Semantic segmentation of off-road images using transfer learning and DeepLabv3+

Nathaniel M Haddad
Khoury College of Computer Sciences
Northeastern University
Boston, MA, United States
haddad.na@northeastern.edu

Amit Vijaykumar Mulay
Khoury College of Computer Sciences
Northeastern University
Boston, MA, United States
mulay.am@northeastern.edu

***Abstract*—Many state-of-the-art deep learning algorithms require both a large training dataset and compute power, which for a variety of reasons is not always available to the user. Training large networks from scratch becomes tedious, and time consuming for the end-user. Transfer learning is a machine learning method where knowledge is transferred from one domain to another, which ultimately eliminates the need to train from scratch. In this paper, we will examine the effect of transfer learning on large encoder-decoder style deep neural networks applied to the task of semantic segmentation. DeepLabv3+ is one such architecture, combining Atrous Spatial Pyramid Pooling modules into an Encoder-Decoder network to extract both rich contextual information, and create sharp object boundaries. By applying depth-wise separable convolution to altrous spatial pyramid pooling and decoder modules, DeepLabv3+ was also able to successfully incorporate techniques from the 2016 Xception model, resulting in faster, more powerful, and larger networks. We propose extending the use of a pre-trained DeepLabv3+ model to the challenging task of off-road perception. Utilizing the newly available Yamaha-CMU Off-Road Dataset, we successfully employ transfer learning techniques to a pre-trained model for the task of semantic segmentation of off-road images.**

***Index Terms*—Computer Vision, Neural Networks, Transfer Learning**

I. Introduction

The last decade has seen extraordinary growth in the autonomous vehicles sector. With the incorporation of a variety of new techniques in deep learning, as well as modern sensor capabilities, autonomous vehicles are quickly approaching Society of Autonomous Engineers (SAE) Level 4. Semantic segmentation has proven to be a useful tool in this process.

Semantic segmentation is a task in computer vision that refers to the process of assigning each pixel in an image or multi-dimensional scene to a specific class. For example, in two-dimensional image of a city scene, every pixel in the image would be assigned to a specific class, such as person, building, sidewalk, car, bus, sky, etc. Compared to the other vision techniques such as object detection, semantic segmentation focuses on the assignment of every pixel in the image to a class, versus simply segmenting a single or numerous objects from a scene.

In semantic segmentation, pixel level classification provides more informative output such as an object's shape and boundary in relation to the entire scene. Semantic segmentation also plays a critical role in the three-dimensional space. For example, semantic segmentation may be applied to a pointcloud created by a LiDAR sensor to assist with road segmentation, object detection and tracking, lane marking segmentation, and a variety of other tasks. These specific categories become critical to autonomous vehicle perception, allow the vehicle to identify its surroundings, such as the road, other cars, people, obstacles, and more. In this scenario, the end goal is to create information to assist in building a geometric structure of the world so that the agent, or ego-vehicle, can navigate the scene [3].

However, while a number of companies and organizations are making huge strides in the operation of autonomous vehicles in urban and on-road environments, the task of learning to navigate off-road scenarios remains a challenging one. It requires the agent to be able to navigate a variety of different terrains, some which may appear to be impassible when trained on traditional datasets of more urban environments. Many of the tasks mentioned above with regard to LiDAR pointcloud processing become less powerful, as leafy paths, dirt roads, and rocky trails do not contain lane markings to help guide the vehicle. The shape of the ”road” may also be different, as a variety of terrains aside from traditional asphalt roads become traversable. Here, utilizing new off-road data and an model previously trained on a similar task such as semantic segmentation becomes useful.

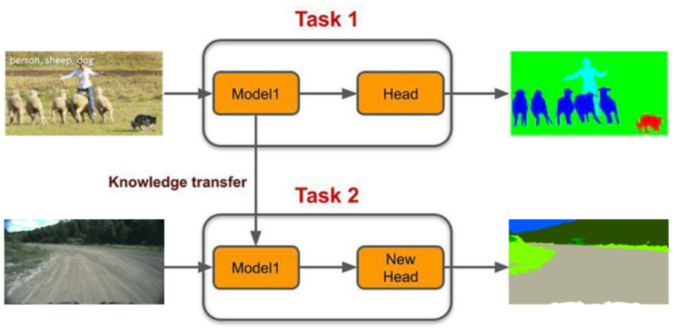

Fig. 1. Transfer learning: training a network pre-trained on MS COCO with the Yamaha-CMU Off-Road Dataset

Outside of autonomous vehicle companies and auto manufacturers, access to autonomous vehicle semantic segmentation models pre-trained on off-road data is limited. However, with transfer learning, it becomes easier for a researcher at a smaller institution to utilize some of the most exciting and newest architectures for this tasks by simple using a model pre-trained on the task of semantic segmentation and fine-tuning it to an off-road dataset.

