Semi-supervised Learning for Self-Driving Cars: Predicting Bird's Eye View Bounding Boxes and Binary Roadmap Images using UNet and CNNs

Aja Klevs Micaela Flores

Abstract

In this project, we train a model that uses images captured by six different cameras attached to a car to generate a top down view of its surrounding area. We aim to predict not only a bird's eye view binary roadmap of the entire area, but also the bounding boxes of the nearby vehicles and objects. Our implementation, though it draws heavily from existing semantic segmentation methods such as UNet, yields a hybrid approach that integrates semi-supervised learning techniques to utilize all available data. We visualize our model's prediction capabilities for both tasks, noting that it achieves comparatively decent performance on both the bounding box and roadmap tasks in the final model leaderboard.

1. Introduction

In computer vision, object detection involves not only classifying objects, but also identifying their respective locations within an image. It is an incredibly relevant task in today's society as the world of autonomous vehicles continues to grow. With this comes an ever-heightening need to mature current methods and technologies in order to ensure the safety of the vehicles, passengers, and pedestrians involved. In this project, we develop a semi-supervised convolutional neural network to hopefully improve upon predicting vehicle bounding boxes and roadmap layouts for self-driving cars.

2. Literature Review

We explore existing methods and previous work in this area, focusing especially on semi-supervised efforts in identifying bounding boxes.

2.1. YOLO

YOLO (You Only Look Once) is a real-time object detection algorithm that capitalizes on a single convolutional neural network pass, where the image is split into a grid of cells and each cell predicts a bounding box and class probabilities (Brownlee, 2019). These bounding box predictions are then consolidated into a final bounding box for each of the detected objects in the image. This approach differs from prior detection systems in that it does not apply the model to multiple regions and scales within an image, predicting the region with the highest score. YOLO is superior to these alternative methods as it looks at the entire image at test time, meaning the global context of the image informs its predictions. Moreover, it also makes predictions with a single network evaluation unlike systems like R-CNN which require thousands for a single image. This ensures incredibly fast runtime, making YOLO more than 1000x faster than R-CNN and 100x faster than Fast R-CNN (Redmon et al., 2016).

Our one hesitation with YOLO is that there currently does not appear to be a functionality that transforms dashboard camera images into their corresponding bird's eye view representations. However, we adopt a similar strategy when predicting the bounding boxes for our dataset by attempting to predict the existence of an object on each pixel. As with YOLO, our model looks at the the data only once in each forward pass.

2.2. UNet

UNet is a semantic segmentation method that was originally developed for biomedical imaging. The model first identifies objects within the image through its contractive encoder and then localizes objects in the image using its expanding decoder. The UNet encoder begins by employing traditional convolutional and max pooling layers to shrink the input, followed by its decoder upsampling with transposed convolutional layers to increase output resolution (Ronneberger et al., 2015). As seen in Figure 1, the large number of feature channels during upsampling allows context information to be propagated up to higher resolution layers, giving the network its characteristic U shape. Finally, localization occurs when the high resolution features from the contracting step are combined with the upsampled result, concluding with a convolution layer that learns a more precise output based on this information (Ronneberger et al., 2015).

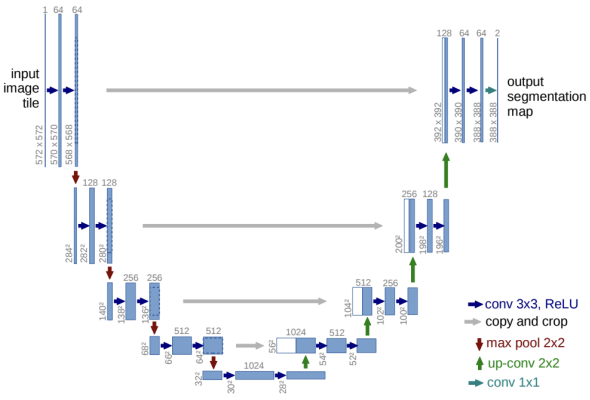

Figure 1. Out-of-the-Box UNet Architecture. Each blue box corresponds to a multi-channel feature map, the number of channels is on top of the box, the x-y-size is at the lower left edge of the box, and white boxes represent copied feature maps. Note that our implementation uses different image input/output sizes, while still retaining 2 output channels for a binary result.

The UNet model is particularly advantageous in vision applications in that it is completely end-to-end convolutional, meaning it can accept image of any size and achieves excellent performance with limited training data. Moreover, it exhibits superior performance over prior object detection and localization methods as these often sacrifice localization accuracy for context and vise versa (Ronneberger et al., 2015). As such, UNet provides a very promising approach to our binary roadmap and bounding box prediction tasks, especially as we have comparatively less labeled data than unlabeled. This model serves as our primary backbone architecture and achieves somewhat reasonable average threat scores for both tasks, especially for our first attempt.

