Yandhi: A Simple Multitask Approach to Bounding Box and Road Map Classification

Vishwaesh Rajiv (vr1059) Ben Stadnick (brs426) Tony Xu (tx507)

Abstract

We utilize a YOLO-inspired approach toward the prescribed vehicle detection and road map detection tasks. We re-frame both tasks as classification problems and find that when trained in a multitask fashion, the model gives good performance on the road map task.

1. Literature Review and Related Work

There are various successful object detection frameworks, such as YOLO (Redmon et al., 2015) and Faster R-CNN (Ren et al., 2015). Both methods assume ground-truth bounding boxes for relevant object categories on the input image itself. YOLO (V1) is a fully-convolutional method that splits the input image into grid cells. For each cell, the model predicts the object confidence of that cell (if that cell contains an object), a class (among C classes) for that cell, and two predicted bounding boxes, not necessarily contained within the grid cell itself. The closest bounding boxes to the ground-truth labels are chosen, and the loss function consists of optimizing the bounding box predictions, correct class predictions for each cell, as well as correct confidence score for each cell (whether there is an object or not in that cell).

The road map task can be seen as a segmentation task, as each pixel in the 800 x 800 grid needs to be classified as road or not-road. U-Net (Ronneberger et al., 2015) is a commonly-used architecture for pixel-level segmentation on input images: each pixel is classified between C classes. The architecture itself consists of an encoder (that downsamples the input image into a set of lower-dimensional feature maps) and a decoder that upsamples those feature-maps into a segmentation map of the same dimension as the input. The architecture also syncs each step of the down-sampling and up-sampling operations: feature maps from the down-sampled operation are concatenated to the representations in the up-sampling operations. This results in better fine-grained pixel-level segmentation performance.

Another paper that we looked into (Palazzi et al., 2017) utilized a model that took in synthetic annotated video game camera views from Grand Theft Auto V and warped them into a bird's eye view. Due to a lack of annotated bounding boxes on the camera images themselves, we were not able to directly leverage the architecture of the model, but we were able to take inspiration from their coordination prediction approach while developing our own methods.

2. Methods

2.1. Vehicle Detection

In the vehicle detection task, rather than directly regress the bird's-eye view bounding box coordinates from the camera views, we re-frame it as classification problem. We predict whether or not a given block in the 800 x 800 bird's eye view contains a vehicle. We also predict the total count of vehicles from the camera views, and use this during inference to select blocks with the highest probability to contain vehicles.

We use the following approach:

1. We divide the 800 x 800 bird's eye view into user-defined blocks.
2. We assign the block a label of 1 or 0 whether the center of the bounding box falls inside that block.
3. We predict total count of vehicles.

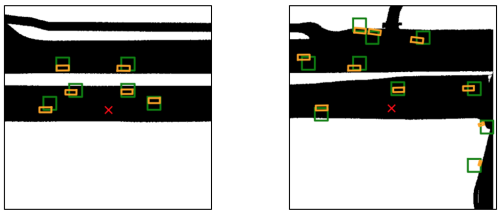

Figure 1. Examples of original bounding box targets and modified grid block targets

We used a block size of 50px x 50px, of which there are 256 in the grid. Thus, the bounding box regression problem now becomes a multi-label classification problem. We used binary-cross entropy for the multi-label block classification

(BCEWithLogitsLoss in PyTorch) and cross-entropy loss for the counts prediction. During inference, the total count of vehicles is used to find the blocks with the highest probabilities of vehicles inside of them. Bounding box generation is deterministic: for each of the selected blocks, a box is drawn in the middle third of the block, and its coordinates are used to calculate the IoU (Intersection over Union) score.

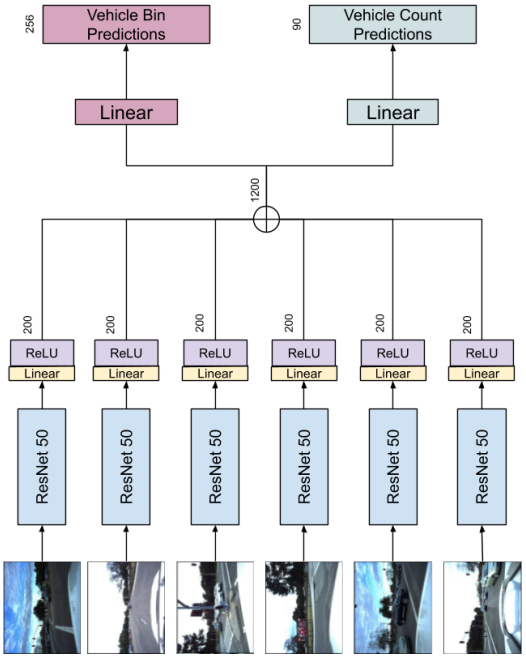

Figure 2. Stand alone vehicle detection model

2.2. Road Map Detection

In the road map detection task, we attempt to predict whether each pixel in the 800 x 800 bird's eye view image is part of the road or not. We can think of this as a binary classification problem on a 640,000 dimensional vector. A classification problem of this size is difficult for a neural network to handle. In order to make the problem more tractable, we simplify it in the following manner:

