



CSRBE 549 Computer Vision

Project Report Team II

Semantic Segmentation for a Self Driving Vehicle

Signature		
Member		Contribution
Keith Chester		33.3%
Bob DeMont		33.3%
Zeke Flaton	<u>(Zeke is Travelling and unable to sign)</u>	33.3%

Grading:	Approach	_____/15
	Justification	_____/5
	Analysis	_____/15
	Testing and Examples	_____/15
	Documentation	_____/10
	Difficulty	_____/10
	Professionalism	_____/10
	Presentation	_____/20
	Total	_____/100

Table of Contents

Content	Section	Page No
Introduction	I	1
Problem Statement	II	1
Related Work	II.A.	1
Significance	II.B.	1
Methodology	III	1
Approach	III.A.	1
Data Gathering	III.B.	1
Data Augmentation	III.C.	2
Challenges	III.D.	2
Code	III.E.	3
Metrics	IV	3
Results	V	3
Accuracy	V.A.	3
Simulation	V.B.	4
Discussion	VI	4
Conclusion	VII	5
Future Work	VIII	5
References		5

Semantic Segmentation for a Self Driving Vehicle

WPI Robotics Engineering - RBE549 Term Project

Zeke Flaton

WPI Robotics Engineering Dept
Worcester Polytechnic Institute

Keith Chester

WPI Robotics Engineering Dept
Worcester Polytechnic Institute

Robert DeMont

WPI Robotics Engineering Dept
Worcester Polytechnic Institute

Abstract—Semantic segmentation for a self driving vehicle achieved in live simulation using CARLA. A deep neural network is utilized to categorize pixels in an image generated by a virtual front-facing camera on our vehicle.

I. INTRODUCTION

For our term project, we built a system that can perform semantic segmentation on images with a focus on self driving vehicles. Our goal was to distinguish different parts of an image so that the self driving vehicle can distinguish the road, sidewalk, vehicles, pedestrians, road signs and lights. Our approach incorporated training a U-Net based deep neural network for fast, efficient, correct image segmentation.

II. PROBLEM STATEMENT

A. Previous Work

As noted in [1] many previous efforts have accumulated human labeled datasets of various sizes to enable deep learning for image segmentation. This is time-consuming, costly, and tedious. For expediency, most research is now incorporating use of video game scenarios and simulations to quickly advance their work [2] and [3].

B. Significance

Today most automakers are endeavoring to incorporate some level of autonomy in their offerings while the general public is alternately enthusiastic and hesitant to accept and adopt these systems. In the US alone, there were over 12MM traffic crashes in 2019 involving nearly 36,000 fatalities. The NHTSA estimates that 94% of these were caused by human errors. The NHTSA believes autonomy will greatly reduce these numbers as well as have desirable effects on efficiency, convenience, economic, and environmental concerns and has advanced policies to encourage various levels of autonomy for road use.

Given that auto manufacturers are using some version of image segmentation in autonomous commercial products already, we are unlikely to add significant contributions to the general knowledge base. However, for our continued progress in the field of robotics, a practical, working knowledge of image segmentation will be valuable in future projects including our capstone exercises.

III. METHODOLOGY

A. Approach

For this project we aim to utilize CARLA, a simulator developed by Intel to assist the research and development of technologies for autonomous vehicles. The simulator creates a virtual neighborhood with vegetation, traffic signs, other moving vehicles, and pedestrians. In this simulator we will have a target car with a virtual front-facing camera.

The simulator provides an additional "view" for this camera; since it has perfect knowledge of the location and category of all objects within the simulation, it can correctly identify and categorize all pixels witnessed by the front-facing camera. CARLA provides a dozen categories such as vehicles, pedestrians, buildings, fences, traffic signs, vegetation, and so on. From these we focus on a subset of important categories necessary for navigation and task completion in a driving environment:

Roads	Road lines	Sidewalks	Ground
Vehicles	Traffic Signs	Pedestrians	Other

TABLE I
SEMANTIC CATEGORIES

B. Data Gathering

To build our dataset, we utilized CARLA's Python API to build a pipeline for label generation. The script would create a new world with a random city map provided by CARLA. Other environment parameters would then be randomly selected; the time of day, angle of the sun, precipitation, cloud cover and color, wind, and road wetness. Each are randomly chosen to make each run unique and diversify our data. A random number of vehicles and pedestrians are spawned throughout the map, each with random waypoints for their pathing. One such vehicle is paired with both our RGB and semantic labels camera, where our software collects images and labels along its path every 1.5 seconds. We collected approximately 15GB of images and labels, or 14,390 dataset pairs.

