detections including the pixel location in the frame, the score, and the label. This detection matrix is constructed for each timestep, or frame, and so over time, this data structure is a nice container of spatio-temporal information which the algorithm can use to filter out the valid detections scoring less than 0.5.

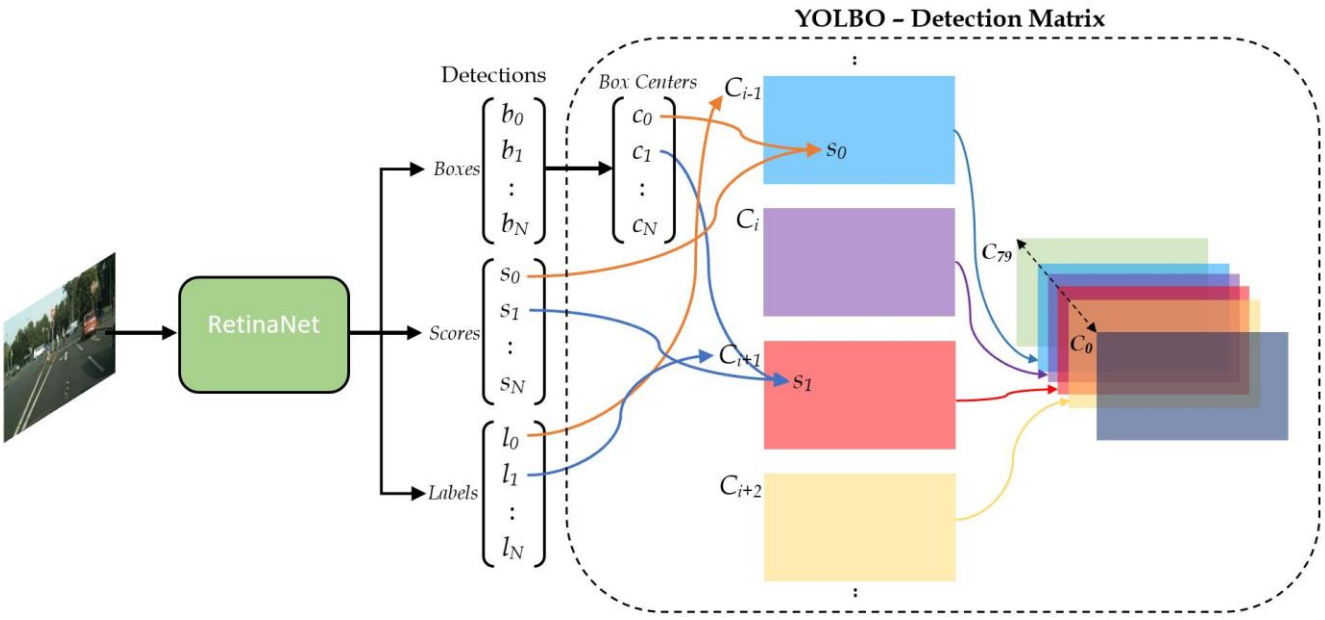

Figure 3 – The data structure used in the YOLBO algorithm is the detection matrix, a set of spatial layers that the RetinaNet detections are mapped on to. There are N detections, consisting of bounding boxes (represented as pixel locations), scores, and labels. The centers of the bounding boxes, c, are calculated and the indices of the spatial layers correspond to the object classes {C}. The box centers and the labels are used to map the scores to a spatial layer, where $l = C$, at the location where the detection occurred.

