Using FCNs to identify roads from satellite imagery

Carlos Daniel Mondragon Chapa
University of Massachusetts, Amherst
cmondragonch@umass.edu

Salar Satti
University of Massachusetts, Amherst
salarsatti@umass.edu

Abstract

We aim to semantically segment a road and cars in an image by implementing a Fully Convolutional Network that is, given an image, we are able to predict which pixels of the image are roads and cars. We use 125 training images of satellite imagery from the Google Maps API and create the training masks manually. We also use transfer learning and VGGNet trained on a classification dataset to boost accuracy, getting extremely good results. Finally since we have successfully identified roads and cars with a high degree of accuracy, we propose a simple way to identify the level of traffic in a certain image and propose ways of how this information can be used by multiple stakeholders.

1. Introduction

Neural Networks and Deep Learning have recently been gaining ground in solving many problems in different industries and in solving many different types of problems within industries from NLP in financial analysis [3] to image analysis in self-driving cars [2]. Neural Networks are increasingly solving problems once thought intractable.

We wanted to use Neural Networks to help stakeholders understand traffic in their city. One of the ways we can do this is through aerial observations of urban areas. We initially decided to use satellite imagery to take aerial photos but we envision this to be easily expanded to drones with cameras and CCTV cameras as well.

Traffic is a persistent problem in most cities in the world. Congestion, pollution [9], frustration [7], and lost productivity [8] are a few of the negative caused by traffic. Several solutions have been proposed to improve flow of traffic: increase availability and access to public transportation, enlarge infrastructure budgets, introduce congestion charges or highway tolls, and many others [6].

However, making these changes is often expensive and at times unfeasible due to the state of the current infrastructure. An alternative approach is to improve the traffic flow by optimizing how traffic flows in the current infrastructure. To optimize any system, we need to have more information about agents operating inside the system. We could optimize this system by making traffic signals responsive to the level of traffic, direct traffic police officers, direct emergency services, apply it to personal GPS navigation, and use the information for long-term infrastructure planning. We initially aim to use neural networks and satellite photography to automatically detect roads, detect cars, and determine the level of traffic in city streets.

The satellite images should contain areas with traffic and no traffic, so that the neural network is able to segment both areas. The long term goal is to define an algorithm that can use the traffic level information at different locations to optimize the overall flow of traffic, but this is beyond the scope of this paper. In this paper we aim to only detect the level of traffic using satellite imagery.

2. Background/Related Work

Our project closely follows the methodology described in the Fully Convolutional Networks for Semantic Segmentation paper by Johnathan Long, Evan Shelhamer and Trevor Darell at Berkeley [5]. We also looked at some implementations and projects online [4] that implement the approach for semantic segmentation mentioned in the paper. We were not able to find a project that did semantic segmentation of roads from satellite images, however, one of the projects we found, did semantic segmentation via Fully Convolutional Networks from footage from dash a cam for autonomous driving [4]. The images were taken from the dash of the car and classified into multiple objects such as cars, roads, and pedestrians. The results are very impressive as they report to have an accuracy of over 90 percent for their test set [4]. The paper by Long and others [5], explains how transfer learning can be used in semantic segmentation. The idea is to extract layers at different points in the neural network after a pooling layer. By doing this, we can get features from the first layers that are usually learned in any neural network that is trying to classify an image. Later layers in the network learn features for a particular task. According to the the paper how transferable certain features in deep neural where they are in the netwrok networks [5]. The first few layers of a pre-trained model usu-

ally contain more generalizable features such as detecting curves and lines. The final layers are not as generalizable, unless the new task is similar to the one of the pre-trained model. Another advantage of using transfer learning is that even after transferring these parameters and tuning them for a different task, some generality is kept and that is good the model is validated.

