Generating top-down view from multi-image scene captured by moving vehicles

Eelis Virtanen [*] **Mrinal Jain** [*]

Abstract

This paper aims to look at the task related to training a model using images captured by a vehicle in order to generate a top-down view of the surrounding area. Two separate tasks are evaluated: (1) the ability to draw the road layout (2) the ability to detect surrounding objects (primarily other vehicles). The first task was performed using a U-Net like architecture, that consumed the features generated by a denoising auto-encoder, whereas the second task used Single Shot Detection (SSD). The initial testing of the system showed that the results for the road map generation were promising whilst the object detection and bounding box mapping appeared to need more work in order to achieve useful results. Link to code repository

1. Introduction

One crucial component of an autonomous self-driving system is the ability to incorporate the information from all of the cameras capturing the images around the car into a single top-down view of the surroundings, which in turn can feed other systems which relate to making decisions based upon a comprehensive understanding of the environment from all of the directions. There are several distinct elements in forming this top-down view of the surroundings. The first is the ability to predict how the actual road looks like. This is important so that the car knows where in fact it can drive, how many lanes there are and so forth. The second task is to understand what surrounding objects exist that may interfere with the driving process, such as other vehicles or pedestrians. This paper aims to provide solutions for both of the tasks by utilizing deep learning techniques found within the literature.

2. Literature Review

Given the shear amount of unlabelled data, self-supervised learning can give a significant boost to the performance of the models on the downstream tasks. The authors of (1) proposed a unique method for pre-training models to learn a semantically correct feature mappings. The proposed technique converts a given image into a jigsaw puzzle, and the network is supposed to find the correct arrangement (out of all possible arrangements), and in essence "solve" the puzzle. In our initial experiments, this method for pre-training proved to be computationally expensive, and as a result, we decided to explore simpler approaches to use for pre-training.

Transforming an image into the corresponding top-down view is an active area of research, and there have been multiple attempts to achieve this via an end-to-end pipeline. The authors in (2) take a similar approach where a single model called SDPN (Semantic-aware Dense Projection Network) is comprised of two paths. One consumes a patch of an object in an image, and the other path takes the bounding box coordinates of that specific object. The image patch is passed through a ResNet, and the coordinates through a MLP to produce a concatenated feature representation, which is decoded back to generate the top-down view. The methodology in a way shaped our thinking about applying similar techniques, but for the scenario where we have multiple images from every vehicle.

2.1. Object Detection

One of the early methods for object detection was the Deformable Part Model (7) which was based upon using sliding windows without the use of neural networks. It was the R-CNN (8) which initiated the trend of using convolutional networks and became one of the most prevalent methods using neural architectures for object detection. However, one issue with it was that it was quite expensive leading to long predictions times. As a result, approaches such as Fast R-CNN (9) were developed which allowed the reuse of feature maps created by using several image resolutions.

Another set of techniques dealt with region proposal classification. These included works like Multibox (6), whose region proposals are directly replaced by proposals generated from a separate deep neural network. Later methods involved skipping this proposal step and predicting the bounding boxes and the objects directly without the use of two different neworks. Such approaches include YOLO (10) which uses the final feature maps for detecting object categories and the bounding boxes. The SSD (11) method that this paper implements falls under this category as well. It can be seen as a more more flexible method than its predecessors since it can use default boxes of different aspect ratios on each feature location from multiple feature maps at different scales.

3. Modelling

3.1. Self-supervised pre-training on unlabeled data

In order to leverage the unlabelled data for the downstream tasks, a denoising auto-encoder was trained with the hope of learning latent feature representations and semantics contained in the individual images in each scene.

3.1.1. Training of the Denoising Auto-encoder

- The individual images in the unlabelled data were resized to (224×224) pixels, normalized, and finally corrupted with Gaussian noise ($\mu = 0, \sigma = 0.25$).
- The encoder created a low dimension feature representation of corrupted image, which was then used by the decoder to reconstruct the actual image.
- Training was performed using the Adam optimizer, with a default learning rate of 0.001, and a L_2 penalty of 10^{-5}. Mean Square Error (MSE) was used as the loss function.

