Road-Based Semantic Segmentation Experimentation

Chris Comeau, Brevin Lacy, Max Wang
MA416
11/16/2020

Abstract

For our project, we focused on the problem of semantic segmentation with application to autonomous driving. This entails classifying the pixels in an image based on the object that the pixel is part of. We chose to use a simple dataset in this domain that focuses on the classification of pixels in the image in a binary method: either the pixels are part of the road or they are not. We were able to perform experimentation involving a few established models in the field of semantic segmentation. We found that FCN8 resulted in the highest accuracy on the test set, but that UNet and SegNet had similar performances with significantly lower inference time, potentially making them more feasible to a real autonomous driving scenario. These models could potentially be augmented by adding more modern approaches that would potentially increase the complexity beyond the scope of this project or class.

1 Introduction

In a time of rapid automation of tasks to reduce the amount of human labor, driving has been a primary focus for data scientists and deep learning engineers. Over the past decade, there has been a vast increase in the interest in autonomous vehicles due to the cost-effectiveness of creating an alternative to human drivers. One way the autonomous driving field has been able to see such large-scale improvements is the progress made in the semantic segmentation of deep learning models. Semantic segmentation is the ability for a vehicular system to use visual input, either images or videos, to classify the pixels of an image based on the objects that it sees. Fascinated by this concept of challenging a computer's intelligence to match that of a complex human task, we decided to examine semantic segmentation models firsthand to see the success we could have in creating a system to recognize its surroundings.

The first step in creating and testing a model is finding a data source that has enough data points while also provides enough useful features for our model to process. We examined three well-known datasets commonly used to test autonomous driving models: CityScapes [1], KITTI [2], and nuScenes [3]. We decided to each investigate a dataset and discuss our findings. It was clear once we went over our results that KITTI was the best dataset to pursue. While CityScapes and nuScenes provided massive quantities of data and in-depth classifications tags, loading these datasets requires not only techniques and

nonstandard packages, but also a very long loading time (several hours for the CityScapes dataset, for example). On the other hand, the KITTI dataset is much more usable. Not only did it provide enough data (288 images for the training set and 300 images for the test set), its key targets are binary, meaning that each pixel is labeled either road or not road, which is easily trainable and testable for the scope of our project.

Figure 1. CityScapes [1] Example Image

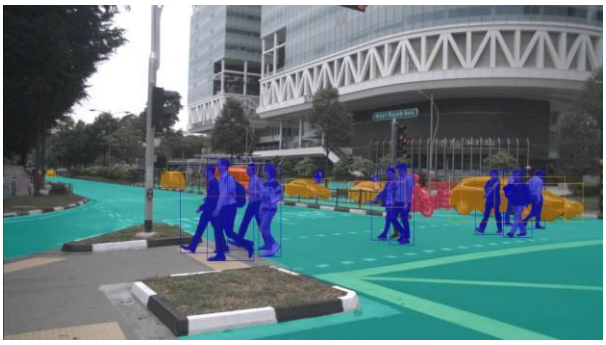

Figure 2. nuScenes [3] Example Image

Figure 3. KITTI [2] Example Image

Overall, our goal was to use this KITTI dataset and conclusively answer the question: can we make a model with similar intelligence to that of a human when it comes to recognizing the roadway for a vehicle?

2 Models

For this task, we decided to investigate the most common approaches taken for these applications across multiple domains. We found a specific subset of Convolution Neural Networks (CNNs) that are specialized in the unique problem of predicting each pixel of an image and we decided to focus on three categories of them: Fully Convolutional Networks (FCNs) [4], U-Net [5], and SegNet [6].

FCNs are a very popular and widely used network architecture for the task of semantic segmentation. As its name suggests, all dominant layers of an FCN are convolutional layers, which is also where it differs significantly from most of the network structures we covered in class (VGG16, for example). For most CNN architecture, the last several layers are usually fully connected, since spatial information is no longer needed, and the output of the last convolutional layer can be flattened to a large vector. For FCN, however, spatial information needs to be preserved as we need to reconstruct the full image from those layers. To do so, fully connected layers in CNN models are replaced with convolutional layers with a very small representation per channel (usually 1x1). Another major difference between FCN and the model we used in class is the existence of upsampling and transposed convolutional layers. Both are easy to understand, as they are the opposite counterpart of pooling and convolutional layers. These layers help to reconstruct full resolution images from the intermediate result of smaller but

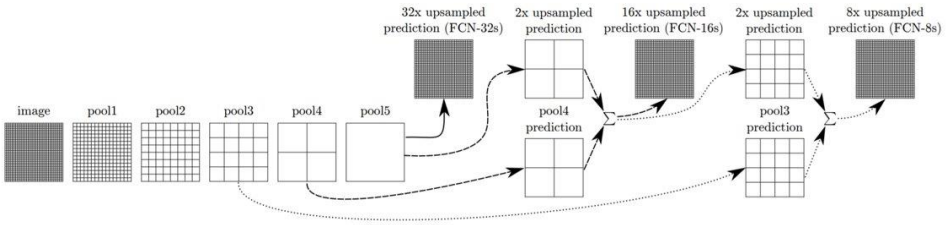

Figure 4. Network structure of FCN8, FCN16, and FCN32 [4]

deeper images produced by previous convolutional layers.