We attempted to add additional data sets from online repositories. We identified Lyft Challenge, CityScapes and CAMVID datasets. In at least some of these, we encountered aliasing issues in the ground truth images that would have required

extensive cleaning before being usable. As a result we decided to generate a larger dataset directly out of CARLA instead of mixing in these datasets.

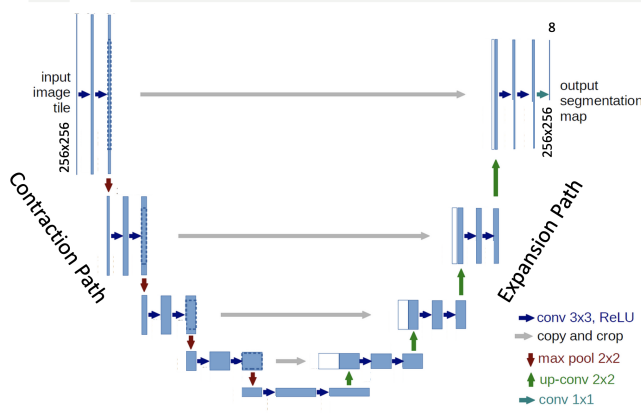


Fig. 1. U-Net Model Architecture

We utilized a standard U-Net model with two significant changes - the first is our input size is adjusted for larger images (either 256x256 or 512x512). Our second change is, unlike the original U-Net, we upsample back to our original input width and height.

To build, train, and execute the deep learning models for semantic segmentation we will be utilizing the Tensorflow framework and platform, created by Google. We choose this framework for its vast popularity and expressiveness for training neural networks. We utilized an NVIDIA RTX 3080 GPU with 10.2 GB of RAM. During training, we found that we utilized 9.2GB of RAM when training batches of 32 images at an input and output resolution of 256x256, and 9.1GB of RAM when training batches of 8 images at an input and output resolution of 512x512.

C. Data Augmentation

When we began training, we split the dataset randomly into a train and validation dataset, a 75%/25% split. Our custom dataset loader would apply data segmentation to the training dataset and utilize the validation dataset as is. Data augmentation virtually increased the size of our dataset to combat overfitting. For data augmentation, we employed several techniques of randomly applying:

- Adjustments of brightness and contrast
- Light to heavy amount of salt and pepper image noise
- Degree of blurring of the entire image
- Chance of horizontally flipping the image
- Chance of utilizing a small focused cutout of the image instead of the whole image

With 14,930 base image pairs, this data augmentation and our dataset size provided us great variety for training our network.

During training, images were resized to either 256x256 or 512x512 (or directly subsampled at that size) before being fed

into the network. The output tensor of the network would be an equivalently sized $(w, h, 8)$ tensor, where the dimension represents the number of categories (7 + a generic unlabeled category). This axis is treated as the calculated probability that a given pixel in that location represents the category. To create human-readable labels, we find the index of the highest probability for each category. We then convert the resulting $w \times h$ tensor into an assigned RGB color value for each category.

We utilized sparse categorical crossentropy (SCCE), and sparse categorical crossentropy focal loss (SCFL) as our loss functions in various experiments and Tensorflow's Root Mean Square Propagation (RMSProp) optimizer.

Once the model was trained, we applied tooling we had written that would utilize the model to apply to additional images or video footage to generate labeled overlays.

D. Challenges

We encountered numerous difficulties in labeling due to our severe imbalance of categories. The unlabeled, road, and sidewalk categories are not only present in every image, but also consist of the vast majority of each image. As a result our network would rapidly learn to identify their pixels, and the resulting calculated loss would rapidly shrink. This means categories that showed on fewer images and represented smaller percentages of the overall image (such as traffic signs and pedestrians) were not being labeled correctly, nor was the loss high enough to mitigate the incorrect labeling during back-propagation. We addressed this by switching from our original loss function, sparse categorical crossentropy, to sparse categorical crossentropy focal loss. Focal loss is an approach to losses that provides weight adjustment to imbalanced categories during training. The loss would calculate our normal sparse categorical cross entropy loss for each pixel, but then adjust it per the count of present rare categories.

Paired with focal loss, in later experiments, we utilized loss weights. This feature allows us to weight the loss against a given category by a weight such that certain categories act as if they have higher or lower learning rates. We calculated the number of pixels for each category as represented across the entire dataset. From here we experimented with multiple approaches, but settled upon a given loss weight L_W :

$$L_W = 2 \times \sigma \left(\frac{\frac{\text{Total \# of pixels}}{\# \text{ of categories}}}{\frac{\# \text{ of pixels for this category}}{\# \text{ of pixels in dataset}}} \right) \quad (1)$$

This produced a result within the range of $(0, 2)$ that affected our learning rate of applied loss across each category.