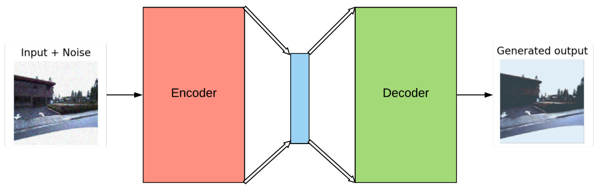


Figure 1. An overview of the denoising auto-encoder. The encoder was a 4-layer convolutional neural network, with LeakyReLU used as the activation function. The decoder correspondingly, was composed of de-convolutional layers, which were used to recreate the (denoised) image from the feature representation "learned" by the encoder. The code for the denoising auto-encoder (model architecture and training) can be found here.

3.2. Road Layout Generation

The encoder part E of the trained denoising auto-encoder was used as the backbone for extracting the features, by essentially freezing its parameters. By following a Siamese net (3) like approach, each of the 6 images in a scene was passed through the trained encoder E, to get a feature representation of dimension `(batch_size, 32, 13, 13)`. The 6 feature representations were then concatenated channel-wise, and the resulting tensor was considered to contain the information relevant to construct the road map of the given scene.

The concatenated tensor of shape `(batch_size, 6` $\times$ `32, 13, 13)` was passed to a model that followed a U-Net (4) like architecture, and it produced the corresponding road layout as the output of shape `(batch_size, 1, 13, 13)`. This can essentially be thought of as a miniaturized version of the actual road map, and its spatial dimensions are then up-sampled to 800×800 pixels using bi-linear interpolation. This generated map however is continuous in terms of the pixel values. In order to convert this to binary, we used the median pixel of each individual road map as the threshold for categorizing a pixel as "road" or "not road".

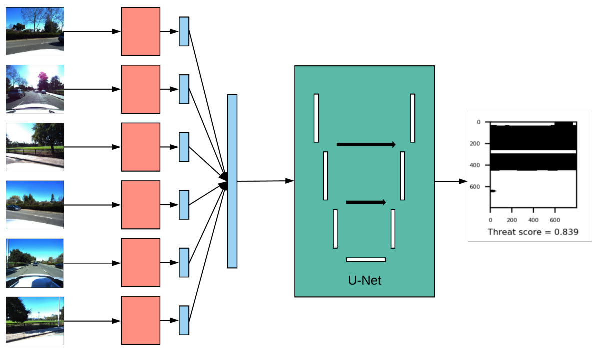
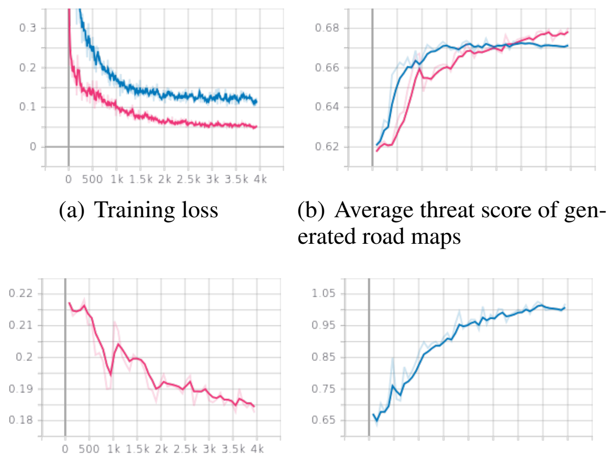


Figure 2. Model architecture for Road Map Generation. The U-Net model is composed of multiple convolutional blocks, where each block is a sequence of convolutional layer, with ReLU activation and batch normalization. There are equal number of down-sampling and up-sampling blocks, where de-convolution layers are used for up-sampling. The down-sampling module is also interspersed with dropout layers. Each down-sampling module effectively doubles the number of channels, and reduces the spatial dimensions by half. The code for the entire road map network (model architecture and training) can be found here.

We interpreted this task as a binary image segmentation problem, where if a pixel value is 1 (in the road map), then it belongs to the "object" road, and otherwise it does not. U-Nets have shown tremendous potential in image segmentation tasks, especially in the medical domain due to their high precision in localization. Before experimenting with U-Nets, we tried some simple stacked de-convolutional and interpolation layers to generate the binary road map, but were unable to successfully train the model.

