# Towards Autonomous Vehicles with CNN: U-Net Convolutional Neural Network for Obstacle Detection in Self-Driving Cars Tasks

Albert A. Cervera-Uribe*

Facultad de Matemáticas, Universidad Autónoma de Yucatán

## Abstract

Development of Self-driving cars aims to drive safely from one point to another in a coordinated system where the on-board autonomous vehicle system should react and alert drivers about the driving environments and possible collisions that may arise between drivers and obstacles. In order to achieve a high level of autonomy in urban scenarios with unpredictable traffic, these systems must have robust and reliable obstacles detection systems. In particular, this paper describes an application of a U-Net Convolutional Neural Network for the task of identification and localization of cars, trucks and pedestrians from a stream of images from a video of driving in heavy traffic. The architecture of the network was constrained and specifically designed for this task, but it showcases that such a network could be trained end-to-end to real image recognition and classification without a large, complex preprocessing stage.

**Keywords:** deep learning, self-driving cars, autonomy, context aware, neural network, CNN, convolution, image processing.

## 1 Introduction

The ability to perceive the vehicle's environment at any moment is one of the main challenges in the field of autonomous vehicles. Environmental and meteorological conditions such as lighting, fog and rain are permanently changing, that in addition with static and dynamic objects in a scene, it shows that more robust sensing systems must be enabled. In the recent years, there has been an increasingly attention to vision-based vehicle detection with convolutional neural networks (convnets) for semi-autonomous vehicles tasks. The typical use of convolutional networks is mainly on image classification tasks, where the output of an image would be the single class label where it is predicted to belong. However, in the particular task of object detection for autonomous vehicles, the desired output should not only include the class label prediction, but its localization, i.e., the class label is meant to be assigned to each corresponding pixel on the image.

This article presents an overview of the implementation with of a U-Net convolutional neural network proposed in the strategy of Ronneberg et al [3] for the task of car, truck and pedestrian classification and localization from a dataset that consist on a stream of images from video of driving in heavy traffic. Implementation was done with Tensorflow [1] and Keras [2]. U-net is an encoder-decoder type of network for pixel-wise predictions that is unique because the receptive fields after convolution are concatenated with the receptive fields in the up-convolving process. This allows the convolutional neural network to use original features in addition to the features after up-convolution steps, allowing U-net to have a better performance than a network that only has access to features after up-convolution [3]. The dataset of selection was the one provided by Udacity which is composed of 9,420 frames from a video driving in Mountain View, California collected from a Point Grey research cameras running at full resolution of 1920x1200 at 2 hz. The dataset contained a label file with bounding boxes marking other cars, trucks and pedestrians during daylight conditions. Across all the dataset, there are labeled 5,675 pedestrians, 3,819 trucks and 62,570 cars which gives a total of 72,064 labeled objects in the dataset. In this work, the classes car, truck and pedestrian were combined into a single class called obstacles mainly because in urban scenarios were pedestrian and vehicles are way closer than in rural environments, it is critical for a system to detect them accurately.

*e-mail: aaron.cervera.u@gmail.com

## 2 Network Architecture

The main idea with the proposed convnet is to supplement a usual contracting network by successive layers, where pooling operators are replaced by up-sampling operators with a large number of feature channels which in turn allow these layers to propagate context information and to increase the resolution of the output. In order to localize objects of interest, high resolution features from the contracting path are combined with the up-sampled output. A successive convolution layer could then learn to create a more precise hypotheses based on this resulting information. The network architecture is illustrated in Figure 1. This architecture consists of a contracting path (in the left side) and an expansive path (right side). The contracting path follows the typical architecture of a convnet. It consists of the repeated application of two convolutions with a 3x3 kernel with zero padding (used for maintain input size) and each one followed by a rectified linear unit (ReLU). Then a 2x2 max-pooling operation with stride 2 is performed for down-sampling. At each down-sampling operation, the number of feature channels is doubled to get more context information into higher resolutions. In the other part, each step in the expansive path consists of an up-sampling of the feature map followed by convolution with kernel 2x2 ("up-convolution") that halves the number of feature channels, then a concatenation operation with the correspondingly feature map from the contracting path. After that, two 3x3 convolutions followed by a ReLU are performed. At the final layer a convolution with kenel 1x1 is used to map each feature vector to the number of classes. In total, the network has 23 convolutional layers.