3. Approach

We tried four different approaches which all involved using Fully Convolutional Networks. The reason behind using Convolutional Layers was that they used less parameters than Fully Connected Layers and are easier to train. This was very important since our images were very large: Height: 640, Width:640, Depth:3. Additionally, Fully Convolutional Networks have the advantage of handling input of any size, so if at some point the image size changes, the architecture of the Neural Network does not need to be modified. The first Fully Convolutional Architecture we designed ourselves. Each image was labeled pixel wise: Road, and No Road. The design of the architecture was based on the idea of taking a large image, convolve it with different kernel sizes, applying a ReLU non-linearity, and using max-pooling to downsample it. This process was performed until the image was downsampled by a factor of 8. This way, the amount of computation needed was reduced, as there were less parameters to learn. Afterward, the image was upsampled again gradually to its original height and width. We used Nearest Neighbour as upsampling method. Figure 1 shows our first approach.

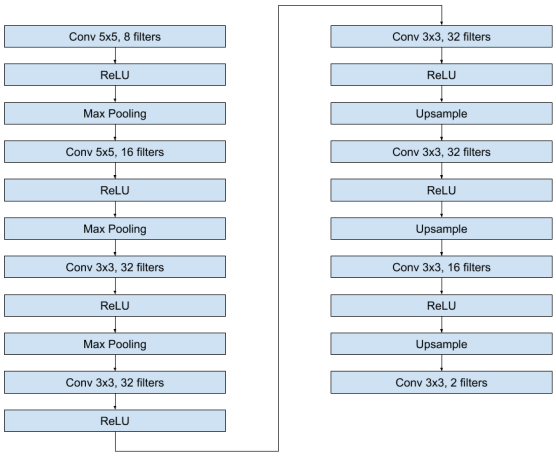

Figure 1. First Architecture

For the second architecture, we changed the way in which the image was upsampled. Instead of upsampling using the Nearest Neighbor, we used Transposed Convolution. The rest of the architecture stayed the same.

For our third architecture, we decided to do transfer learning from VGG16. We could not use a bigger model with more parameters since it would not fit in the memory of our GPU. We first extracted layers at different points from the architecture. All layers but one did not have the width and the height of the input image, so we performed the upsampling starting from the lowest resolution layers using transposed convolutions all the way to the coarser but higher resolution layers. At each upsampling step, we applied a ReLU non-linearity followed by a 2D batch normal-

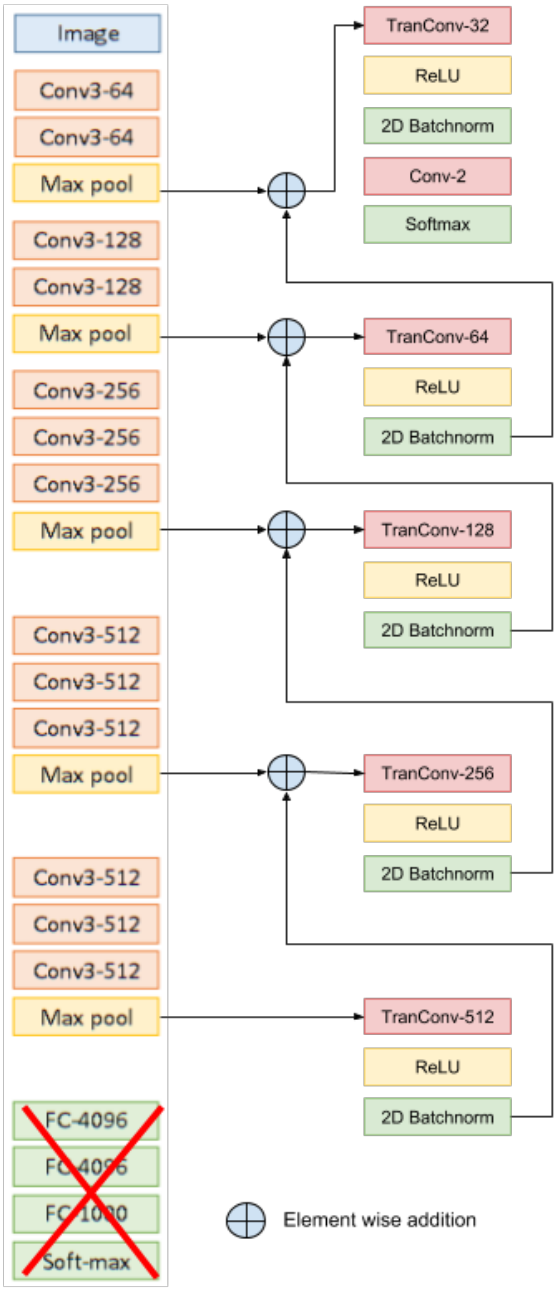

Figure 2. Transfer Learning Architecture

ization. We then added this resulting layer with the next higher resolution layer until we had an output that had the same height and width as the input. We finally removed the Fully Connected Layers and replaced them with a Fully Convolutional Layer which used 1x1 convolutions to reduce the depth of the tensor to be of size 2, since we only wanted to classify as Road and No Road, or in the case of car classification, Car or No Car. Figure 2 shows this architecture.

