CS230

CNN-based robust sidewalk identification for autonomous drone applications

Milan Bidare
mbidare@stanford.edu

Arvind Srivastav
arvindsr@stanford.edu

Tri Khuu
tkkhuu@stanford.edu

1 Introduction

As a society, our needs have evolved with advancements in science and technology. Today, we stand at the juncture where we feel the necessity of instant services, such as shipment delivery and emergency support. Among our current systems, autonomous drone is only viable option that can meet these requirements cost-effectively. And some companies, like Amazon.com, are already testing these services for shipment delivery within an hour. Unfortunately, our current city infrastructures are not flexible enough to easily accommodate drone navigation: flying too high in air, they'd pose security threat to city, and flying over roads, they pose a risk to the traffic. In this scenario, a practical solution would be to fly drones over sidewalks. The advantage of using sidewalks is two-fold: they facilitate planned routes to almost all locations in cities, but without posing any risk to the traffic flow.

In this project, we've proposed a sidewalk detection network that identifies sidewalks, as well as objects on it (traffic signs, vegetation, barriers) and its vicinity (people, roads, vehicles, buildings) in real-time from from camera video input. This would aid planning the route for autonomous drones over sidewalks. Since sidewalks appear in different shapes, colors, and orientations, their identification is a quite complex problem. To tackle this problem, we've taken a nuanced data-driven deep learning approach that results in reliable identification of sidewalks from camera images.

2 Related work

In the past, significant research has been done in identifying lanes using computer vision methods [3]. The more recent works focus on lane detection using deep learning methods, such as spacial CNN (SCNN) [8], fully convolutional network (FCN) [4], or point instance network (PINet) [6], with largely positive results. While presence of sidewalks is correlated with the side lanes, computer vision based methods cannot aid this problem due to immense variations in locations and forms of sidewalks.

Most of the work in identifying sidewalks has been in the context of image segmentation [7], where broad classes are identified from images using CNN. An example of this is *SegNet* [10], based on [1, 5]. SegNet is primarily motivated by scene understanding applications and is designed to be efficient both in terms of memory and computational time during inference. It is also significantly smaller in the number of trainable parameters than other competing architectures. However, it focuses on segmentation tasks, so it computes loss weighted to accuracy of detecting classes present in image. Essentially, the classifier gives weights to different classes based on the number of pixels belonging to that class in the image. This means that if a sidewalk lies at the peripheral edge of an image, then less weight is assigned to correctly identifying the sidewalk, which reduces reliability of sidewalk detection.

CS230: Deep Learning, Spring 2020, Stanford University, CA.

In addition, some other deep-learning based works focus on accessing sidewalk quality [11]. They use sidewalk focused images and classify them based on quality of sidewalks. Since they rely on sidewalk focused images and have different objective altogether, these works are beyond the scope of this work.

3 Dataset

Ideally, for sidewalk detection from drone, we needed a top view video dataset of streets. However, we could not find such a dataset. The closest one we found –and used in our project– is a synthetic dataset collected from Grand Theft Auto 5 (GTA5) [9]. This dataset includes high-resolution video frames in lossless PNG format (1920× 1080 pixels) annotated with ground-truth data that can be used for semantic segmentation. The data was collected while driving, riding, and walking a total of 184 kilometers in diverse ambient conditions in a realistic virtual world. The dataset is already split into training, validation, and test sets, which are further divided into a number of video sequences. Since our ultimate application, drone navigation, would have downward facing camera, we manually went through the data to identify top down view images. We found a few thousand images with views of sidewalks and their surrounding areas that are better suited to train, validate, and test our models. This subset of images was used to create our actual training/validation/test datasets. The following table gives some insights about our dataset.