## 3 Data and target preparation

Cross-validation is a technique to evaluate models by partitioning the original sample into training and testing sets to evaluate the performance of the model. Therefore from the 9,420 frames of the dataset, 7,420 images were used into the training set and the last 2,000 images for the testing set. Within the dataset it was provided information that defines the bounding boxes of cars, trucks and pedestrians as well. The region within those bounding boxes was used to define the region of interest (ROI) mask which was employed as the ground truth segmented mask. This process is illustrated in Figure 2 in which the left image is a randomly sampled frame from dataset and the right image is the computed ground truth ROI mask. The ultimate goal of the convolutional neural network is to predict a ROI mask given the original frames.

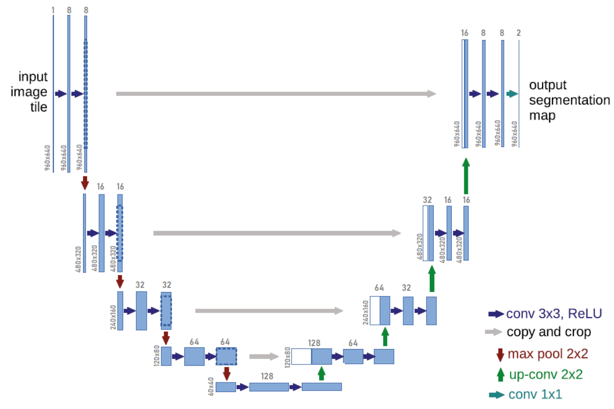

Figure 1: U-net architecture (with 60x40 pixels in the lowest resolution of an input image). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The resolution of the input is provided at the lower left edge of the box. Notice that convolution layers are done with zero padding for maintenance of input size. Gray arrows represent copied feature maps.

## 3.1 Intersection over Union evaluation metric

Intersection over Union (IoU) is an evaluation metric used to measure the accuracy of an object detector on a particular dataset. In order to apply IoU to evaluate the performance of the predictor model, it is necessary to have: (1) The ground truth bounding boxes, and (2) The predicted bounding boxes from the network. Computation of the Intersection over Union could be determined as the area of overlap divide by the area of union as Figure 3 shows. In general, an IoU score greater than 0.5 is considered as a ”good” prediction. As the IoU performance score varies between 0 and 1, the objective of the model is to maximize this score to obtain a good accuracy. As a custom objective function in Keras was made for computation of this metric, it was set that the goal of that function is to reduce the cost error established as the negative IoU score value.

## 4 Training

The input images and their corresponding segmented masks were used to train the neural network with an extension to the stochastic gradient descent called Adam optimizer to update network weights iterative based on the training data. Its learning rate was set to 0.0001. In deep neural networks with many convolutional layers and different paths through the network, a good initialization of the weights is extremely important. For a network with convolutional and ReLU layers it could be achieved a Gaussian distribution of the initial weights to adapat each feature map for having unit variance.

Implementation was done with Tensorflow in a Linux environment with an Intel Core i7 octa-core CPU running at 3.40 GHz and with a Nvidia GeForce GTX 745 GPU. To minimize the overhead and make maximum use of the GPU memory, it was favor a large input over a large batch size and hence reduce the batch to a single image. This single image was randomly sampled from the training set. The number of samples per gradient update was 1,000 while the number of epochs was set to 20. The total time of execution was 2 hours 46 minutes with a mean time of executing an epoch of 8 minutes 19 seconds with the hardware configuration previously mentioned. Table 1 summarizes the results obtained in each epoch. The accuracy was about 76.56%. Figure 4 illustrates the accuracy growing rate followed by the 20 epochs.

