Torch Enjoyers Final Report

Michael Garbus
Torch Enjoyers
University of Illinois at Urbana-Champaign
Urbana, Illinois
mgarbus2@illinois.edu

Petar Becarevic
Torch Enjoyers
University of Illinois at Urbana-Champaign
Urbana, Illinois
petarb2@illinois.edu

***Abstract*—Convolutional Neural Networks (CNNs) have been dominant in segmentation tasks for some time already. Along with changes to CNNs, different and novel loss functions have been proposed in combination with different layer architectures to create CNNs that are more powerful than ever before. We would like to apply one of these newer architectures, DeepLabv3 to semantic segmentation for road scenes based on the KITTI-360 dataset, with a real-life application of Autonomous Driving. Furthermore, we will be applying transfer learning to our dataset based on DeepLabv3's original training set, COCO 2017. We will analyze the results obtained from training different classes and analyze how the network performs using pretrained weights. The point of transfer learning on this segmentation is to understand how well the network performs when given a new cost surface to minimize.**

Keywords—Autonomous Driving, transfer learning, Convolutional Neural Network (CNN), segmentation, deep learning, artificial intelligence (AI)

I. Introduction

The problem topic we are addressing is segmentation in regard to autonomous driving. We are interested in this because of the implications this could have on computer vision and autonomous driving progress. Performing segmentation is important to the safety of autonomous driving along with improving the current autonomous driving methods. This problem is challenging to our group because we have a large amount of data that will need to be trained on HAL in addition to implementing a neural network that can deal with spatio-temporal data along with generalizing properly. The expected network is a fully convolutional network that has Resnet like architecture that uses bottleneck layers and down sampling. We are investigating the deeplabv3 Resnet50 like architecture that utilizes Atrous Convolutions along with Atrous Spatial Pyramid Pooling (ASPP). The Resnet backbone aids in feature extraction. The atrous convolutions help to control the size of the feature maps while a ASPP layer towards the end helps to classify pixels into classes by summed dilation convolutions. The ASPP layer is finally passed through a 1x1 convolution that yields the correct image size and segmentation mask.