II. Related Work

As mentioned earlier, Transfer Learning [4] is a way of utilizing the knowledge of a trained model on a dataset in one domain to solve the problem at hand in another domain. Convolutional Neural Networks (CNNs) trained on one domain (source domain), may learn some generic features that are relevant to the other domain (target domain). Therefore, the network with learned features in the source domain could be used as a baseline to solve similar problem in target domain with less training required to achieve desired result in target domain. Pan et al. [6] categorize the transfer learning methods as inductive transfer learning, transductive transfer learning, and unsupervised transfer learning. In deep learning, they are also represented as the fine-tuning approach, the feature extraction approach, as well as multi-task learning and meta-learning. In the fine-tuning approach, the CNN learns general features through lower layers and specific features through higher layers. For the purposes of transferring the general features knowledge learned by lower layers, these layers are frozen and only few higher layers are modified with the dataset in target domain. In the feature extraction approach, the most important features from source domain, those that might better represent the features in target domain, are extracted and the model is retrained with the most important features interleaved with the target dataset. In the multi-task learning approach, a model is trained on multiple source tasks to increase the generalization of the network and then fine-tuned with the target domain. Finally, the Meta-Learning technique helps a model to learn about what to learn in order to make sure the knowledge is best fitted for the target domain. In this work, we are using the fine-tuning approach.

A large amount of work exists with respect to semantic segmentation of two-dimensional images, the most significant and relevant to our work being models based on Fully Convolutional Networks (FCNs), encoder-decoder architectures, and in particular, DeepLabv3+ [1]. We were heavily inspired by Chen et al [1], as we use their work as the foundation for ours. We also utilized the dataset created by Maturana et al [3], which provides the other central piece of our work. Our implementation also utilizes a variety of resources, and is one of the first publicly available multi-class semantic segmentation systems in PyTorch that utilizes DeepLabv3+ and the Yamaha-CMU Off-Road Dataset.

With regard to off-road datasets, existing work has been done to make off-road datasets publicly available for use in bench-marking state-of-the-art models. This includes DeepScene's Freiburg Forest dataset. Freiburg Forest is similar to the Yamaha-CMU Off-Road Dataset which we selected for use in this paper. While the Freiburg Forest dataset contains a significant amount of images more than the Yamaha-CMU dataset, it should be noted that it contains less classes (six versus eight, respectively). One of the arguments Maturana et al [3] makes for the Yamaha-CMU Off-Road Dataset is more diverse and challenging than DeepScene, albeit both datasets are small.

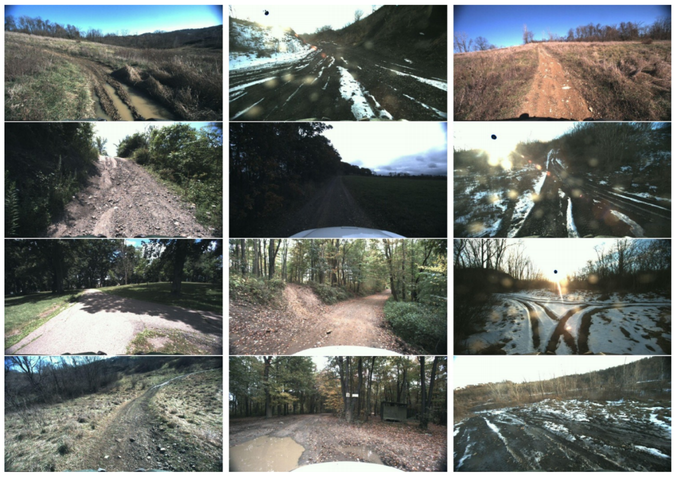
Fig. 2. Example images from the Yamaha-CMU Off-Road Dataset