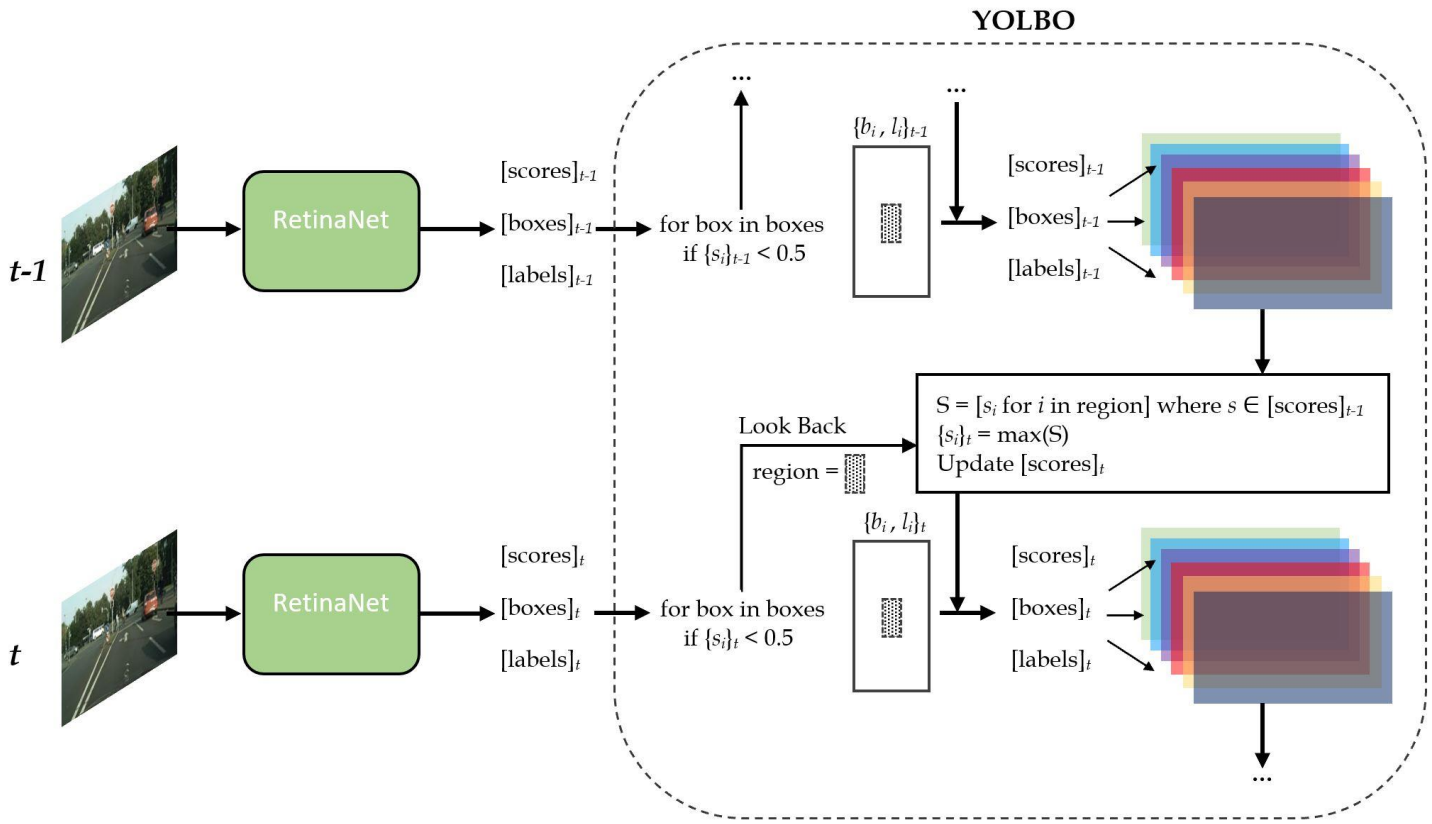

Figure 4 – The YOLBO algorithm utilizes a Look Back function to scan for similar detections in the previous timestep. For each detection of the current frame scoring less than the 0.5 threshold, the Look Back function scans a small region around the center of the box in the previous detection matrix of the corresponding spatial layer and gathers all the scores into a list of scores, *S*. The max value from *S* will replace the detection score.

Look Back Function: When constructing the detection matrix of the current timestep, YOLBO utilizes a Look Back function to scan the detection matrix of the previous timestep for similar detections. If a detection scores less than the threshold, 0.5, the Look Back function is triggered (Figure 4).

A scan region is generated around the center of the bounding box and is of the same height-width ratio but at a much smaller scale. The Look Back function pulls the relevant spatial layer (based on the label) from the previous detection matrix and finds the max score in this scan region and sets this value as the new score of the detection. As the algorithm iterates over each detection, utilizing the look back function to update the scores based on similar detections from the previous timestep, a new scores array is generated. For each detection scoring less than 0.5 where a similar detection was made in the previous timestep but at a much higher confidence (score > 0.5), the algorithm will replace the score with that of the similar detection.