For our last architecture we decided to do something innovative and new. We took the VGG16 architecture and stuck our first architecture, our own Fully Convolution Network, at the end. We were interested to see what would happen, but essentially we were hoping that the the model could use pretrained features from VGG16 and also learn new features specific to the dataset from scratch.

Finally, we wanted to use the most successful architecture to create two different models. The first model predicts road pixels and the second model predicts car pixels. Using this approach, predicting the level of traffic becomes simpler. This is because predicting the level of traffic becomes complicated by the fact that we only want to consider cars that are on the road, and not in parking lots. Using two model, the neural network predicting car pixels were does not need to worry if they were on a road or parked in a parking lot, it would simply identify every car in the image. Later on, the results of the Neural Network predicting roads would be combined with the results of the one predicting cars. Therefore, only the car pixels intersecting with road pixels are taken into consideration to estimate the traffic level. Using this method traffic estimation becomes very easy, we simply take the number of car pixels intersecting with road pixels and divide it by the number of road pixels. The car pixels to road pixels ratio tells us the traffic level.

4. Experiment

Due to the lack of an existing data set we decided to create it ourselves. We used the Google Maps API to download Satellite images of the down town Manhattan area in New York City. We then manually created masks for each image using Photopea [1], an online photo editing tool. We did one mask for the road and one mask for the traffic in each image. An example of a raw image can be viewed in Figure 3, an example of a mask of a road can be viewed in Figure 4 and an example of a mask of the traffic can be viewed in Figure 5.

In each of our experiments we centered and scaled our training data using the mean and standard deviation. We used PyTorch to conduct all our experiments. Additionally, we trained every model on a Nvidia GTX 1050 GPU for no more than 2 hours. We also used a per-pixel cross entropy loss for each of our models since we were able to converge with reasonable speed and accuracy. Our first few experiments focused on getting good accuracy for predicting roads. Once we were able to predict roads successfully we applied the same architecture to predict cars. We finally combined both results to calculate the level of traffic.

Figure 3. Example of satellite image downloaded from the Google Maps API of size (640x640x3)


Figure 4. Example of a manually created mask for a road created from Figure 3 of size (640x640x3)

In our first experiment we used our own Fully Convolutional Nets architecture with Nearest Neighbour upsampling procedure. We used a learning rate of 5e-3, we did not use a regularizer of any sort, we used the Adam opti-

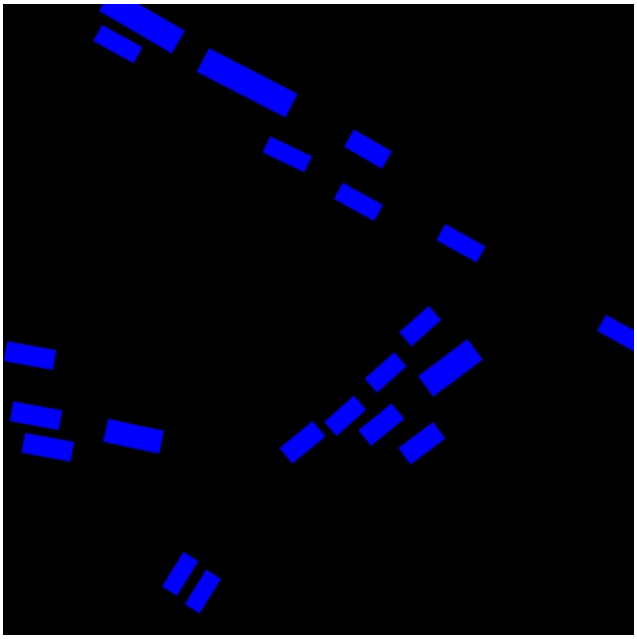

Figure 5. Example of a manually created mask for traffic created from Figure 3 of size (640x640x3)