III. Methods

Our approach to the task of semantic segmentation of off-road data using a pre-trained architecture followed a simple procedure. We first examined a variety of different architectures in the process of selecting one most relevant to our problem with publicly available pre-tained weights from the torchvision package. We found a variety of different architectures to use, but decided to utilize the DeepLabv3+ architecture included in torchvision. We will discuss the architecture in detail in a later section. Using a pre-trained network, our goal was to first evaluate performance of the pre-existing network on off-road images from the Yamaha-CMU Off-Road Dataset, retrain the model on the off-road images, then evaluate performance on the newly trained model. This process was repeated numerous times across three different architectures with auxiliary outputs also applied during training for each.

A. Dataset

The Yamaha-CMU Off-Road dataset from Maturana et al [3] is a new publicly available dataset released in February 2021. It contains 1076 different images of height 544 pixels by width 1024 pixels across three seasons with locations in Western Pennsylvania and Ohio. Images were collected from a sensor attached to an autonomous all-terrain-vehicle (ATV). Labels were created from these images using a polygon-based interface. The eight classes are as follows: sky, rough trail, smooth trail, traversable grass, high vegetation, non-traversable, low vegetation, obstacle. A ninth class is used to label the ego-vehicle, if it appears in the image. More importantly labels were post-processed to increase the density the