1. We divide the 800 x 800 bird's eye view into blocks.

2. We assign each block a label of 1 or 0 based on the majority class of the pixels within the block.

We try to predict the values of the blocks in this new target road map instead of the values of each of the pixels in the original one. Examples of the modified targets using blocks of size 100 x 100 pixels and 5 x 5 pixels along with the original target road map can be seen in Fig. 3. We see that by using larger blocks, the modified target is much coarser compared to the original target. However, by making the blocks small enough, we recover the majority of the fine-grained information in the original target road map while reducing the size of the binary classification problem by an order of magnitude.

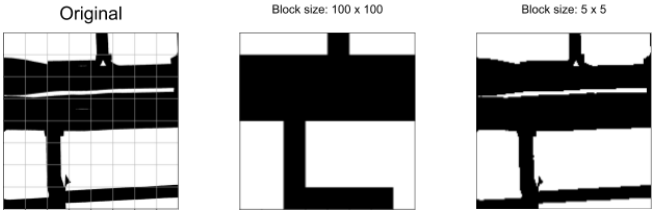

Figure 3. Original road map target and modified targets for block sizes of 100 and 5

The architecture for the road detection task is shown in Figure 4. Each camera view is fed through a ResNet-50 backbone (He et al., 2016) and a linear layer to obtain a 200-dimensional vector representation. These representations are concatenated together and fed through another linear layer to obtain a 25,600-dimensional vector, which is then fed through a sigmoid nonlinearity to obtain probabilities. Here each component of the output vector represents the probability that a 5 x 5 pixel block is part of the road. The final 800 x 800 road map can be reconstructed from this 25,600 dimensional vector. This model is trained using binary cross entropy loss. Results for different block sizes can be seen in Table 1.

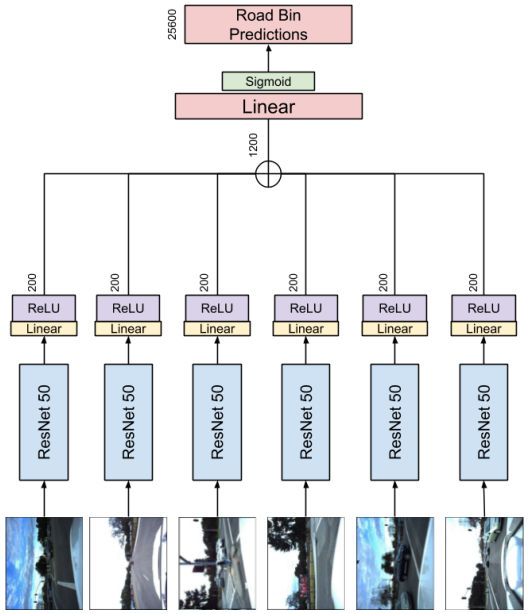

Figure 4. Stand alone road detection model

Block Size	Threat Score
5	0.8212
10	0.8279
25	0.8073

Table 1. Road detection threat scores for different block sizes

2.3. Self Supervised Learning

In order to leverage the unlabeled dataset, we experimented with the self-supervised Jigsaw pretraining task (Noroozi & Favaro, 2016). In this task, we extract a random 225 x 225 patch from a view. This patch is then divided into a 3 x 3 grid. Each cell in this grid is 64 x 64 pixels. Finally, these cells are shuffled and fed to a network that tries to predict the permutation applied in order to recover the original image. To limit the size of the problem, we limit the possible permutations to just 250. Noroozi & Favaro (2016) show that the Hamming distance between permutations affects the difficulty of the task. Following the algorithm in their paper, we select 200 permutations with maximal Hamming distances between one another. We then randomly select 50 permutations from those left over. By making the task more difficult, we hope that the network will learn more meaningful representations. For this task we use a ResNet-50 backbone without batch norm that we will be able to transfer to our other models. An outline of the jigsaw task is shown in Figure 5.

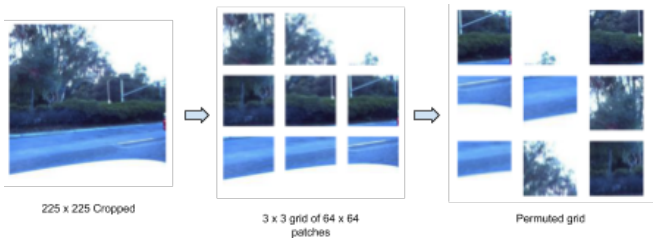

Figure 5. Jigsaw task

In our experiments, we did not see any improvement when fine-tuning from this task, so our approach ended up being fully-supervised.