	Training Set		**Validation Set**		**Test Set**	
	# Images	**% Images**	**# Images**	**% Images**	**# Images**	**% Images**
Total images	2147	62%	953	28%	350	10%
Lighting						
Day	1415	66%	635	67%	266	76%
Night	472	22%	138	14%	54	15%
Gloomy (snow/rain)	260	12%	180	19%	30	9%
Weather						
Snow	182	8%	0	0%	0	0%
Rain	78	4%	180	19%	30	9%

Table 1: GTA5 Dataset statistics

The GTA5 dataset has 30 labeled classes including objects such as person, animal, car, and tree, as well as environmental regions such as road, sky, sidewalk, and fence. Since we are interested in the identification of sidewalks, as well as detecting anything present on it or interfering with it, some of these classes would be useful in training our model. There are also two types of labels associated with each image. The first is a semantic class segmentation label, which is a single channel 8-bit image with mapping from label IDs to semantic classes and colors provided for each class. The second is a semantic instance segmentation label that is encoded as a 3-channel (RGB) 8-bit unsigned int image. Here, the first channel (R) encodes the class ID, and the two remaining channels encode the instance ID (= 256 × G + B).

(a) (b)

Figure 1: Sample Sidewalk Images from the GTA5 Dataset

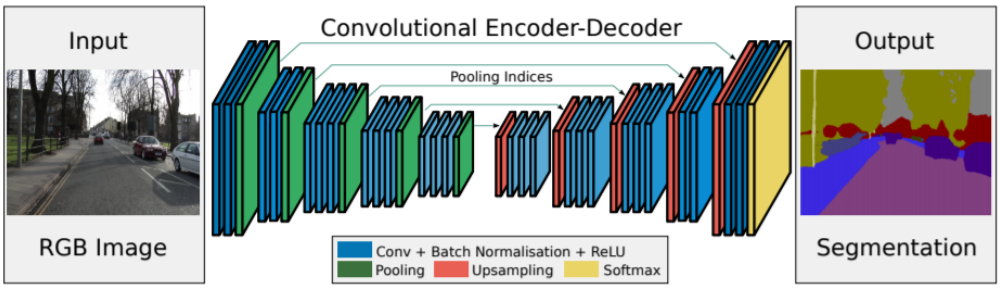
In most aspects, this dataset seems to be well suited for our task. However, we are aware that this is a synthetic dataset, which means our model is not guaranteed to work equally well on real world

images. Fortunately, if a better dataset is found, we can apply transfer learning to train our model for real images. In addition, this dataset has many more daytime images than night time images (or images in bad weather). However this did not seem to drastically negatively impact our performance in later tests, and thus, we found no need for data augmentation in this regard.

4 Methodology

4.1 Baseline

In the baseline implementation, we performed pixel segmentation using SegNet[10], by modifying the network to train on two classes (background and sidewalk).We started with a segmentation algorithm as it naturally lends itself to identifying groups of pixels that belong to a desired class (in our case, sidewalks). SegNet, in particular, was chosen due to its high accuracy and relatively small number of parameters as discussed in Section 2. Fig 2 provides a brief illustration of the architecture of SegNet.



(a)

Figure 2: SegNet architecture. It is purely convolutional network, without any fully connected layer. A decoder upsamples the input using the transferred pool indices from its encoder to produce a sparse feature map. It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification [1].

SegNet has an encoder network and a corresponding decoder network, followed by a final pixelwise soft-max classification layer that produces class probabilities for each pixel. The encoder network consists of 13 convolutional layers which correspond to the first 13 convolutional layers in the VGG16 network.Each encoder layer has a corresponding decoder layer and hence the decoder network (13 decoder layers). Each encoder in the encoder network performs convolution with a filter bank to produce a set of feature maps, which are then batch normalized, after which ReLU is applied. Then, max-pooling with a 2x2 window and stride 2 (non-overlapping window) is performed and the resulting output is sub-sampled by a factor of 2. The appropriate decoder in the decoder network upsamples its input feature map(s) using the memorized max-pooling indices from the corresponding encoder feature map(s), which produces sparse feature map(s). These feature maps are then convolved with a trainable decoder filter bank to produce dense feature maps. A batch normalization step is then applied to each of these maps. The high dimensional feature representation at the output of the final decoder is fed to a trainable soft-max classifier.The output of the soft-max classifier is a K channel image of probabilities where K is the number of classes. The predicted segmentation corresponds to the class with maximum probability at each pixel. SegNet uses all of the pre-trained convolutional layer weights from VGG net to begin with.