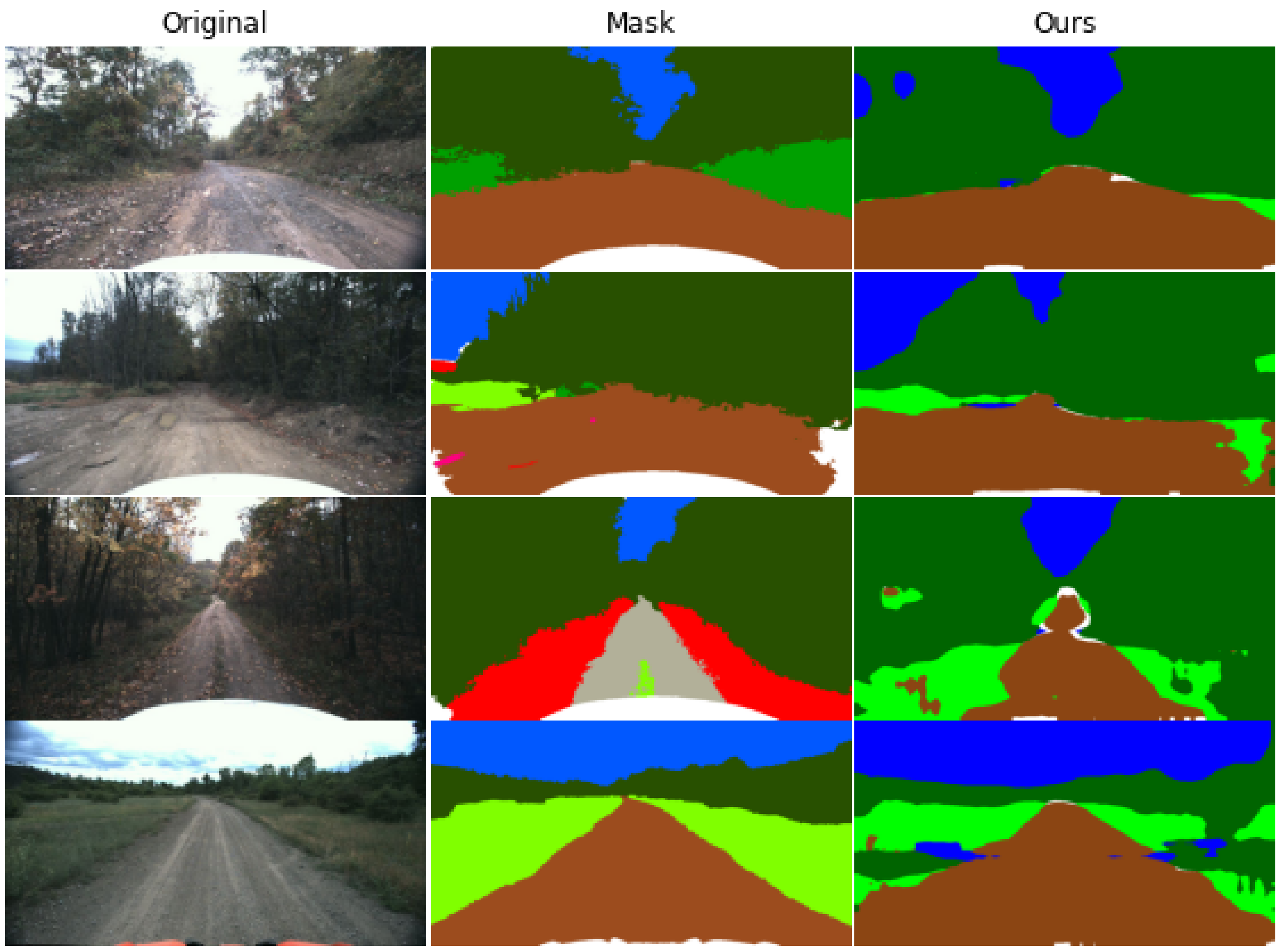

Fig. 3. Example output from MobileNetV3-Large after 10 epochs of training

labels and then manually inspected to ensure the correctness of the labels. The dataset was then randomly split into 931 training images and 145 validation images. Working through this lesser known dataset proved challenging, but rewarding. One important note to make about this dataset is that some of the labels from image to image appear very similar, and it is sometimes confusing to the end-user. Be aware that this is the first release of this dataset.

B. Architecture

We will now describe our procedure for setting up the experiments in terms of architectures. Torchvision's DeepLabv3+ modules include a variety of backbone architectures (convolutional stacks) that create a model zoo environment for users. When initializing the model, make sure that the flag to include pre-trained weights is set to true. We also experimented with all of the available architectures included with the torchvision package. This includes a ResNet50, a ResNet101, and a MobileNetV3-Large. All are trained with auxiliary loss. To replicate our best work, please use a ResNet101 as the model backbone. We had most success with this version.

Overall, we observed that each backbone had its own strengths and weaknesses. In the demo notebook provided with the repository attached to this paper, we allow users to experiment with training each of the three backbone architectures mentioned above.

C. Evaluation

We wrote a custom training loop that was required for handling our loss function (Cross Entropy) and our actual and predicted labels, which resulted in eight channel images, one channel per class. The argmax of the predicted label produces the output mask.

For evaluation we used also mean intersection over union (mIoU). mIoU is a great choice for semantic segmentation tasks, and has proven to be one of the most popular metrics in assessing the performance of state-of-the-art models on benchmark datasets such as COCO and CityScapes. However, with the limited time allotted for this project (about two weeks), we still believe it would be worth considering mean average precision (mAP) over mIoU, or using both, to assess