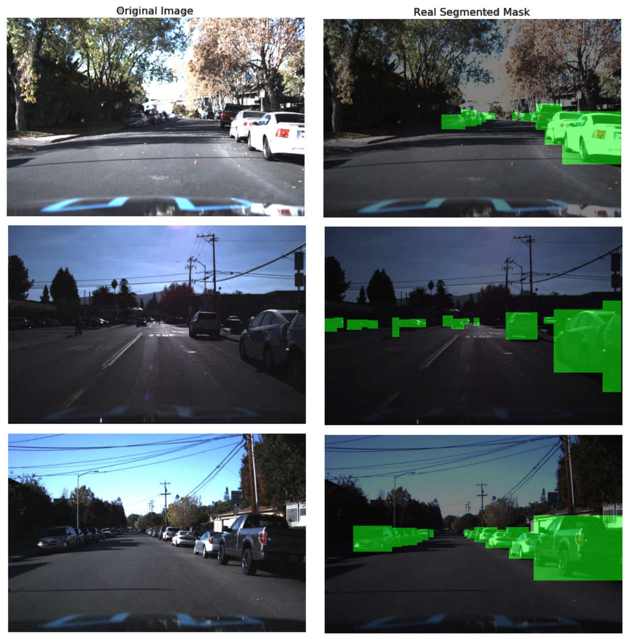

Figure 2: Ground truth ROI masks: The images on the left panel are the original randomly sampled frames from dataset, while the images on the right panel presents the object ROI masks from the bounding boxes provided within the dataset. The goal of the convnet is to predict those ROI masks.

| Epoch | Time | Time per step | Loss | Accuracy |
|---|---|---|---|---|
| 01 | 9m46 | 586ms | -0.4020 | 0.4020 |
| 02 | 8m14 | 514ms | -0.5009 | 0.5009 |
| 03 | 8m14 | 514ms | -0.5540 | 0.5540 |
| 04 | 8m27 | 507ms | -0.5879 | 0.5879 |
| 05 | 8m21 | 501ms | -0.6312 | 0.6312 |
| 06 | 8m16 | 497ms | -0.6554 | 0.6554 |
| 07 | 8m13 | 494ms | -0.6722 | 0.6722 |
| 08 | 8m13 | 494ms | -0.6751 | 0.6751 |
| 09 | 8m10 | 490ms | -0.6916 | 0.6916 |
| 10 | 8m10 | 491ms | -0.6884 | 0.6884 |
| 11 | 8m12 | 492ms | -0.7236 | 0.7236 |
| 12 | 8m12 | 492ms | -0.7212 | 0.7212 |
| 13 | 8m12 | 492ms | -0.7342 | 0.7342 |
| 14 | 8m10 | 491ms | -0.7453 | 0.7453 |
| 15 | 8m10 | 491ms | -0.7486 | 0.7486 |
| 16 | 8m09 | 489ms | -0.7534 | 0.7534 |
| 17 | 8m10 | 491ms | -0.7577 | 0.7577 |
| 18 | 8m10 | 490ms | -0.7622 | 0.7622 |
| 19 | 8m12 | 492ms | -0.7601 | 0.7601 |
| 20 | 8m07 | 488ms | -0.7656 | 0.7656 |

Table 1. Summary of the results obtained by epoch. Time is given in minutes while time per step is given in milliseconds. Accuracy of this training configuration with the training set was about 76.56% with the IoU metric.