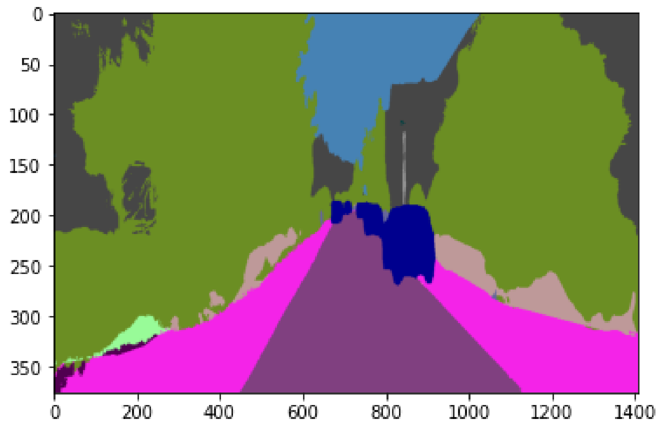
II. Related Works

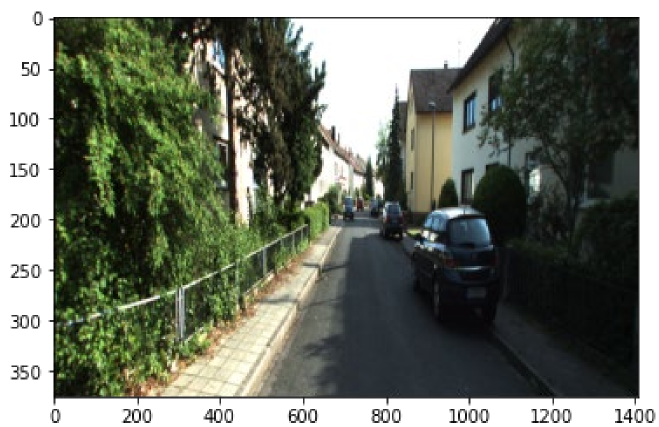
There have been updates to the DeepLabv3 architecture used on popular datasets for semantic segmentation such as PASCAL VOC 2012 and the Cityscapes dataset, "in this work, we propose to combine the advantages from both methods. Specifically, our proposed model, DeepLabv3+, extends DeepLabv3 by adding a simple yet effective decoder module to refine the segmentation results especially along object boundaries" (Chen et al.). This is of importance because Cityscapes classes overlap with the Kitti-360 dataset classes and Kitti-360 contains the classes that Cityscapes does in addition to three additional classes. Additionally, as we have seen previously with PointRend (as discussed in Kirillov et al.), the module was used in combination with DeepLabsV3 on the Cityscapes dataset to yield segmentation boundaries that are fine-tuned with regards to the objects, and particularly smaller objects when used with semantic segmentation. The common ground here is that a previous network has been established and improved upon for some gains in prediction performance, whereas our project will be attempting to increase efficiency on new unseen data by transfer learning from previous similar tasks. The goal is to achieve an efficient prediction with minimal hyperparameter tuning for marginal gains.

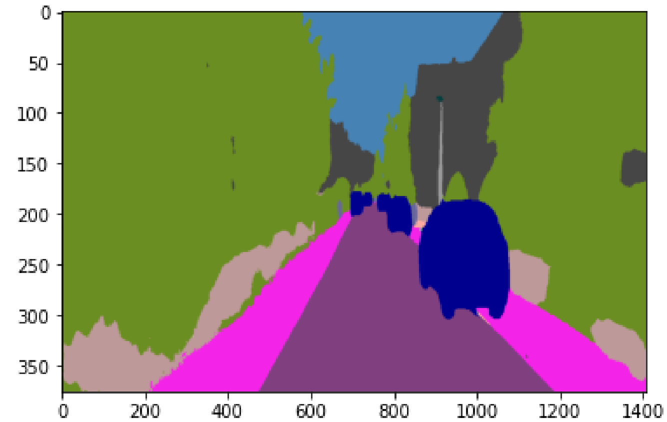
III. Data

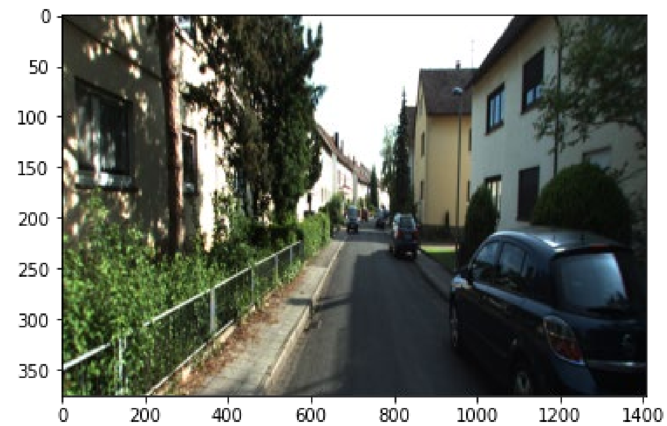
This project deals with mask data and raw image data, from a dataset from the KITTI-360 dataset presented by Yiyi Liao and Jun Xie and Andreas Geiger, "KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D". This data can be found at http://www.cvlibs.net/datasets/kitti-360/download.php, after registration. The resolution of the RGB images is 1408x376. The data is around 100 gigabytes in size, and is spread out across nine folders, containing both validation and training data. There are 49004 training images, 12276 validation images, and 910 testing images, meaning there is a total of 61280 images with approximately 79% belonging to training data, 20% validation data, and 1% on testing data. In regards to preprocessing, we first had to remove extraneous files that were not in our training, testing, or validation datasets. For loading in the data, we need to load images with a normalization provided by the DeepLabv3 architecture, with a mean = [0.485, 0.456, 0.406] and standard deviation = [0.229, 0.224, 0.225]. For our transformations, a color jitter with the settings (brightness=.1, hue=.1, contrast=.1, saturation=.1) and random sharpness of with the settings(sharpness_factor = 1.5, p =.5) via the torchvision package. Additionally, we do a center crop with dimensions (1011x270) via Albumentations to transform both images and the mask. The data had to be loaded in via the modification of the PyTorch DataLoader class, as shown in our code, in order to accommodate our specific directory structure. Additionally, since the training data and validation data (both contain raw image data and mask data) were in the same folders and had no distinction, we had to read data from a text file to create folders of training data and validation data. As the test data is already separate, we did not have to do this part for the

