Semantic Segmentation of Street Scenes

Amey Kulkarni[1], Ajith Kumar Ganesan Govindasamy[1], and Dan Sullivan[1]

***Abstract*— Semantic segmentation is of vital importance in autonomous vehicles for scene understanding and mapping the environment. Making dense predictions for classifying each pixel of the scene into different classes is computationally expensive and multiple deep learning frameworks have emerged for tackling this problem. To develop our understanding of this modern challenge, we replicate, deploy, and tune a state-of-the-art semantic segmentation fully convolutional network. While we struggled to achieve consistent performance with our cloud-based server implementation, we achieved our goal of familiarizing ourselves with Semantic Segmentation.**

I. INTRODUCTION

Semantic segmentation is the process of interpreting a visual scene and finding meaning in its constituent parts. As demonstrated in figure 1, semantic segmentation is actually a development of classifying and image and determining that object's location in the image. Segmenting the object takes the localization task further and assigns each pixel in the image to its predicted object class.

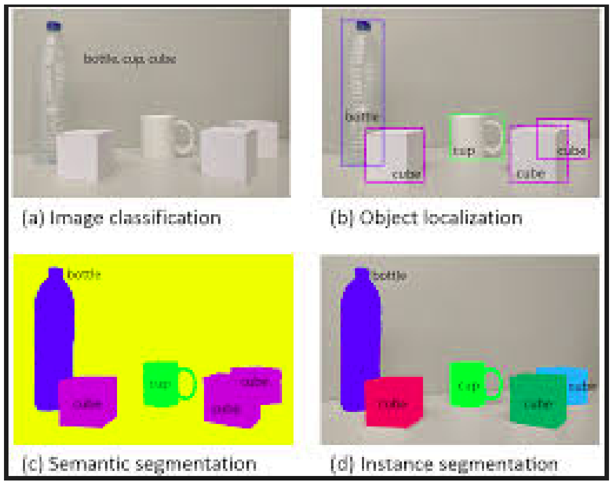


Fig. 1: The different tasks required to process a visual scene [1]

Semantic segmentation allows a machine to understand a visual scene. Recent advances in this have allowed real time processing of images and videos to differentiate subjects from the background and track parts of a person's face [2]. Takeki et. al. created an algorithm to recognize small birds in photos to help plan the location of wind energy plants [3]. Semantic Segmentation has been applied to aerial images to classify roads, tree cover, and buildings [4] [5]. The technique is also being used to automate processing of medical imagery [6]. One of the most challenging applications, and the focus of this paper, is that of autonomous vehicles, particularly those navigating busy and cluttered urban areas. This application cares about rapidly identifying and locating objects like pedestrians, street lights, and other vehicles as demonstrated in figure 2.

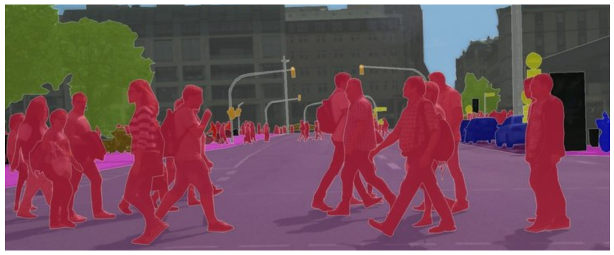


Fig. 2: An example urban scene (CityScapes dataset) [7]

While the task of distinguishing objects among a scene is trivial for humans, it is a complex task to implement for a computer. Deep learning networks have made strides with the task, but even the most advanced algorithms are limited in the number of classes they can recognize. Even if the number of classes is limited, these networks require many resources and time to train. As such, many algorithms are adaptations and additions to existing algorithms, like VGG and ResNet. The process of adapting these pre-trained classification networks into convolutional networks is called transfer learning [1]. Even if a network achieves high levels of accuracy with classes relevant to an application like autonomous vehicles, the network must be capable of running fast enough to be relevant to a vehicle attempting to navigate the environment. Thus, given the challenges associated with the task and its importance to autonomous vehicles, semantic segmentation continues to be a topic of great interest and development in the academic and professional community.

The remainder of this paper is organized as follows. First, we provide a literature review of our research of deep learning methods applied to the semantic segmentation task. We then describe our efforts and lessons learned from replicating an existing network's architecture and training it on the CityScapes dataset. We then experiment with and tune this network, again describing the lessons learned and results gained. Finally, we present our conclusions for the project and describe areas deserving future work.

II. LITERATURE REVIEW

A. Datasets

One of the most important ingredients for a successful deep learning algorithm is data in high enough quantity and

[1]J. Kulkarni, A. Govindasamy, and D. Sullivan are graduate students of Worcester Polytechnic Institute, Worcester, MA 77005 USA.

quality. The PASCAL Visual Object Classes (VOC) challenge is another popular dataset used to benchmark different algorithms. The challenge has been organized annually since 2005, increasing its labeling of thousands of images from four classes in 2005 to ten in 2006, and twenty in 2007 [8]. Similar datasets include PASCAL Context which classified areas the image into three high-level subjects and PASCAL Part which decomposed objects into their constituent parts (e.g. the wheel of a car) [1]. Microsoft's Common Objects in Context, or MS COCO, is another challenge dataset with tens of thousands of training, validation, and test images and 91 classes [9].