There exist several variants of FCN, with the most common ones being FCN32, FCN16, and FCN8. The difference lies in how many time upsampling is done in the reconstructing part of the network, with FCN8 having at most 8 upsampling at once and should, in theory, preserve more details of the original images and yield a better result, and FCN32 having at most 32 upsampling at once and lose the most details from the original images among its variants. UNet and SegNet share the same components as FCNs, both adopted the upsampling layers and transposed convolutional layers, but differ in the number combination of how many convolutional, pooling, upsampling, and transposed convolutional layers they use, as well as how results of lower-level pooling layers are used as supplementary details to help reconstruct the prediction images to their full resolution.

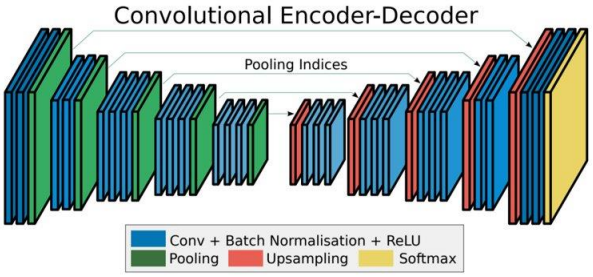

Figure 5. Network structure of SegNet [6]

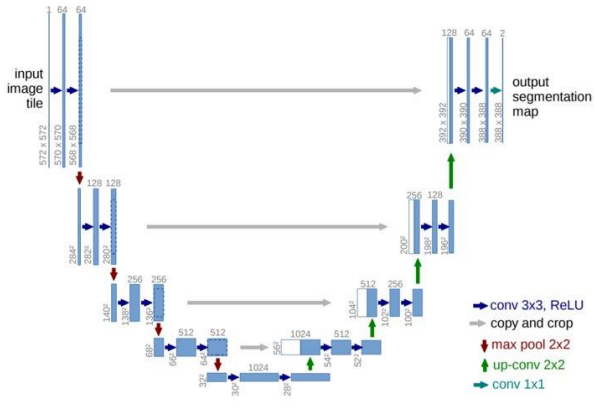

Figure 6. Network structure of U-Net [5]

3 Experiments

We performed 8 experiments using the dataset with the models previously shown. The models used in the experiments were FCN8, FCN32, UNet, and SegNet. Each of these models was used in two separate experiments, with two models in a pair and tested for a different purpose. For FCN8 and FCN32, we tested the vanilla versions of them (without pre-trained weight) and the versions using pre-trained VGG16. For UNet and SegNet, we also tested the vanilla versions but conducted comparison experiments with different optimizers, Adam and SGD, for each network structure.

All experiments were done in 5 training epochs, with 512 steps in each epoch and 2 as the batch size (2 images per step). We split the data of 288 images into 256 for training and 32 for testing, meaning that each training image is used 4 times in an epoch. Testing images were

never used in the training process, thus should have a good representation of how well the model would perform on images it has never seen. We kept the balance of the category of images when splitting the training and test set. All experiments were done on the GPU of Gauss (Tesla K80), each using one GPU at a time.

4 Results

4.1 Model complexity (this section should go with Models)

Out of more four models we tested, FCN8 has the most layers (44) and FCN32 has the least (34 layers). SegNet and UNet come in between 38 and 42 layers respectively. When it comes to the total number of parameters, FCNs are significantly larger than SegNet and UNet, with FCN32 having the most parameters (339,204,736) and taking up the most VRAM on GPU (about 11GB) and thus can only be run on the K80 on Gauss. Same with FCN8, which has 89,775,232 parameters, taking up about 8.6GB of VRAM. This is largely due to the huge transposed convolutional layers in these structures. For example, one transposed convolutional layer in FCN32 has parameters of shape (192,608,256), which by itself producing more parameters (268,435,456) than the whole SegNet or UNet model. SegNet and UNet have much fewer parameters in contrast, coming in at 3,844,160 and 4,618,304 respectively.

Model complexity comes not only in the number of layers and parameters but also in how layers are structured. This is where FCN8 and UNet are more complicated than SegNet and FCN32. Both FCN8 and UNet have concatenation layers, which neither SegNet nor FCN32 has. These concatenation layers take input from multiple layers, usually its previous convolutional layer and an earlier pooling layer, and concatenate or sum them to create the input of the next block. This allows the model to retrieve more details from earlier, less compressed images and makes it easier to reconstruct to the original resolution.