2.3. Deep Generative Models for Semi-Supervised Learning

Generative models, trained specifically on unlabeled data to predict latent factors, create a way to augment classification tasks that have few labeled data instances and many unlabeled instances (Kingma et al., 2014). In generative models, each layer learns a representations of the input, which can treated as embeddings that encode information about the data. After these layers are trained on the large unlabeled dataset, the generative models can then obtain useful embedded representations of the labeled data inputs. Those representations are passed through a supervised classification model and fine-tuned for the desired task. The introduction of generative models can increase the performance of a purely-supervised classification model, sometimes drastically.

The effectiveness of this kind of semi-supervised approach varies greatly depending on the data, task, and model architectures of both the generative model (which we refer to as the feature extractor) and the supervised classification model. While semi-supervised transfer learning techniques are generally much more successful in natural language processing applications than in computer vision ones, we still attempt to build and utilize a generative feature extractor here.

3. Methodology

3.1. Initial Model: Supervised Learning

In the early stages of model development, we simply averaged the six 256×306 dashboard images into one 256×306 input image to pass into the model. The averaged image provides a decent average threat score for both methods, particularly for the binary roadmap prediction.

3.1.1. Binary Roadmap Prediction

In our initial implementation for the binary roadmap prediction task, we essentially used the out-of-the-box UNet model with only a single minor adjustment. In the original code, the output of the final layer is upsampled to be the same dimensions as the original image. For our purposes, we need the network to return a binary image of dimension 800×800 in order to compare our predictions to the true binary roadmap label. As such, the final upsampled dimensions are changed to reflect the desired dimensions, leaving the remainder of the existing implementation in tact. Figure 2 exhibits a particularly successful roadmap prediction. With only the output upsampling dimensions changed, our initial model achieves a threat score of 0.7036 against the first submission test script.

3.1.2. Bird's Eye View Bounding Box Detection

Our initial attempt at the bounding box task is remarkably similar to our binary roadmap approach. We convert the bounding box labels into 800×800 binary images, where the road is represented by zeros and any pixel occupied by an object is represented by a one. As seen in Figure 3 , we

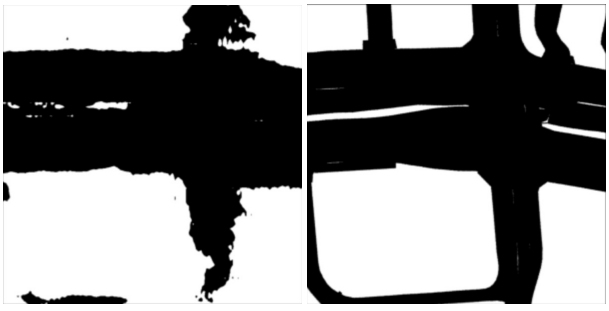

Figure 2. Example of binary roadmap prediction (left) and its corresponding label (right)

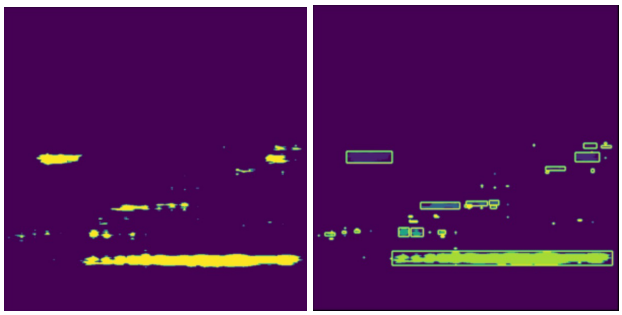

Figure 3. Example of our bounding box prediction (left) and the bounding boxes drawn around the predictions using `skimage measure` (right)

utilize the exact same UNet architecture from the roadmap model to predict whether there exists an object in each pixel in our 800×800 prediction. We then use the `skimage measure` package to draw bounding boxes around our predictions, returning the coordinates of the bounding boxes as our final result. Our initial supervised model, trained on the entire labeled dataset, achieves an average threat score of 0.00134 on the first submission test set.

3.2. Final Model: Semi-supervised Learning

In the later iterations of our model, we experimented with using a concatenation of all six images into one larger 512×918 input image, hopefully preserving as much detail as possible from all six original images (Figure 4). While we didn't have time to fully test the difference in performance between both input types, our preliminary experiments indicate that the averaged image performs better on the purely supervised model. As such, we assume it would also achieve better performance with the semi-supervised approach, so we use the averaged image our final models.

Figure 4. Example of concatenated sample input

3.2.1. Training the Feature Extractor

To incorporate the unlabeled data into our model, we train a feature extractor on this data to create potentially useful representations for our UNet classifiers. While we experimented with different feature extractors, we settle on a neural network with three convolution layers, three max pools, and two fully-connected linear layers. Lastly, we train the averaged input images on black and white versions of themselves, training for three epochs until a plateau in the mean square error loss emerged.