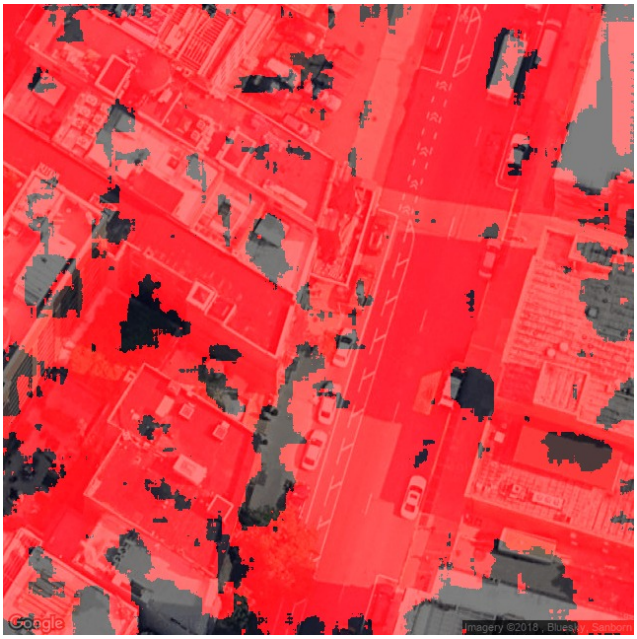

Figure 6. Prediction of a road using our first model (FCN with 5 training examples).

mizer as our step algorithm, and we used only 5 training images. Since the images were of relatively high resolution and we trained for a long time we were hoping that our model would be able to learn to a point where it could make reasonable predictions. Our accuracy on the training images was in the high 90s so we knew the model was converging. However, the predicted result for a validation image was very poor. The result for one validation image can be viewed in Figure 6. Additionally, we tried several different learning rates and different optimizers but it showed no improvement in accuracy.

In our second experiment we again used the same architecture as in our first experiment, our own Fully Convolutional Nets architecture with Nearest Neighbour upsampling. We also increased the size of the training data form 5 to 125. We again attempted many different learning rates and optimizers but using a learning rate of 5e-3, an Adam optimizer, and no regularization worked the best. This vastly improved the accuracy of our model, mainly because of the increase in training images. The result of these experiments for one validation image can also be viewed in Figure 7 and the Intersection Over Union results can be viewed in Table 1. From Figure 7 we can tell the model is able to detect the road much more effectively by increasing the size of the training data set. We knew that by having more training examples we would continue to increase the accuracy of our model but it would be at a diminishing rate, thus we decided to try a different technique.

In our third experiment we used the same architecture and hyperparamters as the second experiment but we used

Intersection Over Union Statistics			
–Model 1–	–Model 2–	–Model 3–	–Model4–
0.9271	0.9105	0.9809	0.9853
0.9091	0.8912	0.9531	0.9432
0.8658	0.9091	0.9549	0.9849
0.8557	0.8787	0.8862	0.9173
0.8857	0.8583	0.8553	0.9074
0.8739	0.8718	0.952	0.9533
0.8801	0.8262	0.9223	0.9349
0.9428	0.9311	0.8828	0.9481
0.8252	0.866	0.9146	0.8618
0.9353	0.9354	0.9476	0.9605
Average	Average	Average	Average
0.8901	0.8878	0.9250	0.9397

Table 1. Intersection over union results for each model. Model 1: Our FCN architecture with Nearest Neighbour upsamlping, Model 2: Our FCN architecture with Transposed Convolution upsampling, Model 3: VGG16 FCN with Transfer Learning, Model 4: VGG16 with our own FCN architecture at the end

transposed convolution as our upsampling procedure. The Average Intersection Over Union results for this model can be viewed in Table 1. We hoped this would increase accuracy but it was not the case. We also tried different hyperparamters but we could not increase the accuracy further.