## 4.1 Experiments

With a number of samples per gradient update and number of epochs set as section 4 states, it was decided to test a new configuration for training to see how the U-Net convolutional neural network performs. The new configuration was to set the number of samples per gradient update to 6,000 and 3 epochs to test. Before running this training configuration, the parameters of the model were reestablished to

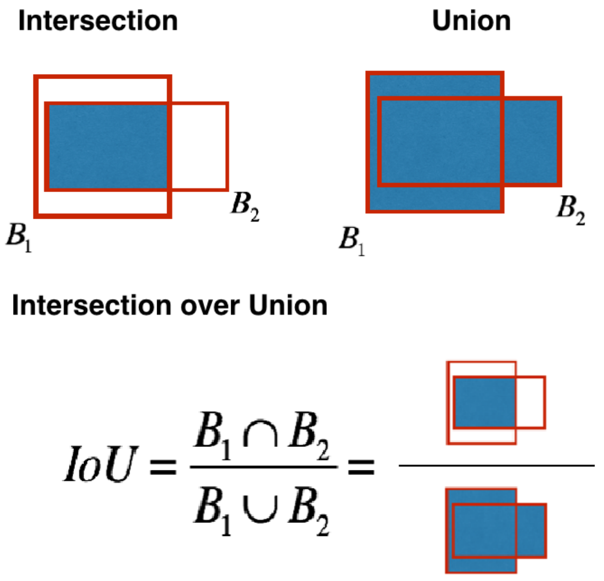

Figure 3: IoU: The numerator is the computation of the area of overlap between the predicted bounding box and the ground truth bounding box. In the other hand, the denominator is the area of union encompassed by both the predicted bounding box and the ground truth bounding box. Result of this division is the Intersection over Union.

count totally as a new training time with the training dataset. Table 2 summarizes the results of this training configuration while Figure 5 illustrates the accuracy growing rate followed by the 3 epochs. The total time of execution was 2 hours 28 minutes with a mean time of executing an epoch of 49 minutes 30 seconds. Despite the accuracy obtained was about 75.69% with the training set (0.87% below than the training configuration in section 4), an empric evaluation of the results obtained, indicates that the the model increases its accuracy faster with the new configuration in comparison with the one used in section 4. However more analysis is needed to prove this claim.

| Epoch | Time | Time per step | Loss | Accuracy |
|---|---|---|---|---|
| 01 | 50m46 | 508ms | -0.5560 | 0.5560 |
| 02 | 48m57 | 490ms | -0.6975 | 0.6975 |
| 03 | 48m47 | 488ms | -0.7569 | 0.7569 |

Table 2. Summary of the results obtained by epoch; 6,000 samples per gradient update were set with 3 epochs. Time is given in minutes while time per step is given in milliseconds. Accuracy of this training configuration with the training set was about 75.69% with the IoU metric.

## 5 Results

Once the model was trained, the performance of the convolutional network was evaluated by using the testing set previously separated from the original dataset. Despite the experimental results discussed in section 4.1, for testing the performance of the model, the weights obtained with the training configuration of section 4 were used, mainly because it was the one with the highest accuracy on the training set. This process is illustrated in Figure 6 in which the left image is a randomly sampled frame from testing set, the center image the predicted ROI mask which denotes obstacles and the right image is the computed ground truth ROI mask. The accuracy obtained with the testing data was about 85.54% with the IoU metric.

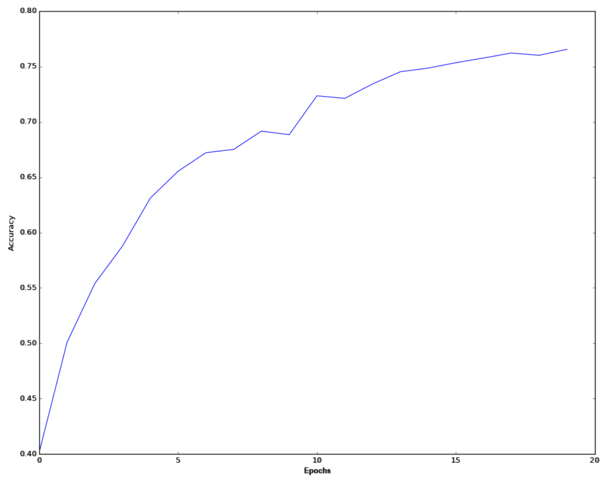

Figure 4: Accuracy graph of the model with the training data when using 1,000 samples per gradient update and 20 epochs.