Choosing an appropriate loss function was a crucial step. Using cross-entropy (weighted and unweighted) gave us counter-intuitive results, where training the model increased both, the validation loss and the threat score (Figure 3). After experimenting with several loss functions, we decided to use dice loss (equivalent to `1 - dice score`) as our loss function. The objective of the dice loss is to essentially maximize the overlap between the actual and the target image, which aligns closely with our final evaluation metric (threat score). Moreover, using dice loss improved the generated road maps qualitatively by having sharper boundaries for the roads. Some results of the experiments conducted are shown in Table 1.



(a) Training loss

(b) Average threat score of generated road maps

(c) Validation loss (Dice Loss)

(d) Validation loss (Weighted (Binary) Cross-entropy)

Figure 3. Training and evaluation metrics on validation data. Out of the 28 scenes available in the labelled data, 8 scenes were used for validation. The model was trained for a total of 50 epochs, and the figure shows the trend when we used Dice Loss and Weighted (Binary) Cross-entropy

The final model was trained using the SGD optimizer, with an initial learning rate of 0.1, momentum of 0.9, and an L_2 penalty of 0.0005. The initial learning rate was chosen based on methodology described in the paper "Cyclic Learning Rates for Training Neural Networks" (5). Moreover, the learning rate was reduced by half, if the validation loss did not improve for 10 consecutive epochs. The results were significantly improved after using a U-Net like architecture, combined with dice loss as the objective function, and per-image thresholding using the median pixel value to convert the resulting road maps into binary.

3.3. Object Detection

For the object detection, a technique called Single Shot Detection (SSD) was used. This technique combines both object localization and detection into one network so it can be seen as a efficient approach as opposed to having two networks for each task. Specifically, there is a regression tasks which aims to predict the bounding boxes followed by a classification tasks which attempts to find the best object prediction for a particular box. In addition to using the

Table 1. Threat scores achieved on the validation set when using different loss functions to train the network for generating road maps

U-Net	Dropout	Loss Function	Threat Score
3	0.25	BCE	0.6674
3	0.25	MSE	0.6628
3	0.25	MAE	0.6509
3	0.25	Weighted BCE	0.6713
3	0.25	Dice Loss	0.6734
3	0.40	Dice Loss	**0.6783**

actual labels provided in the dataset, a negative background class was used as well which the network was able to predict in addition to the true positive classes. However, note that only object localization was needed in this particular project.

Another important component of the approach was the use of multi-scale feature maps which instead of producing predictions on the final feature maps, various intermediate convolution layers were used as well in the prediction. This had the benefit of allowing the network to detect object of various sizes and shapes based upon which intermediate convolution layer was being looked at.

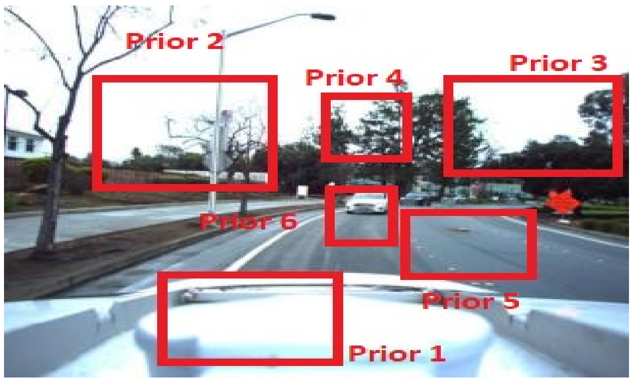


Figure 4. Example of priors on sample image for object detection (done on feature maps in reality). Each prior is matched to a ground truth based upon the Jaccard overlap. Only priors with a overlap of greater than 0.5 are considered to contain the object.

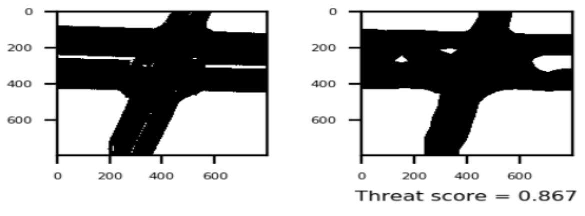
Similarly, the use of around 8000 priors was important (Figure 4). These are effectively pre-computed boxes at specific feature maps which effectively discretize the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. The priors allow us to incorporate our prior beliefs of the task at hand in order to deal with otherwise very large output space.