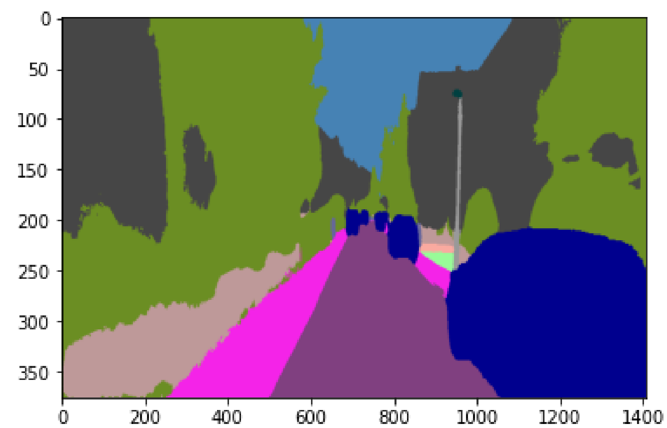
test data. Below are examples of three training files, showing the raw and mask data. These images in specific are the 1st, 5th, and 10th images from the first training data batch.











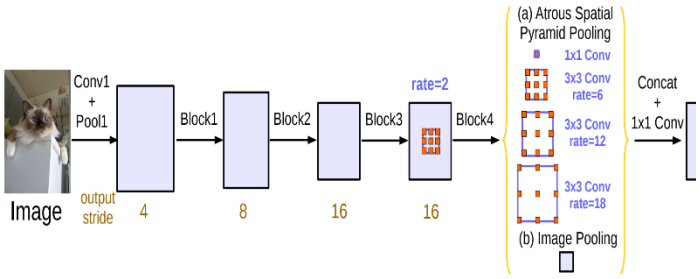
IV. Methods

Since we have used DeepLabsV3 with a ResNet101 backbone, the way this network works on KITTI-360 images is that the images are passed through the ResNet101 backbone. The network begins with a 2D convolution with a kernel size of 7x7, the input channel size is 3 and the filter size is 64, a stride of (2,2), and a padding of (3,3). The convolution is followed by 2D batch norm layer and a ReLU. The first block ends with a 2D max pool with a kernel size of 3, stride of 2, padding of 1, and dilation of 1. The architecture then has 4 separate layers that each contain bottleneck blocks. Each layer starts with a

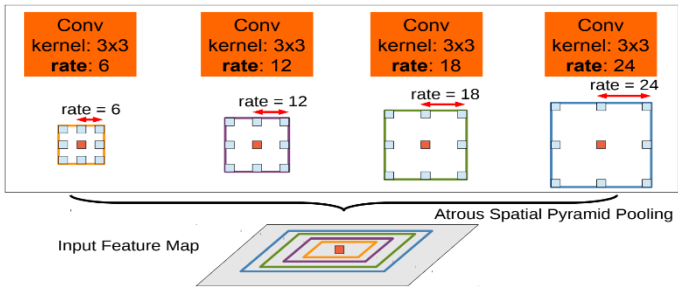
bottleneck block that also down samples the feature map with a 1 by 1 convolution with a stride of 1. The third layer contains many more bottleneck blocks than any other layer with a total of 23.