In our fourth experiment we used transfer learning. We used the VGG16 model pre-trained on an image classification data set. We again trained on our entire data set, used a learning rate of 5e-3, an Adam optimizer, and no regularization. The result for one validation image of this model can

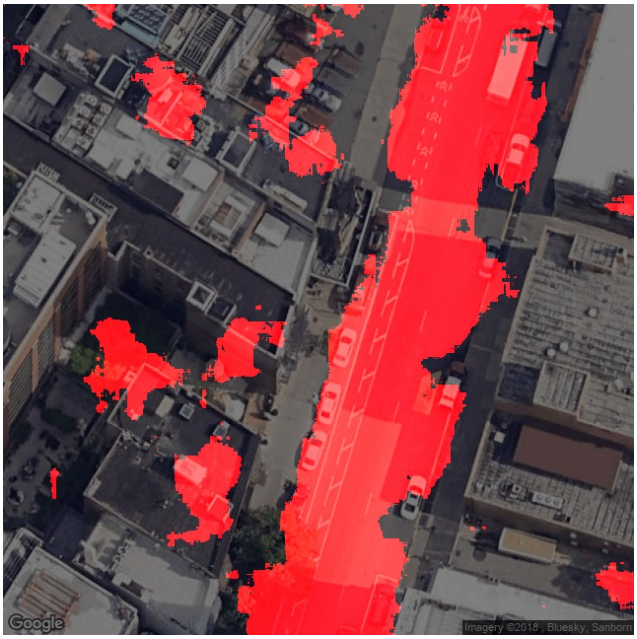

Figure 7. Prediction of a road using our second model (FCN with 125 training examples).

be viewed in Figure 8. The intersection over union results can be viewed in Table 1.

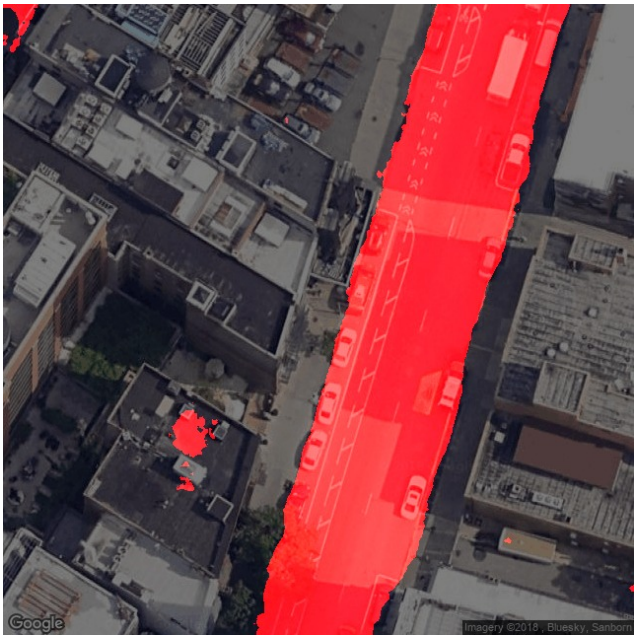

Figure 8. Prediction of a road using our third model (Transfer Learning using VGG16 and 125 training examples).

In our last experiment we use transfer learning from the VGG16 model pretrained on an image classification data set and then stuck our own own Fully Convolutional Network as the last few layers which was not pretrained. We trained on our entire data set, used a learning rate of 5e-3, an Adam optimizer, and no regularization. As we hoped, this model performed better as we can see from the intersection over union results in Table 1. The result for one validation image of this model can be viewed in Figure 9.