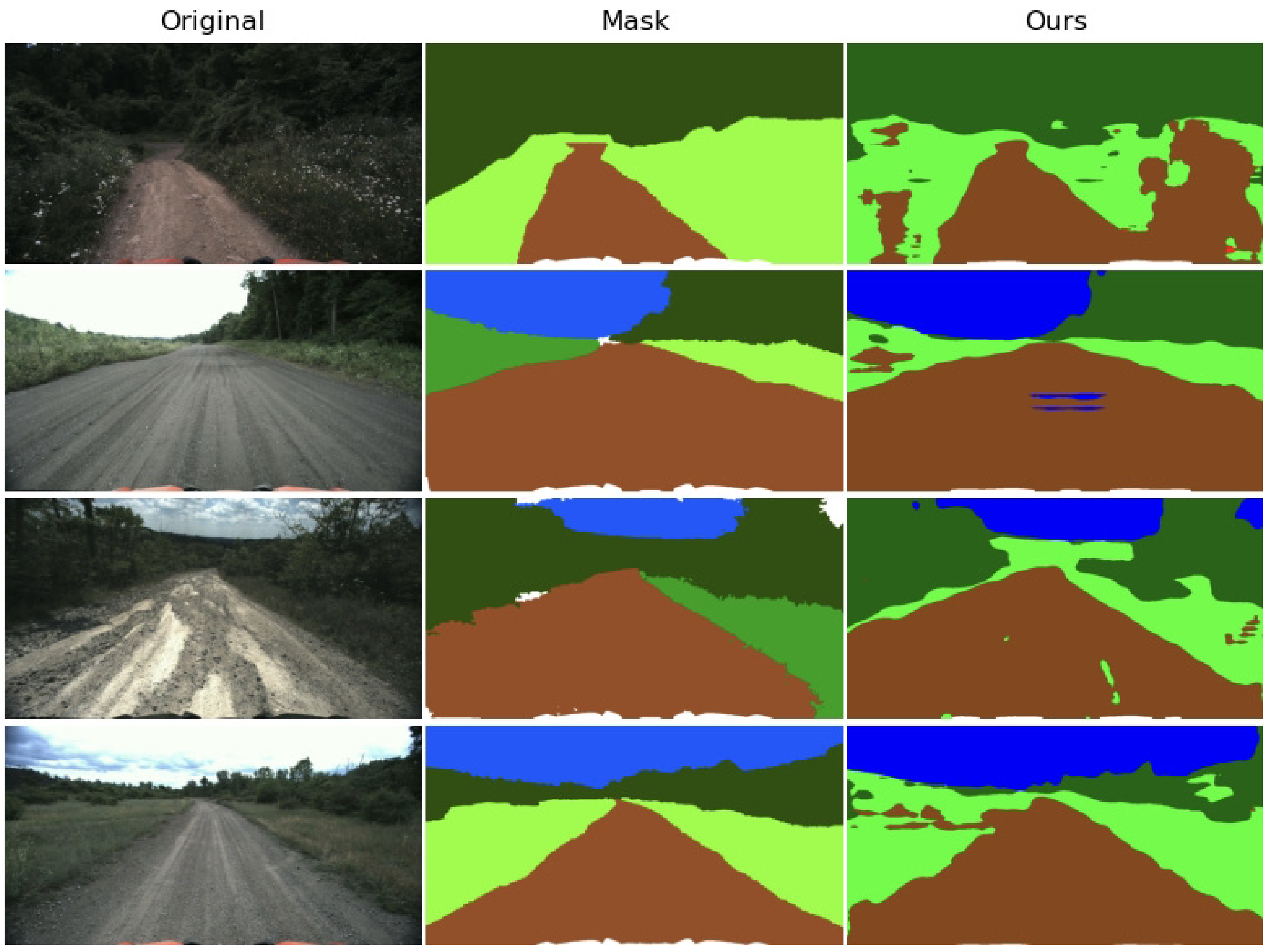

Fig. 4. Example output from ResNet50 after 50 epochs of training

the performance of our system. It would also have been great to create visualizations of both metrics to compare as well.

IV. Experiments and Results

Following the procedure described above, we first created the segmentation system using the torchvision package and instead of using randomly initialized weights, we used pre-trained weights. These weights were trained on the COCO val2017 dataset. After creating a new output and last hidden layer, we examined the output of the networks on an image from the Yamaha-CMU dataset. The model failed to predict the output classes correctly, instead outputting the mask as a single class (in our case, the model predicted the entire image to be the sky). We did not include images or results of this preliminary experiment as it is trivial.

In the following sections, we will describe the results of our experimentation, which was completed on both local and cluster machines. To simplify, we have broken the experiments on both setups into different sections, as to clarify any questions that may arise concerning particular batch sizes, image sizes, and training epochs.

A. *Local experimentation*

Part of our experimentation was split between local and cluster based training. We had access to an NVIDIA Quadro P2200 5GB GPU, with which we were able to train a number of different models. This was our development setup and was used to test and create our segmentation system. On average, the single GPU setup took about eight minutes per epoch using small batch sizes (two images of size 256 by 256) utilizing a ResNet101 architecture. Depending on the backbone architecture, we were also able to increase the batch size, the speed of each training iteration, and overall performance. Since this was a personal computer, we did not use it for longer training sessions, so training here was limited to 10 epochs.