The new scores array, along with the boxes and labels, serve as inputs in generating the detection matrix of the current timestep, which ultimately will be used in the Look Back function of the next timestep. What this essentially does is leverages spatio-temporal information to find valid detections scoring low due to anomalies. In the scenario where RetinaNet is unsure about a detection because of lighting, interference, angle, or any anomaly, the YOLBO algorithm will look back at the previous frame to see if RetinaNet made a similar detection in the same spatial area with high confidence, and if it did, then a logical assumption can be made that the object is still there and hasn't moved very far in 1/30 seconds.

4.3 Code

The project was implemented in the Python programming language using OpenCV API to handle video data, TensorFlow and Keras for machine learning, and Numpy for the detection matrix. The code is in a public repository on Github here. For a demonstration of how to use the repo on video data, see the demo.ipynb file here and run it in Google Colab. The weights and demo video can be downloaded from the releases page. The algorithm is in the object_detection python file here, and the Look Back function is in the look_back python file here. For an understanding on how to handle video data for object detection, refer to the video_detection python file here which provides a function that allows for one to utilize the YOLBO algorithm by setting the yolbo parameter to True.

4.4 Results

RetinaNet is a high accuracy, state-of-the-art object detection model, so any improvements to the volume of detections made in a video sequence can be challenging.

For testing, the backbone of the RetinaNet model was ResNet50 pretrained on the COCO dataset with the highest accuracy possible. The COCO dataset includes object classes commonly seen in urban street driving, such as cars, trucks, buses, people, traffic lights, etc.

There were 7 sample videos used to test the algorithm. The only variable parameter in the YOLBO algorithm is the center box ratio, which is the ratio of the size of the scan region to the size of the bounding box. Results were collected for a center box ratio of 1/5 (Table 1) and 1/3 (Table 2).

The results show a strong improvement in the volume of detections, with an increase between 1-5% for a center box ratio of 1/5 and 1.5-6% for a center box ratio of 1/3. Since these additional detections are based on similar detections from the previous frame, it can be assumed the accuracy is that of RetinaNet. Figures 5-7 show examples of frames with more detections as a result of the YOLBO algorithm. These results clearly show there is value in spatio-temporal information.

Table 1 – Summary of results for YOLBO algorithm on 7 sample videos using a center box ratio of 1/5.

Street Video	Additional Detections	Additional Detections Per Second	RetinaNet Detections above 0.5 score	% Increase
A	143	14.3	3667	3.9
B	53	5.3	4591	1.2
C	104	10.4	4734	2.2
D	61	6.1	2602	2.3
E	83	8.3	1941	4.3
F	100	10	3154	3.2
G	95	9.5	4071	2.3

Table 2 – Summary of results for YOLBO algorithm on 7 sample videos using a center box ratio of 1/3.

Street Video	Additional Detections	Additional Detections Per Second	RetinaNet Detections above 0.5 score	% Increase
A	170	17	3667	4.6
B	77	7.7	4591	1.7
C	159	15.9	4734	3.4
D	90	9.0	2602	3.5
E	104	10.4	1941	5.4
F	134	13.4	3154	4.2
G	126	12.6	4071	3.1