Finally, after training the model, inference was performed by using the trained network on unseen images in order detect possible bounding boxes. The prediction was the outcome of an optimization procedure which took into account the score produced by each prior for a bounding pox, the maximum overlap we allowed two predictions to have as well as a hard cap on the maximum number of predictions we allowed the system to make. For instance, in our dataset, we rarely had much overlap between images as well as a number of boxes exceeding 20. The minimum score could then be evaluated as a hyper-parameter by using a validation set.

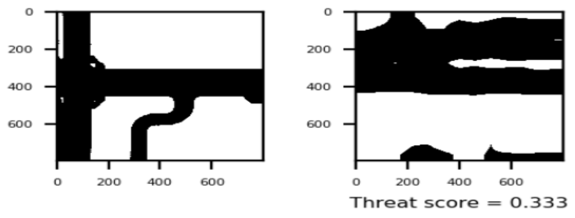
4. Results

4.1. Road Layout Generation

The model seems to perform a relatively good job of generating road layouts that have the 2 horizontal roads. Since most parts of any scene is comprised of these horizontal road patches, the model was able to achieve a high threat score on such layouts, as shown in Figure 5(a). However, the performance was otherwise poor for layouts which involved other types of cross-sections (like turnings, or vertical roads, etc) observable in Figure 5(b).



(a)



(b)

Figure 5. Sample road layouts generated by the final model on the validation data (Left: actual road map; Right: predicted road map)

4.2. Object Detection

So far the results for the object detection have been poor. The results on the chosen validation set (126-135) have been low with a score of around 0.0004. Note, the entire architecture was trained only once before the submission so there is a lot of uncertainty on its feasibility.

5. Potential Improvements

First, it is possible that the priors used have not been adequate for the task at hand. The priors used in this project where tailored for the Pascal VOC dataset which are unlikely to be very good for this task. However, due to time constraints, task specific priors were not created.

Second, the calculation for Jaccard overlap between two boxes was computationally difficult for cases were the objects were not aligned with the car. Normally, if two objects are aligned, we can calculate a simple rectangle in order to find the common overlap, However, if they are not aligned no such easy method exists. Python packages such as Shapely were not efficient enough in order to perform the calculation at a large scale and a simple approximation was used.

6. Conclusion

This paper used two different techniques to create a top-down view based upon multiple images captured from around a vehicle. Both techniques incorporated self-supervied learning (11) to learn effective representations of the images in order to combat the lack of labelled data. The U-Net achieved a decent score in generating the road layouts, but the SSD approach for object localization achieved poor results due to several proposed issues.

References

[1] Mehdi Noroozi, Paolo Favaro Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In CVPR. (2017)

[2] Andrea Palazzi, Guido Borghi, Davide Abati, Simone Calderara, Rita Cucchiara Learning to Map Vehicles into Bird's Eye View. In CVPR. (2017)

[3] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov Siamese neural networks for one-shot image recognition. In: ICML. (2015)

[4] Olaf Ronneberger, Philipp Fischer, Thomas Brox U-Net: Convolutional Networks for Biomedical Image Segmentation.

[5] Leslie N. Smith Cyclical Learning Rates for Training Neural Networks. In CVPR. (2015)

[6] Erhan, D., Szegedy, C., Toshev, A., Anguelov, D Scalable object detection using deep neural networks. In: CVPR. (2014)

[7] Felzenszwalb, P., McAllester, D., Ramanan, D. A discriminatively trained, multiscale, deformable part model. In: CVPR. (2008)

[8] Girshick, R., Donahue, J., Darrell, T., Malik, J Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR. (2014)

[9] Girshick, R Fast R-CNN. In: ICCV. (2015)

[10] Redmon, J., Divvala, S., Girshick, R., Farhadi, A You only look once: Unified, real-time object detection. In: CVPR. (2016)

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed "SSD: Single shot multibox detector," arXiv:1512.02325, 2015