It is important to note that the smaller batch sizes are due in part to the 5GB limit of the P2200. Models trained on this machine ranged from one to five epochs. Results were still significant, and were also used to justify running the segmentation system on a cluster of GPUs. With access to better resources, it should be easier to train larger models with

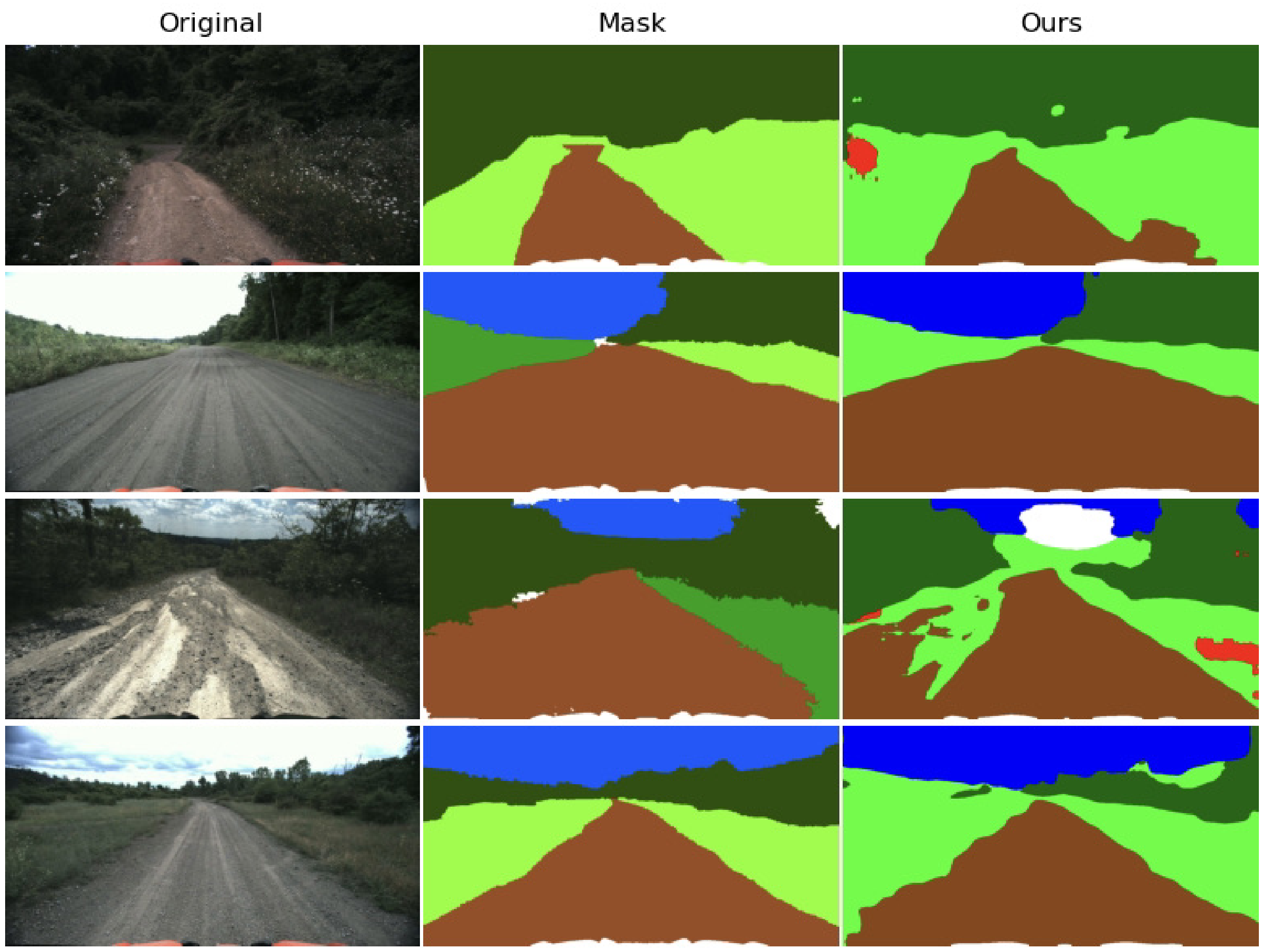

Fig. 5. Example output from ResNet101 after 50 epochs of training

larger batch sizes.

B. Cluster experimentation

We had access to Discovery High Performance Cluster, where we could train models with larger batch size of eight using NVIDIA Tesla P100 480 GB GPU. For cluster experimentation, we used a ResNet50 and ResNet101 as the backbone architecture of our DeepLabv3+ model, as we did locally. We trained several iterations of models here. Quality of the output increased significantly with the number of epochs, as seen in figure 4, which was trained on the above model over 50 epochs. We could run simultaneous experiments changing different parameters of the model having access to high performance computing node. This sped up the experimentation process. Table 1 shows the experimental results and quantitative analysis.