After the bottleneck layers, the DeepLabsV3 contains an ASPP module which contains a same convolution with a stride of 1, followed by dilated convolutions that contain equivalent padding and dilation rates of varying degrees. Finally, there is an adaptive average pooling that has an output size of 1. These layers are concatenated together to provide a larger representation of the feature map. The final layers of the model include a convolution, batch norm, ReLU, and dropout with probability of .5. Finally, the model contains a layer of a convolution, batch norm, and ReLU that is passed into a decision layer with an input channel size of 256 and an output size for our dataset of 21 classes.

It was difficult to represent the DeepLabsV3 model using the visualization tools provided, as ASPP is a unique convolutional tool. Below is an image from the Chen et al. "Rethinking atrous convolution" paper that shows the ASPP module in the model.



Below is an image from Chen et al. from the initial "DeepLabs" paper and provides a straightforward explanation of how ASPP works.



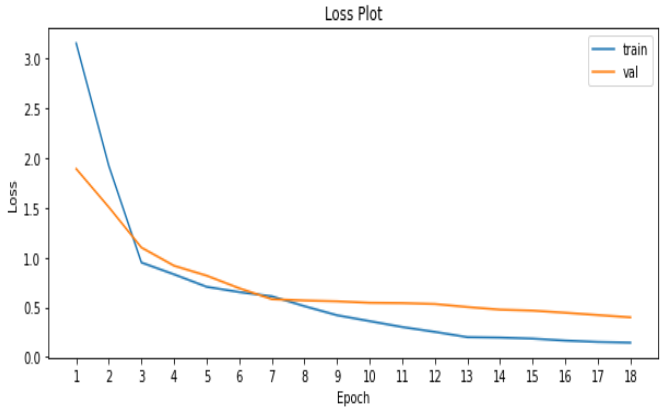
Using DeeplabV3 is an effective model for this purpose. Down sampling the feature map is important, as pixelwise class representation requires in-depth knowledge of pixel values. Additionally, spatial representation is required to achieve larger feature map representation and is provided via ASPP. Some alternative approaches could be using a larger ResNet backbone, such as ResNet 152, however, this would take more training time as there would not be an opportunity to use transfer learning. The learning rate could range anywhere from 1e-3 to 1e-5, and weight decay could be 1e-4 or 1e-5. Optimizer-wise, Adam and SGD could be used. Another model that could be used for this purpose would be U-Net. Although U-Net does perform upscaling, it does not perform it as significantly as ASPP, so may not receive as good results as the Deeplabv3. Regardless, we feel that the same type of learning rates and weight decay could be used, but with a higher number of epochs than previous models.

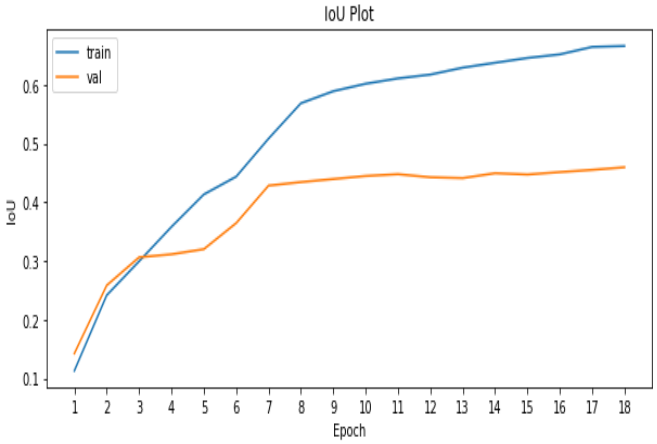
The learning rate for the model is 1e-3, weight decay of 1e-3, we used the Adam optimizer, and we were able to train for 18 epochs. We are optimizing categorical cross entropy loss. We selected the Adam optimizer after reviewing two papers exploring optimizers for semantic segmentation (Yaqub et al. and Parmar et al.).