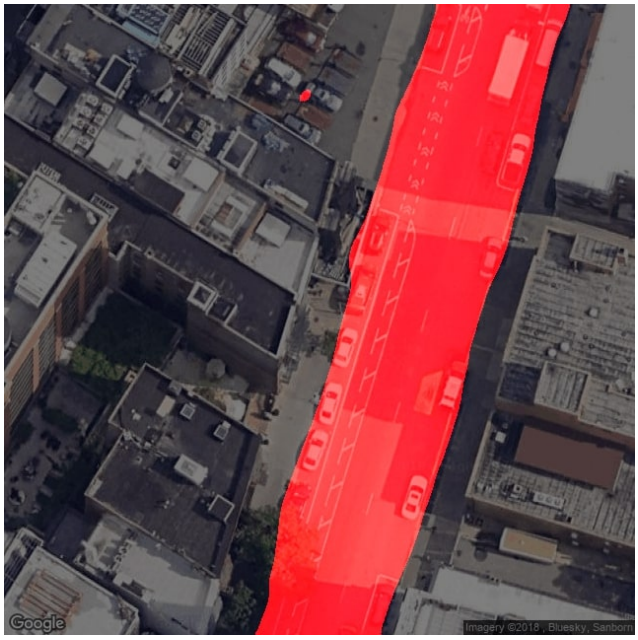

Figure 9. Prediction of a road using our fourth model (Transfer Learning using VGG16, with our own FCN at the end, 125 training examples).

Our combination of pretrained VGG16 and our own Fully Convolutional Network combination worked slightly better than just using pretrained VGG16 as can be viewed in Table 1.

These results definitely exceeded our expectations for identifying roads in images and decided to try the VGG16 pre-trained model with our Fully Convolutional Network at the end to identify cars. We created 63 masks like the one shown in Figure 5 that identified cars in the images and re-trained the model using a learning rate of 5e-3, Adam Optimizer, and no regularizer. The result for one validation image of this model can be viewed in Figure 10.

Once we were able to successfully identify roads and cars, we measured the level of traffic on the roads as was mentioned in the previous section. The results of this approach can be found in Table 2 and the concept is illustrated in Figure 11. The blue and red pixels are are the pixels identified as the road, and the blue pixels are the pixels of only the cars on the road. Note in Figure 11 near the top, the parked cars in the parking lot have not been selected as car pixels, even though they are identified in Figure10. Also this methodology helped us mitigate some errors in the prediction of the neural networks. In some cases, the image would think that certain objects on rooftops were cars. Sim-

Figure 10. Prediction of cars using our third model (Transfer Learning using VGG16 and 125 training examples).

Traffic Levels	
—-Image Number—-	—-Traffic Level—-
1	11.9%
2	8.0%
3	7.5%
4	13.4%
5	7.2%
6	6.9%
7	6.8%
8	15.7%

Table 2. Traffic Levels as a ratio of car to road pixels for 8 validation images. The images can be seen in Figure 11.

ilarly, the other neural network tried to find roads hidden by trees, which was in some cases inaccurate. However, by combining both results, most of the errors were not taken into account when predicting the traffic level. Note that only the cars highlighted in blue are used to measure traffic level, cars out of roads or objects that are identified as cars but are not on the roads were not taken into consideration.