For baseline implementation, we did not any modifications to the original SegNet model. Additionally, we had selected a training set based on the requirements detailed in Section 3, but our validation set was randomly picked and did not contain many sidewalk images. We used the standard normal cross entropy as the loss function for training, which gave us a train accuracy of 95.6% and a validation accuracy of 96.4%. However, after looking into the accuracy metric defined by the model, we found that the large fraction of 'background' pixels lead to a higher weight on correct background classification rather than sidewalk classification. We also visually observed that the network performed really well on detecting the background, but not on detecting the sidewalk. Thus, the accuracy and loss results were inflated and we modified the model to fix them in subsequent improvements described below.

4.2 Improvement 1

To improve upon our baseline, we first changed our validation set to be as discussed in Section 3 to match the distribution of the training set. In addition, we added a few more classes while training to give the network more information about what a sidewalk is. For example, adding a pedestrian class could potentially tell the network that sidewalks are most likely found where the pedestrians are located. Moreover, we wanted the model to clearly differentiate between roads and sidewalks. On the other hand, adding too many classes could make the model take a very long time to train. For this reason we limited ourselves to adding 5 classes. The set of classes we used in this experiment were: *sidewalk*, *person*, *fire hydrant*, *road* and *background*. Finally, to address the issue of inflated accuracy, we changed the normal cross entropy loss to a weighted cross entropy loss. The weight for each class was determined [2] using the formula

$$\alpha_c = \frac{medianfreq}{freq(c)}$$

where α_c is the weight assigned to class *c*, *freq(c)* is the number of pixels of class *c* divided by the total number of pixels in images where *c* is present, and *medianfreq* is the median of these all *freq(c)*.

This improvement step not only resulted in better validation accuracy, but also increased our confidence in the model performance.

4.3 Improvement 2

Looking at the predictions and class accuracies of our model (Table 2), we felt that some of the classes chosen, such as fire hydrant, did not really help the model learn about sidewalks. So instead, we decided to use a more diverse set of classes to help network not only identify sidewalks but also objects on it (traffic signs, vegetation, barriers) and its vicinity (people, roads, vehicles, buildings). In addition, to reduce total number of classes, we combined the similar ones. We believed that this modification would result in more reliable and accurate sidewalk detection, without drastically increasing the training time. The classes we selected were: *sidewalk*, *vegetation* (which combines tree and vegetation on sidewalks), *vehicles* (which combines car, van, bus, truck, and trailer), *bikes* (which combines bicycle and motorcycle), and *background* (everything else). In particular, our hope was that the existence of vehicles would help the model distinguish between roads and sidewalks.

5 Experimental Results

The final accuracies of all the models are listed in Table 2. For a fairer comparison of all results with the baseline model, the baseline predictions were re-run through the new loss function to get the accuracy results and the results are listed in the table.

	Baseline	Improvement 1	Improvement 2
Training Accuracy (Overall)	91.1%	98.3%	93.9%
Validation Accuracy (Overall)	86.4%	92.7%	79.6%
Training Accuracy (Sidewalk)	98.0%	99.5%	97.9%
Validation Accuracy (Sidewalk)	97.0%	96.2%	92.6%

Table 2: Comparison of training and validation results. (Here, the overall accuracy numbers correspond to accuracy in identifying all classes present in the image.)

The baseline accuracies reported in the table are different from those reported in Section 4.1, since we ran them through the modified loss function for a fairer comparison. As expected, the overall accuracies are smaller than originally reported. Moreover, in each model, the total accuracy (which includes the accuracy of all classes) is smaller than that of the sidewalk class. This is because the training data we used was specifically chosen to maximize the performance of sidewalk detection. Objects such as pedestrians or fire hydrants only serve as additional markers to help identify sidewalks and images without these classes are marked as having "0" accuracy for the class, which reduces the overall accuracy of all classes.