The metric we are using to assess our performance is the mean Jaccard index, or the mean Intersection over Union, or mIoU. We selected the mIoU, as it gives a clear representation of how the model result mask performs with respect to the ground truth labels.

V. Results / Discussion

We selected the categorical cross entropy as our preferred loss function, after reading Jadon's "A survey of loss functions for semantic segmentation". However, we would still like to undergo a more thorough investigation of which cost function would perform best for our specific model. For our evaluation metric, we used the mean Jaccard index, also known as the mean IoU, or mIoU. As it takes around 3 to 4 hours per epoch to train the model, we decided to use only 3 epochs for our preliminary model. Still, the results were rather promising. Below are tables and graphs of our categorical cross entropy and our mIoU over our three epochs.

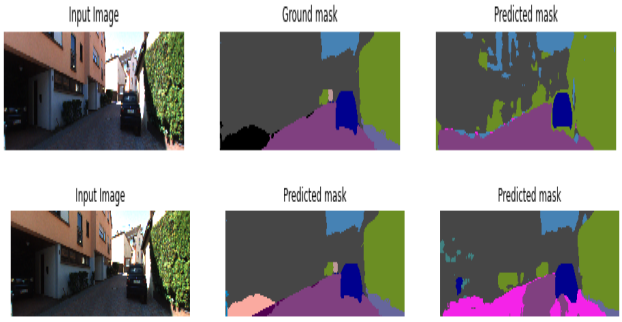




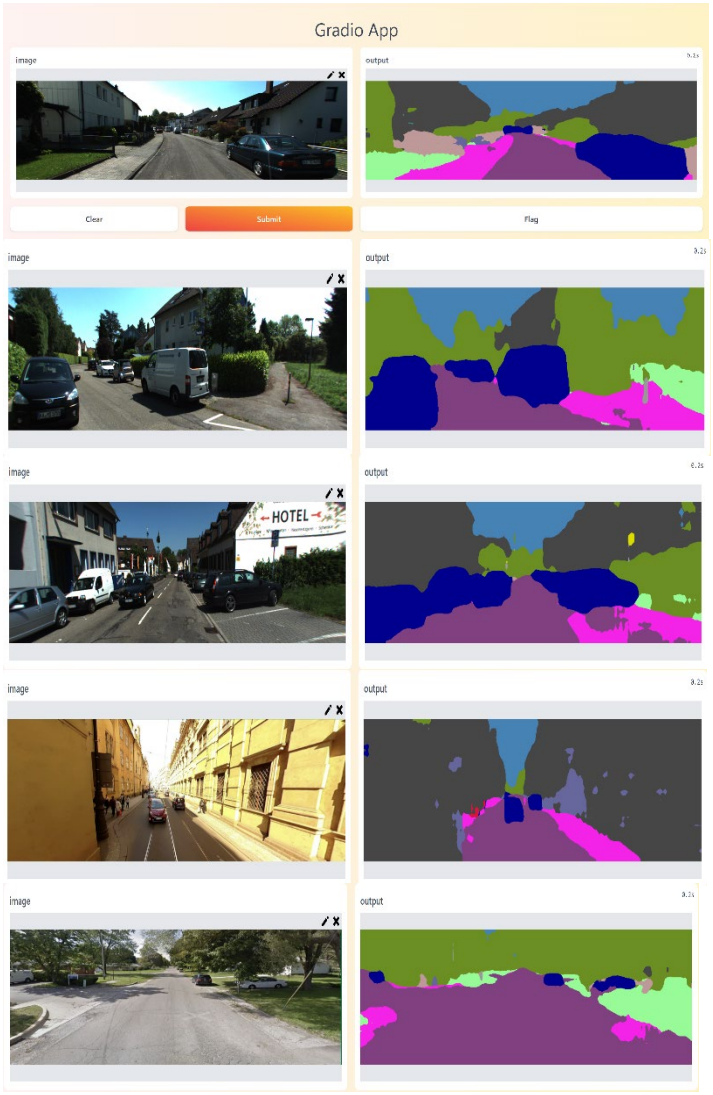
From the loss plot we can see that there is a relatively large gap between the validation loss and the training loss which means that the model is likely not generalizing to non-training data too well. This is a similar occurrence when the IoU plot is observed as we can see that the difference between the plateau of the training data and the validation data is that IoU has a difference of about .2.