Images were resized to the model's input/output resolution prior to inference. We observed that smaller objects within the image would be thinned to a few pixels, without much detail for the network to differentiate the object from its surroundings. To combat this, we attempted to double our initial input resolution from 256x256 to 512x512. Doing this forced us to dramatically slow our training speed, as batch sizes were shrunk from 32 images per batch to 8 images per

batch due to GPU memory limitations. Still, this did result in a notable increase in performance for detection of smaller items in images.

Following up with this idea, we theorized that if we could "zoom" the focus of the network, we could probably generate finer detail for small objects in an image. To that end, we generated additional tooling that would run inference on the entirety of the image, but then perform additional inference on small zoomed-in slices of the image as well. The resulting labels for smaller categories would be composited to a singular output.

E. Code

Code for all dataset generation, U-Net model training and associated utilities can be found at

<https://github.com/hlfshell/rbe549-project-segmentation>

IV. METRICS

We planned to measure success on this project with the following goals:

1. Generating a working simulation generating a real-time segmented image interlaced with the real world image from CARLA
2. 90+% accurate segmentation of vehicles and pedestrian objects.

V. RESULTS

A. Accuracy

With the network trained, we started to output some results by taking new images from our test set and running them through the model. The following image shows an example that we used. Baseline training of our U-Net model utilized



Fig. 2. Original Image

2915 images and their corresponding ground truth masks (Baseline Model) with data augmentation for additional variety

to fight overfitting. Images were resized to 256x256 with Sparse Categorical Cross Entropy(SCCE) loss function.

Category	% Pixels Correctly Inferred
Unlabeled	97.35%
Traffic Sign/Lights	47.18%
Roads	96.34%
Road Lines	51.37%
Sidewalk	83.44%
Ground	77.54%
Vehicles	90.65%
Pedestrians	20.29%

TABLE II
BASELINE RESULTS

While the Roads and Vehicles categories met our goal threshold for accuracy, pedestrians fell well short.

Accuracy was tested by comparing inferred pixel values against the ground truth pixel values and determining a confusion matrix over our 2915 images. The same evaluation dataset was used in subsequent iterations of accuracy computations for equal comparisons.

For a next iteration, we increased the dataset size to 14,930 base images with data augmentation that added a great deal of additional variety. Again we utilized 256x256 resized images with SCCE loss function. Results improved, but did not meet our target accuracy threshold for pedestrians and other underrepresented categories.

Resources [5] and [6] suggested that Focal Loss and/or category weighting might improve performance for underrepresented categories. In the next iteration we added both and again saw improvement.

We continued to seek improvement in accuracy in subsequent iterations. In the next iterations, we pursued the same approaches with model input image size increased to 512x512 with a corresponding batch size reduction for memory constraints. Again we started with the SCCE loss function, followed by trainings with Sparse Categorical Focal Loss (SCFL) and SCFL with class weighting were tested with results as seen in Table III and Figure 3.

Our process resulted in steadily improving results, though we never achieved the desired 90% accuracy on underrepresented categories. However, we did note that category loss weights seemed to have a bigger influence on inference of underrepresented categories vs SCFL. We see in both 256x256 and 512x512 runs, that tests with loss weights as implemented outperformed those without and even those with a focal loss function. A full factorial DOE could confirm this effect, though it appears obvious.

Percent Pixels Correctly Inferred

Category	Base Line	256 SCCE	256 SCFL Wts	512 SCCE	512FL SCFL	512 SCFL Wts
Unlabeled	97.35	98.77	97.73	98.85	98.37	98.8
Signs/Lights	47.18	47.62	56.6	59.49	66.49	67.6
Roads	96.34	96.66	97.17	95.34	97.46	97.16
Road Lines	51.37	65.23	67.47	78.19	75.1	77.07
Sidewalk	83.44	91.19	93.81	93.52	88.23	93.97
Ground	77.54	72.05	86.58	81.21	77.69	77.27
Vehicles	90.65	94.53	95.08	96.98	97.29	97.43
Pedestrians	20.29	32.70	43.85	52.06	39.75	59.93

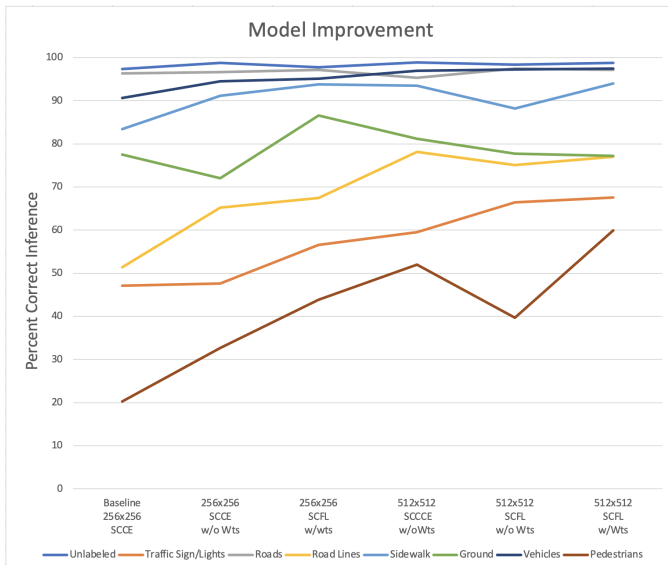
TABLE III
TRAINED MODEL ACCURACY RESULTS