3.2.2. Binary Roadmap Prediction and Bounding Box Detection

For our both our semi-supervised roadmap predictions and bird's eye view bounding box detection models, we first pass the average dashboard images through the previously-trained feature extractor, and then pass the output from the feature extractor through the UNet architecture. The only change we make to the existing UNet model is the number of input channels, which changes from 3 to 1. Using a technique with which we saw success in previous academic projects, we perform one *frozen* epoch epoch and several *unfrozen* epochs until validation loss converged. By this we mean that we do not allow back-propagatation or weight updates on the feature extractor during the first epoch, but we do allow them beginning with the second epoch. Therefore, though both the roadmap and bounding box problems utilize the same architectures, our feature extractor and UNet models have now been tailored to each of the two tasks after the semi-supervised training has finished. The detailed procedure is visualized below in Figure 5.

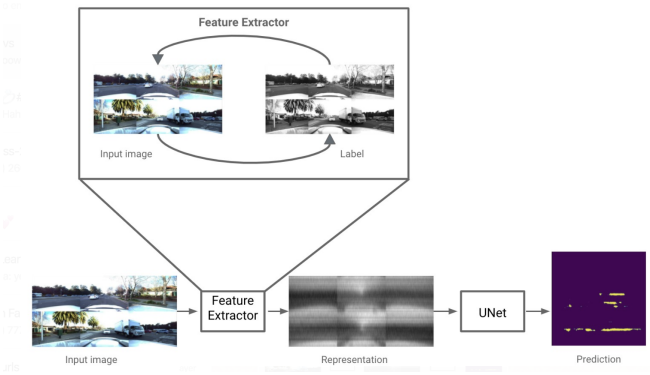

Figure 5. Our semi-supervised learning approach, where the input image is passed into the trained feature extractor before getting passed into UNet for prediction.

4. Results

The performance of the model is determined by its ability to detect objects (like car, trucks, bicycles, etc.) and its ability to draw the binary roadmap layout. For both objectives, a variation of average threat score is used as the evaluation metric, where the bounding box detection is determined by

$$Final\ Score = \sum_t \frac{1}{t} \cdot \frac{TP(t)}{TP(t) + FP(t) + FN(t)}$$

where $t \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ and roadmap reconstruction determined by

$$TS = \frac{TP}{TP + FP + FN}$$

The results for the three iterations of our implementation are summarized in Table 2.

Table 1. Average test threat scores for both bounding box and binary roadmap tasks.

SUBMISSION	BOUNDING BOX SCORE	ROADMAP SCORE
ROUND 1	0.001344	0.7036
ROUND 2	0.000564	0.7042
FINAL ROUND	0.001	0.6697

Our results for the first two rounds are deceptive as we only introduced a train/validation split for the final round. For Round 1, we used the purely supervised models trained on all labeled data, which gave us relatively high threat scores on both tasks.

For Round 2, we attempted a semi-supervised bird's eye view bounding box detection model that was fine-tuned on the entire labeled dataset. While we were getting a threat score as high as 0.005 on the training set, our results on the test set suggest that we were severely over-fitting. Moreover, we also forgot to *softmax* our predictions for this second submission, which is probably another reason our bounding box score is so low. We submitted the same purely supervised roadmap model for both the first and second rounds.

Determined to correct our over-fitting, we make a train/validation split in the labeled data, retraining all models on just 85% of the labeled data. We then use the validation set to tune our prediction thresholds for both tasks, settling on pixel thresholds of .6 for roadmap and .21 for bounding boxes. The validation results of both the supervised and semi-supervised models for the final round submission are shown below.

Table 2. Average validation threat scores for supervised and semi-supervised models.

MODEL TYPE	BOUNDING BOX SCORE	ROADMAP SCORE
SUPERVISED	0.0008727	0.6324
SEMI-SUPERVISED	0.001123	0.6838

5. Conclusion

While we did not have the time for all of the experiments we would have liked to try, our results give us a good idea of potential next steps. In the end, the semi-supervised models performed noticeably better than the purely supervised model, though the difference was not drastic. We tried a completely different, much larger, feature extractor for Round 2, and given its high threat score on the training data, we have some evidence this could have been a better direction. Unfortunately, the unintentional neglect of the *softmax* function in our second submission prevented us from knowing the true test results for this larger feature extractor.

Furthermore, it would also be potentially fruitful to try training the feature extractor with different latent factors (perhaps a rotated version of the images themselves, or even the number of bounding boxes). If given more time, we would ideally like to try hyper-parameter tuning with the layer dimensions.

In all, a generative approach might be fundamentally the wrong choice for this task, as is the case with many other computer vision problems. A recent approach called Contrastive Predictive Coding (CPC), which seeks to build representations as far apart as possible in latent space, seems more promising (van den Oord et al., 2019). Given more time and resources, we would have sought to implement a version of CPC, and compare with our semi-supervised results.

References

Brownlee, J. How to perform object detection with yolov3 in keras. *Machine Learning Mastery*, 2019.

Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. Semi-supervised learning with deep generative models. In *Neural Information Processing Systems (NIPS 2014)*, Montreal, Canada, 2014.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *arXiv*, 2016.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W., and Frangi, A. (eds.), *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2015)*, Munich, Germany, 2015.

van den Oord, A., Li, Y., and Vinyals, O. epresentation learning with contrastive predictive coding. In *arXiv*, 2019.