Epoch	Train CCE	Val CCE	Train IoU	Val IoU
1	3.156	1.8930	0.1132	0.1426
9	0.4198	0.5450	0.5690	0.4395
18	0.1443	0.3987	0.6660	0.4595

Our model performed much better after more training. The first image is a model after just 3 epochs, and the second image is after 18 epochs. Unsurprisingly, our model performed better after more training.



Additionally, we ran some test data as well as data from the internet through our Gradio app and were able to get satisfactory results. The first three images are from the test dataset, and the fourth and fifth are from Google Maps.



The three results from our test data are very promising, with some misclassifications occurring semi-frequently, however, most vehicles, sky, and roads are segmented well by the model. Additionally, there is some room for improvement with sparse classes, such as the traffic sign. The results from using an online image from Google Maps in Austria, Vienna and Champaign, Illinois respectively showed that the model can generalize, but not too well. For example, it can identify cars, sky, road, and sidewalks, but has trouble recognizing people and vegetation.

VI. Conclusion and Future Work

Overall, we are very satisfied with the results of our model. While we do feel the model could perform much better with more training, our model is in a great place for only 18 epochs. Although we are a little disappointed about the IoU starting to level off, we feel that future tuning as well as more training would break through this plateau.

As a group, one of the more costly lessons we learned was the importance of checking if transformations occur. We had to throw out a model after realizing it was flipping the train input,

and not the mask data, meaning that the inputs and masks did not match up properly. Moreover, we also discovered that our model was getting saturated, and had to switch to a deeper Deeplabv3 model. This was for the best, as it provided us with better results over fewer epochs. We also experimented with cropping to reduce image size without mixing information, in order to provide the model with less data to make training faster while still providing good results.

In the future, we would like to not only spend more time training the model, but also applying transforms to further generalize the model. Additionally, it would be nice to train the model on uncropped images to improve performance with sparse classes. It would also be interesting to train the model on a higher ResNet model, such as ResNet 154, and explore different loss functions, such as the Dice loss to see if we could improve results further.

Our code is available on GitHub at https://github.com/mgarbvs/430-07-torch-enjoyers.

REFERENCES

[1] Liao, Yiyi, Jun Xie, and Andreas Geiger. "KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d." *arXiv preprint arXiv:2109.13410* (2021).

[2] Peng, Chengli, and Jiayi Ma. "Semantic segmentation using stride spatial pyramid pooling and dual attention decoder." *Pattern Recognition* 107 (2020): 107498.

[3] Kirillov, Alexander, et al. "Pointrend: Image segmentation as rendering." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020.

[4] Jadon, Shruti. "A survey of loss functions for semantic segmentation." *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, 2020.

[5] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[6] Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017): 834-848.

[7] Chen, Liang-Chieh, et al. "Rethinking atrous convolution for semantic image segmentation." *arXiv preprint arXiv:1706.05587* (2017).

[8] Liu, Chenxi, et al. "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019.

[9] Yaqub, Muhammad et al. “State-of-the-Art CNN Optimizer for Brain Tumor Segmentation in Magnetic Resonance Images.” Brain sciences vol. 10,7 427. 3 Jul. 2020, doi:10.3390/brainsci10070427

[10] Parmar, Vivek, et al. "Exploration of optimized semantic segmentation architectures for edge-deployment on drones." arXiv preprint arXiv:2007.02839 (2020).

Both Michael and Petar worked on modelling the data and typing up the report. Modelling was done collaboratively and then combined into one notebook. Each team member did approximately 50% of the work on each topic.