Fig. 3 shows back-to-back comparison of baseline model, model Improvements 1 and 2 for two representative images each of training, validation, and test. In the figure, the sidewalk prediction

outputs are shown in red. We found that, among three models, Improvement 1 gave us the best sidewalk detection results, and thus, we have shown uncertainty output of Improvement 1 in the figure. We have provided a detailed result including uncertainty for all cases in the Appendix A. It can be seen that despite similar accuracy values, the sidewalk predictions for Improvement $1s$ are much more accurate to the ground truth than the baseline predictions. Therefore, we chose the Improvement 1 model to be our final model. Also, the test set from the GTA5 dataset is unlabeled, which is why the ground truth is all red.

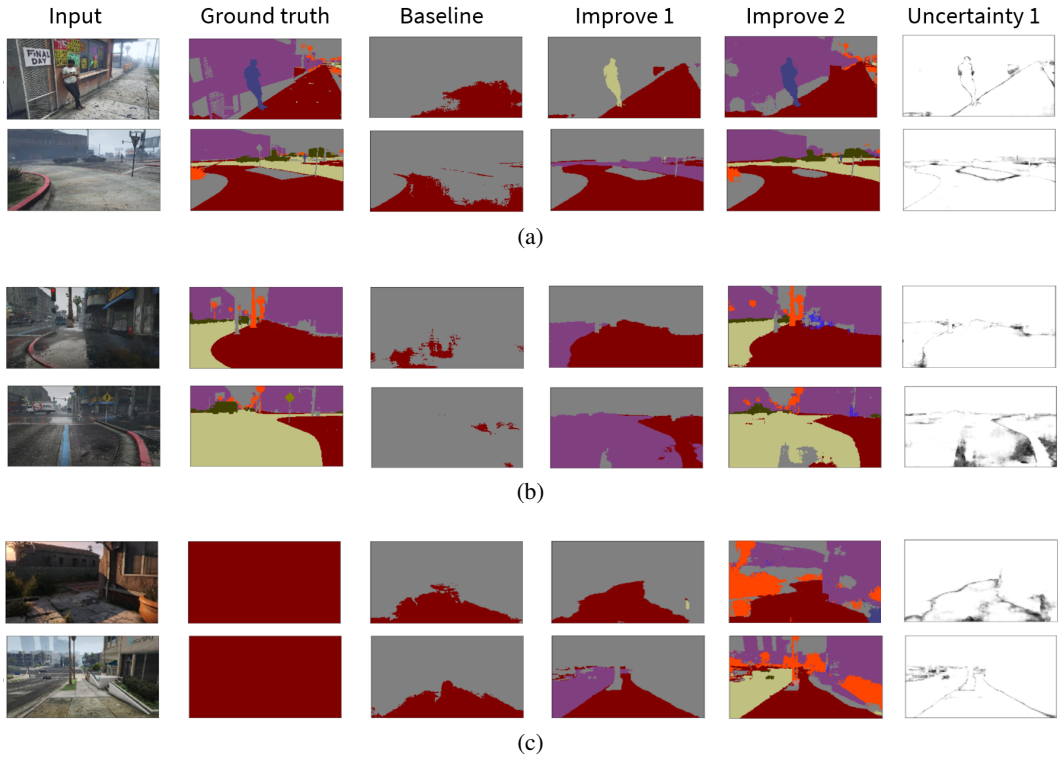


Figure 3: Sidewalk prediction results of baseline model, Improvements 1 and 2 for a) training set b), validation set, and c) test set. Among these three models, we chose Improvement 1 as our final model and for which we've also included the uncertainty in prediction.

6 Conclusion

In this paper, we've presented a sidewalk identification network that performs real-time sidewalks identification to facilitate drone navigation in urban and suburban areas. In this project, we were restrained to front-view synthetic images, though our model can be easily trained using transfer learning for more realistic real world top-view images. We stared our implementation from SegNet - a lightweight model that performs image segmentation for scene understanding. We carefully selected classes relevant to drone navigation on sidewalks, and appropriately modified the loss function for robust sidewalk identification. We changed the loss function from a standard normal entropy loss to a weighted cross entropy loss, which provided an improved metric for identifying sidewalk pixels. Afterwards, we trained two different improvement models on the dataset containing images from various weather conditions and tested the models on the training and validation sets. From the two models, we selected the model with better performance and used it for final prediction on the test set.