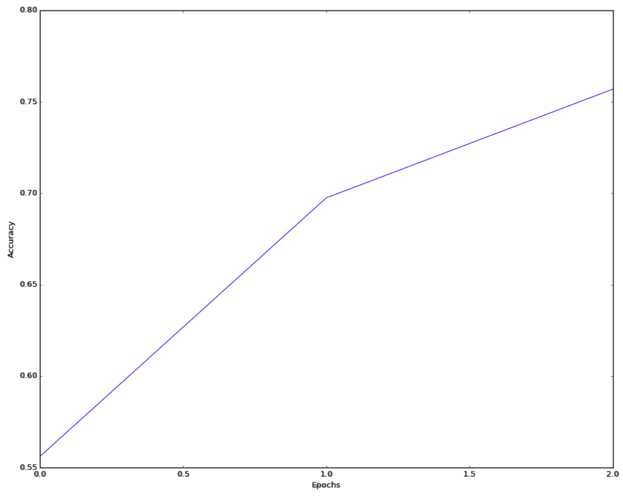

Figure 5: Accuracy graph of the model with the training data when using 6,000 samples per gradient update and 3 epochs.

## 6 Conclusion

The U-Net architecture developed for the specific task of vehicles and pedestrians detection and localization, achieves a very good performance on predicting the ROI mask where it is likely to have an obstacle with an accuracy of 85.54% with the testing set. Despite the computational time for training the net was constrained, a very reasonable training time of 2 hours 46 minutes gives an accuracy of 76.56% on a Nvidia GeForce GTX 745 GPU. The full Tensorflow-based implementation and the trained weights of the network are provided within this submission.

## Acknowledgements

I would like to thank the computing center of Facultad de Matemáticas, UADY for give me the necessary GPU to train my model, otherwise my home laptop would have trained for days.

## References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale

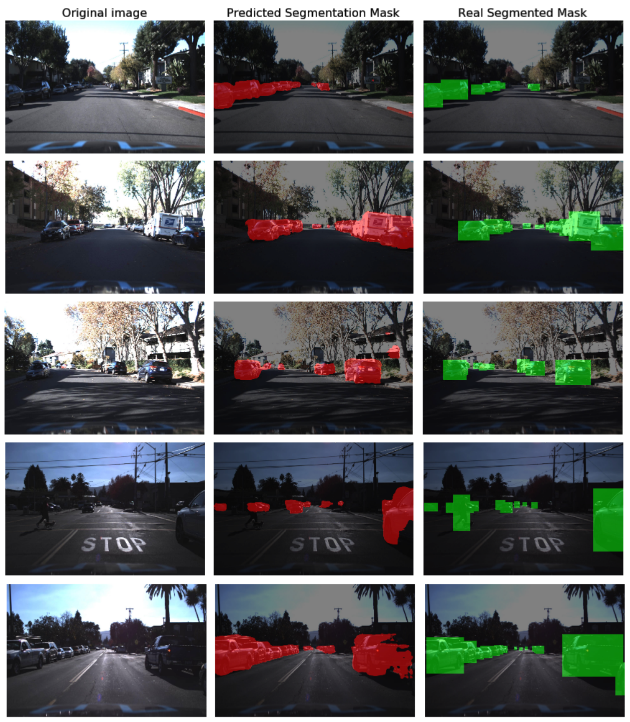

Figure 6: Prediction results of the U-Net model: the left panel is a randomly sampled frame from the testing set, the center panel denotes the predicted ROI mask which stands for obstacles and the right panel is the computed ground truth ROI mask.

machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] F. Chollet et al. Keras: Deep learning library for theano and tensorflow.(2015). *There is no corresponding record for this reference*, 2015.

[3] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.