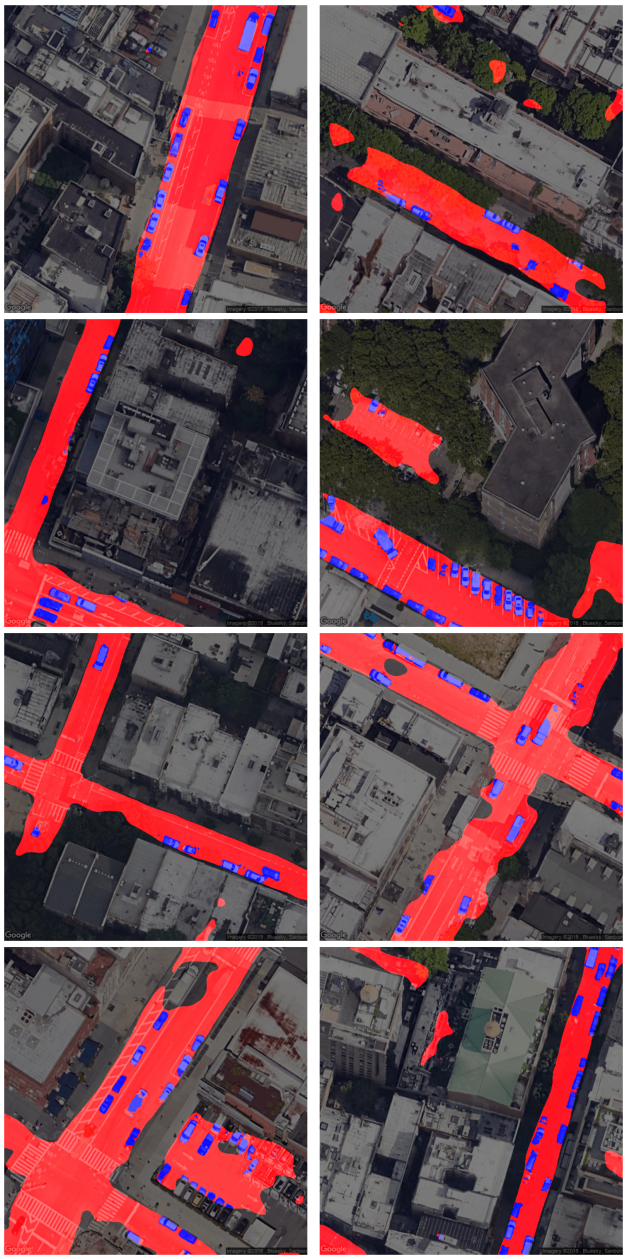

Figure 11. Prediction of a roads and cars using our fourth model. Transfer Learning using VGG16, 125 for training images for roads, and 63 training images for cars (Image 1 is top left, Image 2 is top right, and so on).

5. Conclusion

We learned that in order to train a model from scratch to do semantic segmentation we likely needed thousands of images. Therefore, we were very surprised when we realized how powerful transfer learning actually was when done correctly. We only had 125 and 63 training examples for roads and cars respectively and that still gave us very good results. Specifically we believe our model (VGG16 with an extra Fully Connected Network added at the end) gave us good results for two possible reasons. First, that simply having a model with more hyperparamters gave us a better model or second, that transferring features and learning features from scratch both are helpful in building accurate models. We would like to explore this in future work.

We would have liked to check the amount of necessary training images to get good results using transfer learning,

but we did not have enough time to create these. Additionally, increased training images would increase the running time of our model and since it already took around 2 hours per architecture to train, we decided to not create more. It would be interesting to see the results with a very large dataset if we had more resources.

One of the things we would change is the way in which the masks were created. For instance, some of our masks included parking lots as roads, which we then realized was wrong. Also, in some cases, even when the road was hidden behind trees, we assumed that the road continued; this was also wrong, because there was no way for us to know for sure whether that was true or not and gave us no information in terms of traffic level. We saw these mistake more clearly when we started looking at validation examples; in many cases, the neural network thought that there were roads hidden by trees, when there were none. This did not affect so much our final aim which was to measure the level of traffic, but we could have better results in predicting exactly where the road is. Looking at other semantic segmentation projects, we realized that they have a whole document with a set of rules describing how to create training images. If we had created such a document, we would have gotten better results.

One extension we would have liked would be to try different zoom levels of the satellite imagery. We thought that this could potentially increase the accuracy of the model and also find the zoom level that best describes traffic level.

Finally, we would also want to get images from street cameras or drones to measure the amount of traffic in real time using semantic segmentation. This would help compile a more accurate view of traffic and thus would result in better optimization of traffic flow in city streets.

References

[1] Photopea. *https:www.photopea.com*, 2018.

[2] M. Bojarski, D. D. Testa, D. Dworakowski, et al. End to end learning for self-driving cars. *Nvidia Corporation*, 2016.

[3] X. Ding, Y. Zhang, T. Liu, and J. Duan. Deep learning for event-driven stock prediction. *Research Center for Social Computing and Information Retrieval, Harbin Institute of Technology, China*, 2015.

[4] P.-C. Huang. Fcn-pytorch. *https://github.com/pochih/FCN-pytorch*, 2017.

[5] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. 2015.

[6] K. Mahmud, K. Gope, and S. Chowdhury. Possible causes solutions of traffic jam and their impact on the economy of dhaka city. *Journal of Management and Sustainability*, 2012.

[7] D. Ouis. Annoyance from road traffic noise: A review. *Journal of Environmental Policy*, 21:101–120, 2001.

[8] M. Sweet. Does traffic congestion slow the economy? *Journal of Planning Literature*, 2011.

[9] XiangluHan and L. P.Naeher. A review of traffic-related air pollution exposure assessment studies in the developing world. *Environmental International*, 32:106–120, 2006.