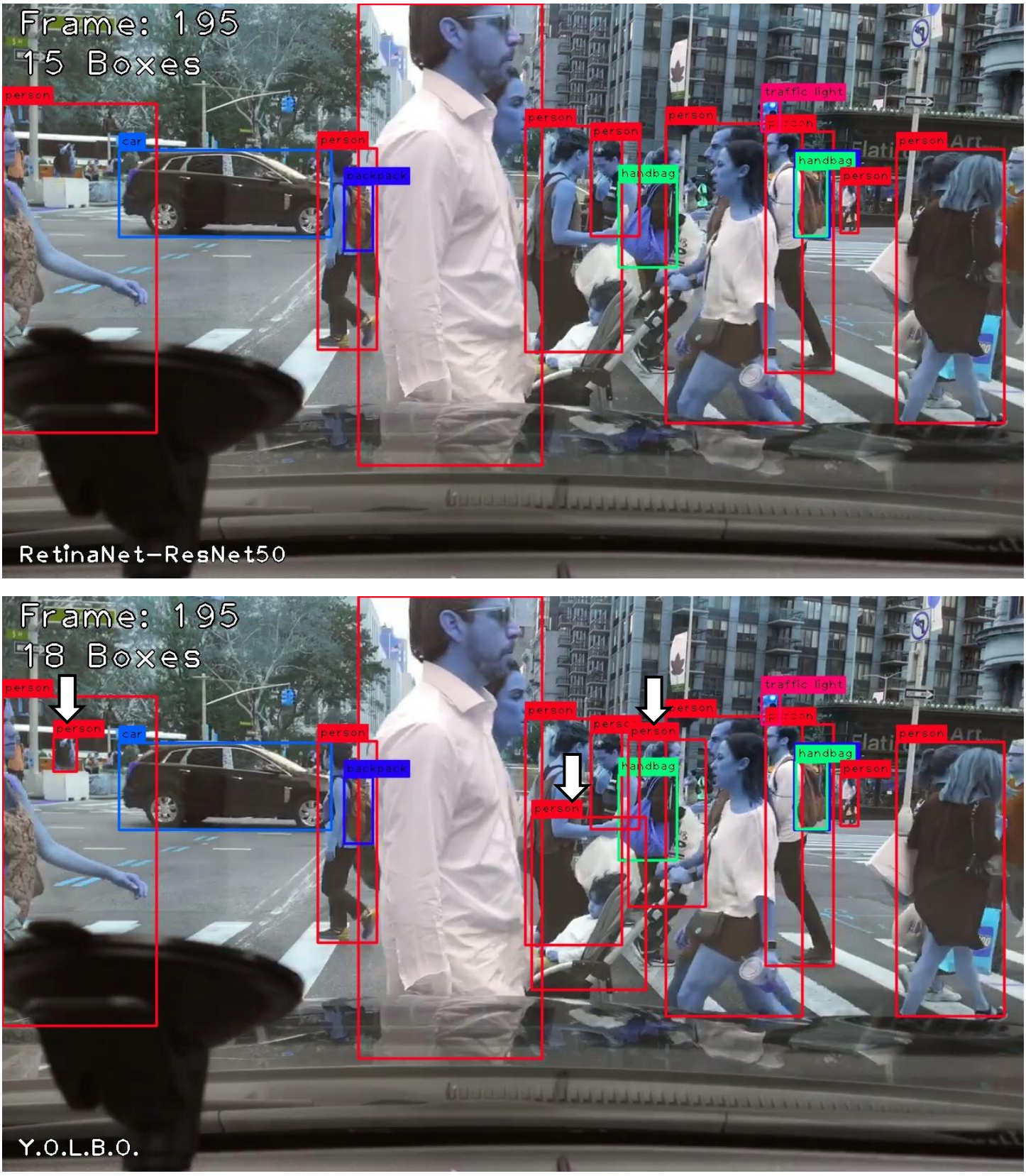

Figure 5 – Frame 195 taken from video sample C comparing RetinaNet (top) to RetinaNet with YOLBO (bottom).

Figure 6 – Frame 76 taken from video sample D comparing RetinaNet (top) to RetinaNet with YOLBO (bottom).

Figure 7 – Frame 66 taken from video sample G comparing RetinaNet (top) to RetinaNet with YOLBO (bottom).

5. LSTM-RetinaNet Ensemble

Rather than incorporating a recurrent component within the RetinaNet model, it could be interesting to see if recurrent neural nets can learn how detections vary spatially over time. If this learning is possible, there could be an LSTM-RetinaNet ensemble, where inputs to the LSTM would be all detections from the RetinaNet model from all previous timesteps, and the LSTM would make a prediction on where certain detections will be located based on movement patterns. These predictions would feed into a CNN-based model that would find the most likely detections in the spatial areas predicted by the LSTM to have certain detections, and these detections coupled with the RetinaNet detections would combine in some mathematical operation or activation function to yield a final output.