2.4. Multitask Model

Since both of the tasks rely on the same architecture, a natural extension to try and improve performance is to combine the two tasks into a multitask model where both tasks can be trained simultaneously. As mentioned in Ruder (2017), by creating a multitask model, we hope that information used in one task will benefit the other task and vice versa. For instance, if the vehicle detection branch of the network predicts a vehicle at a certain location in the bird's eye view map, then it is most likely to be on or near a road. By training both tasks together, the vector representation of the six camera views will have this information encoded and allow the road map branch of the network to respond accordingly. We train this multitask model with the following loss function:

$$\ell_{multitask} = \ell_{road\_map} + 2\ell_{count} + \ell_{vehicle}$$

. where ℓ_{road_map}, ℓ_{count}, and $\ell_{vehicle}$ are the losses used in the stand alone models. The final multitask architecture is shown in Figure 6.

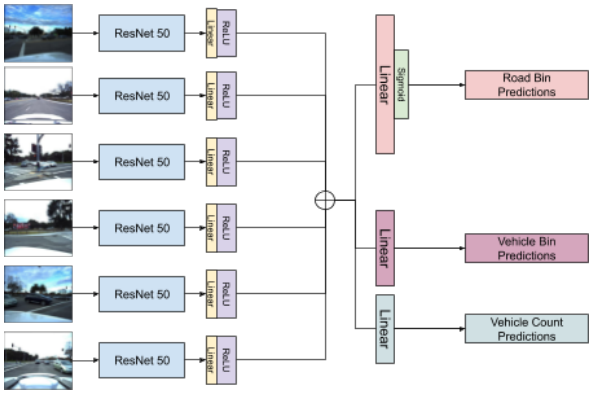

Figure 6. Multitask model

3. Results

The results for both the stand-alone models and multitask model on our validation set are shown in Table 2. The final results for our multitask model on the competition test set are shown in Table 3. We can see that the multitask model outperforms the stand-alone models in both tasks.

Model	Road Map Threat Score	Vehicle Detection Threat Score
Stand-alone (block size 5)	0.8212	0.0005
Multitask (block size 5)	0.8504	0.0065

Table 2. Performance on Validation Set

Model	Road Map Threat Score	Vehicle Detection Threat Score
Multitask (block size 5)	0.7633	0.003

Table 3. Performance on Competition Test Set

Here we present some examples predicted by our model. For the vehicle detection task, the model does well in correctly prediction blocks for some of the cars. Even when incorrect,

it still predicts vehicles in nearby regions, as seen in Figure 7 below. In the unsuccessful error case (Figure 8), however, we see that the model's predicted blocks are far from the correct blocks.

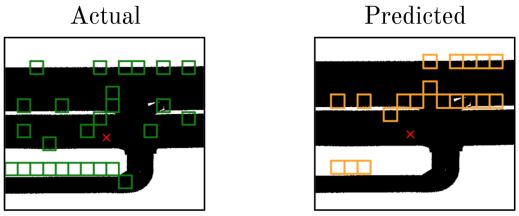

Figure 7. Successful Vehicle Detection Example

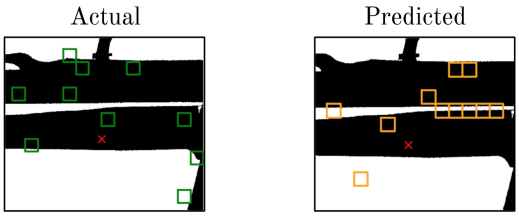

Figure 8. Unsuccessful Vehicle Detection Example

For the road map prediction task, the model does a fairly good job at predicting the large, horizontal roads as shown in Figure 9 but fails at predicting the finer intricacies as shown in Figure 10.

Figure 9. Successful Road Map Detection Example

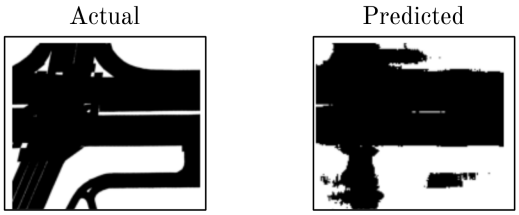

Figure 10. Unsuccessful Road Map Detection Example

4. Conclusion and Future Work

In conclusion, our simplified, multi-task approach produced strong results on the road map task and decent results for the vehicle detection task. For future steps, we plan to incorporate offset and dimension predictions into the bounding box task, similar to how YOLO predicts the offset of each predicted bounding box from the grid cell, as well as the height and width. We believe that this will improve the intersection of predicted and ground truth bounding boxes. We also plan to incorporate an angle prediction, in order to account for rotated cars (cars that appear at a diagonal angle to the ego car's cameras).

References

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Noroozi, M. and Favaro, P. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pp. 69–84. Springer, 2016.

Palazzi, A., Borghi, G., Abati, D., Calderara, S., and Cucchiara, R. Learning to map vehicles into bird's eye view. *CoRR*, abs/1706.08442, 2017. URL http://arxiv.org/abs/1706.08442.

Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL http://arxiv.org/abs/1506.02640.

Ren, S., He, K., Girshick, R. B., and Sun, J. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.