7 Contributions

All members contributed equally to the project. In particular, Tri worked on setting up the AWS infrastructure and implementing the Data Pipeline. Arvind worked on improving the Loss function, and Milan worked on Data and Model Analysis.

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015.

[2] David Eigen and Rob Fergus1. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture, 2015.

[3] Aharon Bar Hillel, Ronen Lerner, Dan Levi, and Guy Raz. Recent progress in road and lane detection: a survey. *Machine Vision and Applications*, 25(3):727–745, feb 2012.

[4] Yen-Chang Hsu, Zheng Xu, Zsolt Kira, and Jiawei Huang. Learning to cluster for proposal-free instance segmentation, 2018.

[5] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *CoRR*, abs/1511.02680, 2015.

[6] Yeongmin Ko, Jiwon Jun, Donghwuy Ko, and Moongu Jeon. Key points estimation and point instance segmentation approach for lane detection, 2020.

[7] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach, 2018.

[8] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial cnn for traffic scene understanding, 2017.

[9] Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2232–2241, 2017.

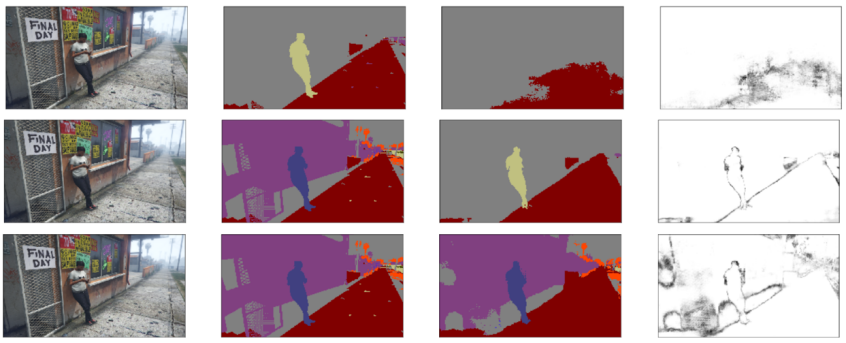
[10] Toimcio. toimcio/segnet-tensorflow, Jan 2018.

[11] *et al.* Weld, Galen. Deep learning for automatically detecting sidewalk accessibility problems using streetscape imagery. Association for Computing Machinery, 2019.

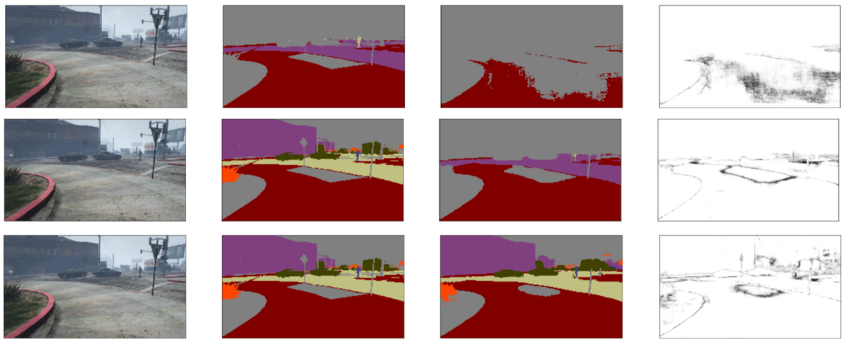
Appendices

A Detailed results

Provided below are detailed results for the summary in Fig. 3. In the figures, the four columns contain Image, Ground truth, Output, Uncertainty in output, respectively, while the three rows contain results for Baseline, Improvement 1, and Improvement 2, respectively. Each figure contains results of two different representative images.