Fig. 3. Accuracy Improvement

B. Simulation

We used the model to infer a set of labels. Since the labels use numbers 0-7 instead of RGB, this results in an image where it is impossible to discern regions based on label. Therefore, a script was used to turn these numeric labels into a human viewable image as seen below.

To achieve higher accuracy on under-represented labels, we then built a script that would sub-sample the main image with higher resolution. We designed it to take 4 sub-images across the horizontal center line and run those through the model as well. Those labels were then overwritten on the original label image only if they matched an under-represented class like pedestrians, traffic lights, or cars.

Once we had a higher fidelity label image, we overlaid it on the original image with 40% transparency to help get a sense of how effective our model was at classifying regions. Our final step was to take dash camera footage, or other footage of a vehicle driving, and apply this overlay process to each



Fig. 4. Image Generated from Inferred Labels



Fig. 5. Image Overlaid with Labels Image

frame. The resulting video shows a set of images with overlaid labels in movie form.

VI. DISCUSSION

After many iterations of development and experimentation, we ended up developing a fairly accurate model. One of the issues we ran into early on was that under-represented classes, like pedestrians and traffic lights, were not being recognized by our early networks because we did not have sufficient data to train the model compared to roads, cars, and sidewalks. After some research, we found a variety of ways to handle this including cutting out data that did not have our under-represented categories. However, given that we felt we did not have sufficient data to lose more, we decided to forgo this route and try another. We instead wrote a custom loss function for our training, which heavily penalized failures to identify

these under-represented categories. By doing this, the model learned that recognizing pedestrians was very important and thus we could train the network with a hugely skewed data set.

VII. CONCLUSIONS

With a U-Net model and over 14,000 images, we were able to train a network to do semantic segmentation on 8 different classes: cars, traffic lights/signs, roads, road lines, sidewalks, ground, vehicles, and pedestrians. By creating a custom loss function we were able to train effectively on the under represented classes, which were road lines, traffic lights/signs, and pedestrians. By using a script that would analyze an image with a base image and 4 higher resolution subsections and overlaying only the under-represented classes, we were able to create an effective output that could recognize these objects from a greater distance despite taking up very few pixels in an image. Finally we used the models and tools we built to create visual overlays where we overlaid the inferred labels on top of actual images and videos.

VIII. FUTURE WORK

Additional work could focus in the areas of model refinement and real life application.

A. Model Refinement

While we went thru a number of iterations of refinement, more are possible.

- We could always add additional images to the base data set.
- We could increase the number of epochs of training: the run time image augmentation permits additional training with reduced overfitting.
- We could add images to the dataset with greater representation of underrepresented categories
- We could expand categories to the full CARLA category library

B. Real Life Applications

We applied the model to a single real life image from a dashcam which had mixed segmentation results. Vehicles were well identified, but some streets and controls were misidentified. Perhaps this is influenced by the fish-eye nature of the dash cam as compared to our training environment.

More work to accommodate distortion and real life frame rate is necessary.



Fig. 6. DashCam Image

- [4] C. Wang, '2D object detection and semantic segmentation in the Carla simulator', Dissertation, 2020.
- [5] 'Multi-class Classification with Focal Loss for Imbalanced Data Sets', <https://www.dlology.com/blog/multi-class-classification-with-focal-loss-for-imbalanced-datasets/>
- [6] 'A Loss Function Suitable for Class Imbalanced Data: "Focal Loss"', <https://towardsdatascience.com/a-loss-function-suitable-for-class-imbalanced-data-focal-loss-af1702d75d75>

REFERENCES

- [1] Liu, X., Deng, Z. & Yang, Y. Recent progress in semantic image segmentation. *Artif Intell Rev* 52, 1089–1106 (2019). <https://doi.org/10.1007/s10462-018-9641-3>
- [2] S. Malec, 'Semantic Segmentation with Carla Simulator', Dissertation, 2021.
- [3] D. R. Niranjana, B. C. VinayKarthik and Mohana, "Deep Learning based Object Detection Model for Autonomous Driving Research using CARLA Simulator," 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), 2021, pp. 1251-1258, doi: 10.1109/ICOSEC51865.2021.9591747.