The aforementioned datasets focus on a variety of everyday objects. Autonomous road vehicles don't care so much about recognizing cats or chairs. Thus, many datasets targeted specifically to driving and urban scenes have been created. Brostow et. al. created the Cambridge Driving Labeled Dataset, or CamVid, in 2008. Consisting of video images mounted on a car, it has per-pixel labeling on about 700 images [10]. The more recent Cityscapes dataset provides annotated images from 50 different cities. In all, it has 5,000 images with pixel-level annotations and a further 20,000 images with coarse annotations [7]. The SYNTHIA dataset is interesting in that it's a collection of images with pixel mapping that were created in a virtual environment, thus guaranteeing 100% accuracy of the ground truth images [11].

These datasets are a small fraction of the many that are available. Garcia-Garcia et. al. provide an excellent summary of many of them [1].

B. Networks

1) Early Semantic Segmentation with Deep Learning Techniques: In 2010, Zhang et. al. implemented an architecture on the CamVid dataset that used its stereo imagery to produce depth maps of the scene [12]. This approach would be surpassed by 2D methods. Arbelaez et. al. produced a segmentation network with a bottom-up approach. The image is first broken into regions and then object candidates to which classification is then applied. After training on PASCAL, the network achieved roughly 40% success on VOC2010 and VOC2011 [13]. In the same year, the benefits of second-order pooling of CNNs were demonstrated by Carriera et. al. [14]. Girshick et. al. demonstrated a method for counteracting the scarcity of labeled data for training their network, supervising their pre-training to produce substantial performance gains on the PASCAL VOC 2012 results [15]. Tian et. al. demonstrated how an algorithm could learn from multiple datasets to overcome deficiencies in the labeled data such as blurriness and uncommon pedestrian poses [16].

2) Instance and Scale-Aware Segmentation: Though less significant to our focus in this paper, two follow on challenges to "vanilla" semantic segmentation are instance and scale-aware segmentation.

Instance segmentation will identify each time an object from a particular class appears in an image. Dai et. al. developed an architecture with three different network structures tasked with differentiation, mask estimation, and categorization to address this problem [17]. In 2016, Li et. al. accomplished the task with a Region Proposal Network (RPN) and CNNs [18].

Scale-aware segmentation networks attempt to improve a network's ability to understand iterations of the same type of object that appear as different sizes across input images. In 2015, Yu et. al. employed diluted convolutions to preserve multiscale contextual information without losing resolution [19]. In 2016, Chen et. al. fed copies of training images into their model with different scales to improve their network's ability to understand scaling [20]. Pan et. al. improved the robustness of the DeepLab network by training a full-scale feature network that enables detection of large objects dominating most of the input image [21].

3) VGG-16 Basis: As mentioned before, many segmentation networks build upon successful classification networks, such as VGG-16. Noh et. al. layered a deconvolution network atop VGG-16, consisting of deconvolution, unpooling, and ReLU layers, significantly helping with scale and instance aware performance [22]. While Long et. al. also looked at GoogLeNet and AlexNet, their application of CNNs to VGG-16 with a skip level architecture successfully combined coarse and fine information to simultaneously train for location and semantic tasks [23].

4) ResNet Basis - DeepLab: As mentioned before, many segmentation networks build upon successful classification networks, such as ResNet. One of the most cited networks implemented using ResNet as a base is DeepLab. In 2014, this network was improved by adding a fully connected CRF to the end of the CNN to smooth noisy results [24]. In 2016, the DeepLab team added dilated (aka atrous) convolutions to the network with fully connected CRFs for dense feature extraction from the generalized layers [25]. later, the team experimented with methods of avoiding the computational expense of CRFs by employing a domain transformation where an edge map is used to process the segmentation output of the CNN layers [26]. DeepLabv3 was introduced in 2017 and featured a cascaded layers to better capture multiscale objects [27]. He et. al. built upon DeepLab and produced minor gains with learned dilation factors [28]. Liu et. al. experimented early this year with Neural Architecture Search (NAS), a process of autonomously designing neural network structures; the auto-deeplab network performed comparably to DeepLab v3 [29].

5) Standalone Architectures: While most of the research on applying Deep Learning techniques to semantic segmentation has taken place in the last decade, there is a plethora of unique approaches to the problem.