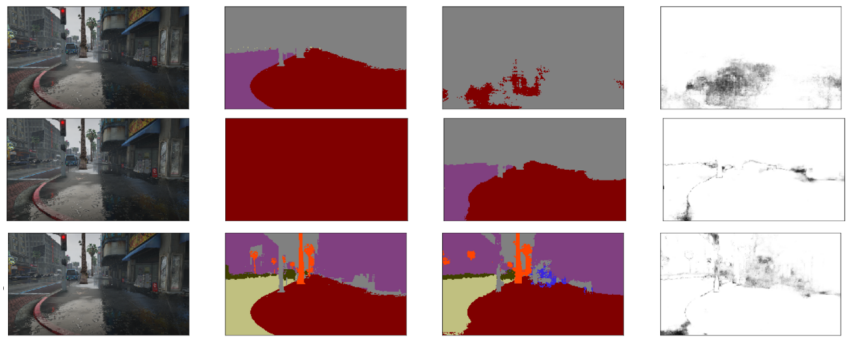


(a)

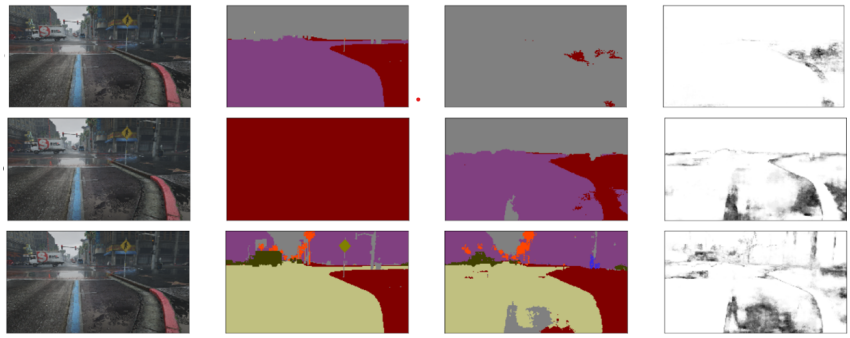


(b)

Figure 1: Extended representative output results of training.

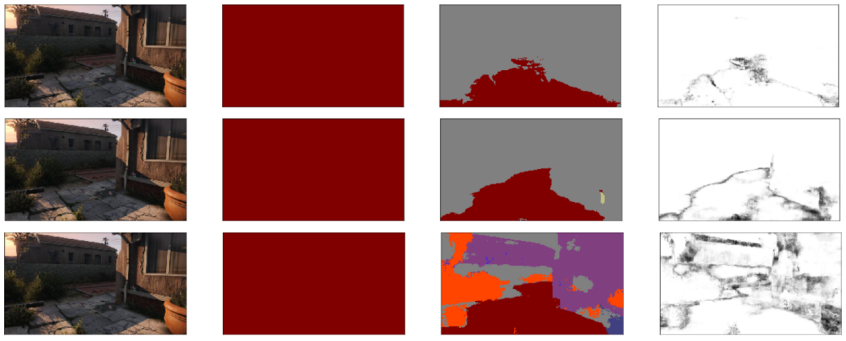


(a)

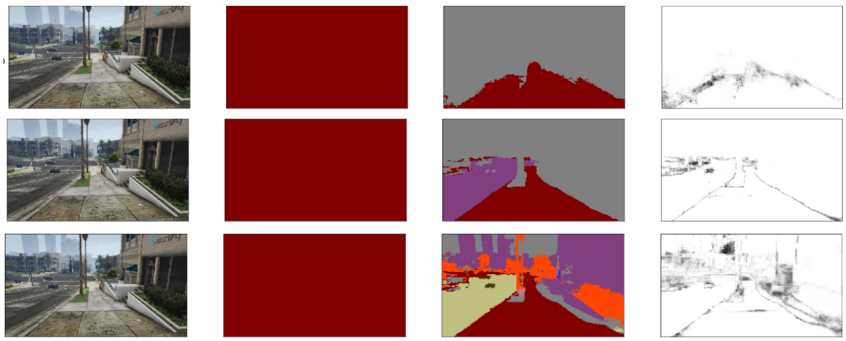


(b)

Figure 2: Extended representative output results of validation.



(a)



(b)

Figure 3: Extended representative output results of testing.