4.2 Model efficiency

Model efficiency can be evaluated with training time and inference time. Training time efficiency aligns very well with model complexity evaluated on layers and parameters. UNet and SegNet took only 184 seconds and 186 seconds per epoch as described above in the experiment section, and FCN8 and FCN32 took 675 seconds and 845 seconds per epoch, respectively. The total training time was about 18min for UNet and 19min for SegNet, whereas for FCN8 and FCN32 training took 57min and 80min.

Model efficiency evaluated on inference time during testing was also as expected. UNet and SegNet were able to conduct inferences for 3 input images per second, and the number dropped to 0.86 for FNC8 and 0.64 for FCN32.

4.3 Model accuracy

Model accuracy will be evaluated on both training and testing accuracy. All four models were able to achieve a decent training accuracy of over 95%, with SegNet taking the lead with 97.5% and FCN8 taking the least with 96.5%. This is not exactly what we expected based on analysis of these models, since FCNs should typically produce a better result than SegNet and UNet. One possible but unlikely explanation is overfitting, but with much more parameters FCNs should overfit easier than SegNet and UNet and should in theory achieve

better training accuracy. An explanation for this issue is the structure of the network since although FCNs have more parameters than the other two, its structure is relatively simpler and would not overfit as quickly. We would investigate this issue if we had more time.

Testing accuracy is exactly the opposite of training accuracy, with FCN8 taking the lead with 93.5% and SegNet taking the least with 90.5%. The result is what we expected: that with enough training epochs FCNs (in particular FCN8) would outperform UNet and SegNet. All four models achieved over 90% accuracy on new input images, making these models quite usable for semantic segmentation in autonomous driving.

TABLE 1
Results of each model where VGG versions of FCN have the weights from said model frozen. SegNet and UNet models were optimized using either SGD or Adam.

	road IoU	road IoU	mean IoU	Freq. wt IoU
FCN8	81.8	95.9	88.9	93.5
FCN8 (VGG)	10.2	66.1	38.2	56.5
FCN32	77.6	95.2	86.4	92.2
FCN32 (VGG)	11.1	81.5	46.3	69.5
SegNet (SGD)	77.7	94.9	86.3	92.0
SegNet (Adam)	72.2	94.3	83.2	90.5
UNet (SGD)	72.6	94.2	83.4	90.5
UNet (Adam)	78.6	95.4	87.0	92.5

5 Conclusion

This project seemed to be significantly more complex than any problem we had encountered in this class, even more so than the other image-based problems that we considered for potential projects. This meant we ran into complexity issues in multiple locations, both in the time taken to load the datasets and train the models, to the size complexity of these models. We had multiple issues with loading the CityScapes and nuScenes datasets in an easy and time-efficient way, with some datasets getting stuck loading for over an hour. This meant we had to focus on reducing the complexity of the dataset as much as possible to allow us to achieve some form of results. This was coupled with the models that are required to accomplish reasonable levels of accuracy in this domain. Many of these highly respected and well-known models that achieve high levels of accuracy are very complex with specific nuances to their construction and very large weight sets. This presented unforeseen issues with running on the servers at Rose-Hulman, as these models were too large to run on any of the GPUs other than those on Gauss, due to their larger VRAM. This meant that the training was slower due to the limitation of the older graphics card as well as the fact that we were only able to run two experiments at a time.

If we were to continue this project with more time and resources, we would experiment with other models and addons such as autoencoders [7] or Recurrent Convolutional Neural Networks (RCNNs) to move towards more current cutting-edge approaches for semantic segmentation. These would most likely require higher levels of complexity that is far outside of the scope that is feasible for this class, and most likely outside the scope based on our current understanding of deep learning. Nevertheless, these additional possibilities lend themselves well towards pushing the limits of model accuracy as well as bringing them closer and closer to being commonplace in autonomous driving applications.

6 References

[1] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth and B. Schiele, *The Cityscapes Dataset for Semantic Urban Scene Understanding,* 2016.

[2] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, "Vision Meets Robotics: The KITTI Dataset," *Int. J. Rob. Res.,* vol. 32, p. 1231–1237, 9 2013.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan and O. Beijbom, *nuScenes: A multimodal dataset for autonomous driving,* 2020.

[4] J. Long, E. Shelhamer and T. Darrell, *Fully Convolutional Networks for Semantic Segmentation,* 2015.

[5] O. Ronneberger, P. Fischer and T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation,* 2015.

[6] V. Badrinarayanan, A. Kendall and R. Cipolla, *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,* 2016.

[7] M. A. Islam, M. Rochan, S. Naha, N. D. B. Bruce and Y. Wang, *Gated Feedback Refinement Network for Coarse-to-Fine Dense Semantic Image Labeling,* 2018.