C. ResNet50

Using a ResNet50 pre-trained on COCO val2017 as the model backbone, we achieved slightly poorer performance than the ResNet101 backbone that we will discuss in detail in the next section. However, we were able to train larger batch sizes and training time per epoch significantly decreased (about half the amount of time per epoch).

D. ResNet101

For our initial experimentation, we selected a DeepLabv3+ architecture with a ResNet101 backbone pre-trained on Microsoft's COCO val2017 dataset. COCO itself is an interesting choice for this specific transfer learning tasks. In COCO, masks are learned of objects, while with the Yamaha-CMU Off-Road Dataset, the model is learning to differentiate between six different classes of terrain, the sky, and obstacles. Nonetheless, performance after using COCO as the pre-trained weights for transfer learning was great, as shown in the results of our experimentation as seen in figure 4. This setup gave us the best performance by far.

E. MobileNetV3-Large

We also trained a model with a MobileNetV3-Large pre-trained on the COCO val2017 dataset using auxiliary heads during training with a batch size of sixteen over ten epochs.

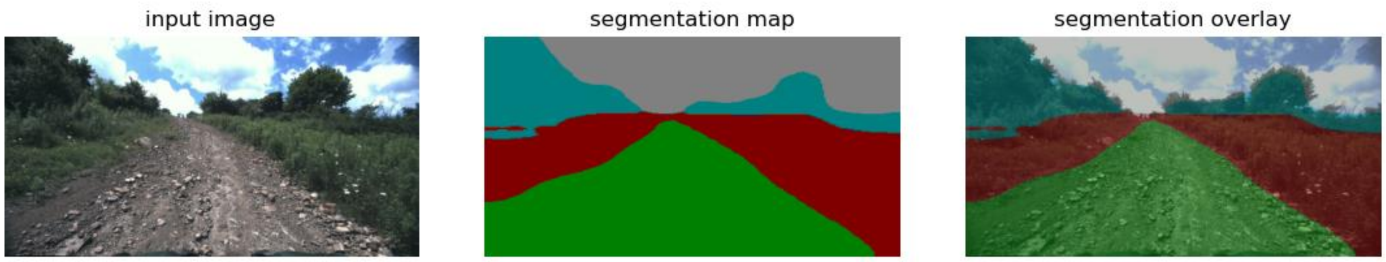

Fig. 6. Example of output from our model trained on the Yamaha-CMU Off-Road Dataset after 50 epochs

TABLE I
EXPERIMENTS WITH DEEPLABV3+ NETWORK TRAINED ON YAMAHA-CMU OFF-ROAD DATASET

Sr. No.	***Pre-trained Networks***	***Training time HH:MM:SS***	***Batch size***	***Number of Epochs***	***Training Loss***	***Validation Loss***	***Best mIoU (mean Intersection over Union)***
1	MobileNetV3-Large	01:02:01	8	50	0.9372	1.1837	85.53%
2	ResNet50	03:09:18	8	51	0.0920	1.3496	0.8481
3	ResNet101	04:12:56	8	50	0.0873	1.2902	0.8537

This model achieved poorer performance, but was much easier to train on a smaller GPU or cluster and enabled larger batch sizes.

We included a grid of images above produced as output of the MobileNetV3-Large experimentation in figure 3. Note that in the images we selected for the figure, we included the third output as an example of strangely labeled data. The red indicates an obstacle which is clearly an error of labelling. Still, our model was able to predict the worn path, surrounding grass, and tall trees. However, this discovery late in our experimentation that the training data may be incorrect was a surprise, but not totally unexpected, as this is a new dataset. Nonetheless, incorrect labelling extremely dangerous for scenarios where autonomous vehicle perception systems may be trained on it.

After ten epochs, we were able to correctly label pixels into the eight classes with a mIoU value of 0.2231 across the validation set. while this is far from a great score, it does demonstrate that the model is capable of learning to differentiate between the classes. With more resources, such as larger GPUs to increase the batch size, and more epochs, we could surely attain a higher mIoU. However, the point of this exercise is to evaluate the performance of MobileNetV3-Large as a backbone in a DeepLabv3+ architecture, relative to the other available convolutional stacks.