I propose a data structure for training the LSTM on to learn specific object movement. In this project, I demonstrate how the data structure would look for learning vehicle movement, people movement, and stationary movement from the perspective of a moving vehicle in urban streets. The outputs of the RetinaNet model would be filtered for scores greater than 0.4 and encoded into a structure similar to the detection matrix, but instead of mapping scores to the spatial layers, the area of the bounding box would be mapped instead. The area is mapped because this captures the relative depth of the detection (larger areas mean the object is closer to the camera), and depth is a factor in movement from the perspective of the viewer. The spatial layers with objects corresponding to one of the three movements are added together to form a new 3-dimensional movement matrix (in code, the matrix is 4 dimensions, but conceptually only the first three matter). This movement matrix is then max pooled using a 1x10x10x1 filter with a stride of 1x10x10x1. This reduces the resolution of the frames by a factor of 10, so the general spatial area of each detection is known. Then, the for each of the three movements, the matrix is reshaped so the low-resolution 2-dimensional spatial layer becomes a single spatial vector, represented as the rows in a new 2-D matrix. As the video runs, each frame goes through this process, and the result is appended to a 3-D data structure, so the depth is now the timesteps, the rows are the spatial vectors for each type of movement, and the columns are general spatial areas on the frame. The spatial vector contains information on where detections are and how far away they are (depth).

To generate training data, the rows of a tensor must represent timesteps in a sequence, and the depth of the tensor represents the batch size, or number of sequences to train the model in each run. For each movement, the data structure is restructured into this tensor. The labels are spatial vectors from 30 timesteps after the last timestep in each sequence (the goal for the LSTM is to predict what the spatial vector will look in 0.5 seconds). Since RetinaNet occasionally misses detections in a frame, the time steps in the labels vector are max pooled across 4 timesteps. Each batch is a 2 second video sequence, which becomes 60 timesteps (frames). Starting at t=0, every 60 timesteps in a video becomes a sequence. Additionally, starting at t=30, every 60 timesteps becomes a sequence. This allows for greater utilization of each video for training and reduces computational cost. Generating this data structure is very complex and required careful analysis of matrix indices, but the final result is here.

If an LSTM, or any recurrent neural net, is able to learn object movement, using the data structure proposed or any other, it could predict the general spatial area of a detection including the class and relative depth. A CNN-based model could then process this area and find the most likely detection that fits the detection predicted by the LSTM. Then, there could be two sets of detections combined into one, theoretically allowing for greater accuracy and volume of detections in video data.

6. Conclusion

Spatio-temporal information can be a valuable tool in computer vision. By just utilizing spatial and detection information from the previous timestep, YOLBO has shown 1% to 6% increase in detections to a state-of-the-art model. There are many paths forward in computer vision with video data. The use of deep learning and recurrent neural networks can be a powerful tool in learning how objects move and understanding deeper patterns in spatio-temporal information. These models could be deployed as part of the object detection models or on top of these models to find patterns in those detections.

There are improvements that can be made to YOLBO, such as measuring accuracy of the additional detections brought by YOLBO, such as metrics like IoU and AP, to understand the algorithm's effectiveness in greater detail, or testing YOLBO with other object detection models, such as YOLO, to see if the improvements are better or worse compared to RetinaNet. Training LSTMs using the data structure I have developed to see if time series AI algorithms could learn the movements of detections such as vehicle movement, people movement, and stationary movement is a path I find promising. An LSTM-RetinaNet ensemble model would come with incredibly high computational cost, but there is potential for learning movement patterns in video data and using these spatio-temporal patterns to dramatically improve accuracy and volume of object detection in video data.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.

[2] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,pages 1–9,2015.

[4] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. arXiv preprint arXiv:1411.4038, 2014.

[5] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. arXiv preprint arXiv:1511.00561, 2015.

[6] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv preprint arXiv:1505.04597, 2015.

[7] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid Scene Parsing Network. arXiv preprint arXiv:1612.01105, 2016.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. arXiv preprint arXiv:1506.02640, 2015.

[9] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal Loss for Dense Object Detection. arXiv preprint arXiv:1708.02002, 2017.

[10] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning Optical Flow with Convolutional Networks. arXiv preprint arXiv:1504.06852, 2015.

[11] SegFuse: Dynamic Driving Scene Segmentation. https://selfdrivingcars.mit.edu/segfuse/

[12] M. Zihlmann, D. Perekrestenko, and M. Tschannen. Convolutional Recurrent Neural Networks for Electrocardiogram Classification. arXiv preprint arXiv:1710.06122, 2017.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv preprint arXiv:1311.2524, 2013.

[14] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature Pyramid Networks for Object Detection. arXiv preprint arXiv:1612.03144, 2016.

[15] Fizyr. Keras implementation of RetinaNet object detection, 2017-2018. https://github.com/fizyr/keras-retinanet

[16] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. arXiv:1805.04687, 2018. https://bdd-data.berkeley.edu/