F. Auxiliary Towers

As mentioned above, we also used auxiliary outputs during training of all of the architectures mentioned above. As mentioned previously, DeepLabv3+ took advantage of prior work in this technique, inspired heavily by the Xcption and Inception architectures. This required creating a customized training loop to combine earlier auxiliary outputs with the loss during training. These auxiliary outputs were not used during validation.

Our findings were that the use of auxiliary outputs slightly increased mIoU, but not significantly. We believe this to be due to the fact that vanishing gradient is already remedied by residual modules. We decided not to include metrics in this report, as our findings are trivial and not significant.

V. DISCUSSION AND SUMMARY

Our findings show that with limited exposure to training data, our semantic segmentation transfer learning system can successfully segment off-road images into eight classes. Across 50 epochs, we see significant improvement, on par with Maturana et al [3]. However, we did notice a variety of problems stemming from the Yamaha-CMU Off-Road Dataset including dataset size, inaccurate labels (see training pair 0 and 477), and the inclusion of the ego-vehicle in some of the images as a ninth phantom class. Future work should be done to improve the quality of the masks in this dataset, as well as the quantity of training and validation images. This is extremely troubling, as this dataset was used to train a perception system of an off-road autonomous vehicle. In the future, we strongly recommend the correction of the aforementioned labelling errors before proceeding with deployment of a system trained on this dataset in a real-world scenario.

As for the output of our segmentation system, future work can be used for lane generation using the boundaries of classes identified as traversable. Lane generation is a critical step in autonomous vehicle navigation and is a logical choice. From the left and the right lane, we may also be able to generate a polyline that may be used to steer the vehicle in the most predictable path of safely traversing the terrain.

Sharma et al. [7], in their work, proposed a new light-weight framework for transfer learning for Off-road Autonomous Driving. They recommend to find appropriate network size for the task at hand. For the off-road Autonomous Driving Dataset segmentation task, they proposed a lighter network for transfer learning rather than using original base architecture as a whole. They experimented with different network sizes to find out the best performing network size for the target domain. In our experiments, we relied on using the base DeepLabv3+ model network architecture as whole. In future, it would be interesting to see the performance of the model when experimented with different network sizes.

As mentioned in previous sections, the dataset we used is relatively smaller than most of the benchmark datasets used for the task of semantic segmentation. It would be interesting to see the results of using combination of original dataset and synthetic data generated for the same domain using GANs synthetic data generation techniques [8].

VI. Acknowledgment

We would like to thank Bruce Maxwell of Northeastern University for supporting us throughout this project and over the semester with throughtful guidance and inspiration. We would also like to than Chen et al [1] and the DeepLab team at Google for creating the architectures we used during this project and for making their work publicly availble. Finally, this work would not be possible without Maturana et al [3] for creating the off-road dataset we used for our project.

References

[1] Chen, Liang-Chieh, Zhu, Yukun, Papandreou, George, Schroff, Florian, and Adam, Hartwig. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation." Computer Vision – ECCV 2018 (2018): 833-51. Web.

[2] Chollet, Francois. "Xception: Deep Learning with Depthwise Separable Convolutions." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017): 1800-807. Web.

[3] Daniel Maturana and Po-Wei Chou and Masashi Uenoyama and Sebastian Scherer, "Real-time Semantic Mapping for Autonomous Off-Road Navigation" in Maturana-2017-102768, September 2017, pp. 335 - 350.

[4] Stevo. Bozinovski and Ante Fulgosi (1976). "The influence of pattern similarity and transfer learning upon the training of a base perceptron B2." (original in Croatian) Proceedings of Symposium Informatica 3-121-5, Bled.

[5] Stevo Bozinovski (2020) "Reminder of the first paper on transfer learning in neural networks, 1976". Informatica 44: 291–302.

[6] Pan, S.J.; Yang, Q. A survey on transfer learning. IEEE Trans. Knowl. Data Eng. 2010, 22, 1345–1359.

[7] Sharma, Suvash, Ball, John E, Tang, Bo, Carruth, Daniel W, Doude, Matthew, and Islam, Muhammad Aminul. "Semantic Segmentation with Transfer Learning for Off-Road Autonomous Driving." Sensors (Basel, Switzerland) 19.11 (2019): 2577. Web.

[8] Neff, T. et al. "Generative Adversarial Networks to Synthetically Augment Data for Deep Learning based Image Segmentation." (2018).