In 2014, Mostajabi and coauthors implemented a feedforward architecture with zoom-out feature construction to good effect [30]. In 2015, Hong et. al. decoupled the segmentation and classification tasks in semi-supervised implementation [31]. Lin et. al. developed a way of training a deep algorithm in a piecewise fashion, combining CNNs with CRFs

[32]. Gidaris et. al. presented a method of using CNNs to iteratively better localize objects in the image with bounding boxes [33].

SegNet and ENet are two frameworks developed around the same time. SegNet focused on improving the network's mapping of deep, generalized features to the full scale resolution with decoder layers [34]. Conversely, the primary goal of ENet is to run on platforms with minimal resources in real time, such as mobile platforms [35].

Luc et. al. created a Generative Adversarial Network (GAN) to help train the primary segmentation network via binary classification against ground truth images. This is the first instance of a GAN being applied to semantic segmentation, though only minor gains were obtained on the PASCAL VOC2012 dataset [36].

Kampffmeyer et. al. experimented with with patch based, pixel-to-pixel, and pixel with median frequency balancing approaches in their neural network before concluding that combining all models provided the best results [5].

Li et. al. implemented a cascading network that focuses on labeling ”easy” classes first before devoting network resources and convolutions to harder resources, saving computational complexity [37].

Finally, Wang et. al. experimented with learnable upscaling and variable dilation rates to better preserve high-level information in the network's deep generalized layers [38].

III. DEPLOYING A SEMANTIC SEGMENTATION NETWORK

There are multiple neural network architectures which have proven to be really accurate on the IMAGENET challenge for classification and segmentation. Some of these are the ResNet, GoogleNet, VGG-166, and VGG-19. VGG-16([1]) had achieved 92.7% accuracy on the ImageNet data and was one of the most popular models of the ILSVRC-2014 challenge. The popularity and significance of the model in segmentation tasks has encouraged us to implement it for the semantic segmentation task. The proposed network architecture here is an add on to the VGG-16 to make it a Fully Convolutional Neural Network. All the final Fully connected layers of VGG-16 are converted to convolutional layers. The main advantage of CNNs is that they accept data of any dimensions and that gives a lot of flexibility. The convolutional extension to VGG is done by using skip connection layers.

A. Network Architecture

A standard network for segmentation uses a encoder-decoder framework for better accuracy. The VGG-16 network acts as the encoder and the final convolutional layers operate as the decoders. As the name suggests, VGG-16 net has 16 layers. Figure 3 details these layers. There are thirteen convolutional and three dense fully connected layers. 5 max pooling layers are used in the network. The fully connected layers may not carry forward or may average out the finer information collected by the previous layers. These small details are important for pixel wise dense semantic segmentation. Hence, a skip architecture is used which combines information from a deep and a shallow layer to get the finer and high level features. The final classifying layer is discarded and the other two fully connected layers are converted to convolutional layers. The results of the final convolutional layer are upsampled by a filter with stride 32 to get the coarse predictions. These predictions are called FCN-32s.

Figure 4 shows hows skip connections are added. This is the important step where a 1x1 convolutional layer is added on top of the results of pool4 layer and the output stride is 16(which is upsampled later). The final layer results are upsampled and both these predictions are added. This basically takes in both the finer and the coarse details. These results are called FCN-16s. An improvement in this is FCN-8s where the results of FCN-16s are upsampled and fused with the results of pool3 layer.

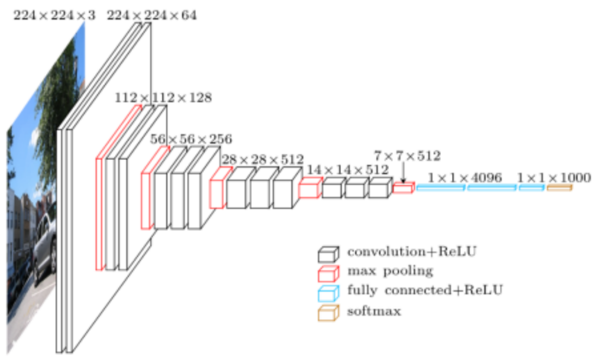


Fig. 3: Convolutionized VGG-16 Architecture [1]

B. Dataset Selection

We h chose the CityScapes dataset as it has a wealth of well-annotated data and is referenced as a baseline in many of our sources [7]. The dataset provides 2,974 RGB training images distributed across 18 European cities. The images are 2,048 x 1,024 in size and are taken from a forward facing camera mounted in a moving car. The appeal of CityScapes as a dataset is that each of these images has a corresponding annotation file with pixel-wise labeling as illustrated in 2.

In addition to training images, the dataset provides three additional cities-worth of validation images. Testing images are also included, but the annotations are only available through the CityScapes webpage, encouraging teams evaluating the dataset to upload and present their results to the public.

C. FCN-8 Implementation

A key tenant of our goal of learning more about Semantic Segmentation was implementing and training a neural network. We accomplished this in three stages: (1) Data Input and Output, (2) Network Structure, Weights, and Training, and (3) Network Evaluation.

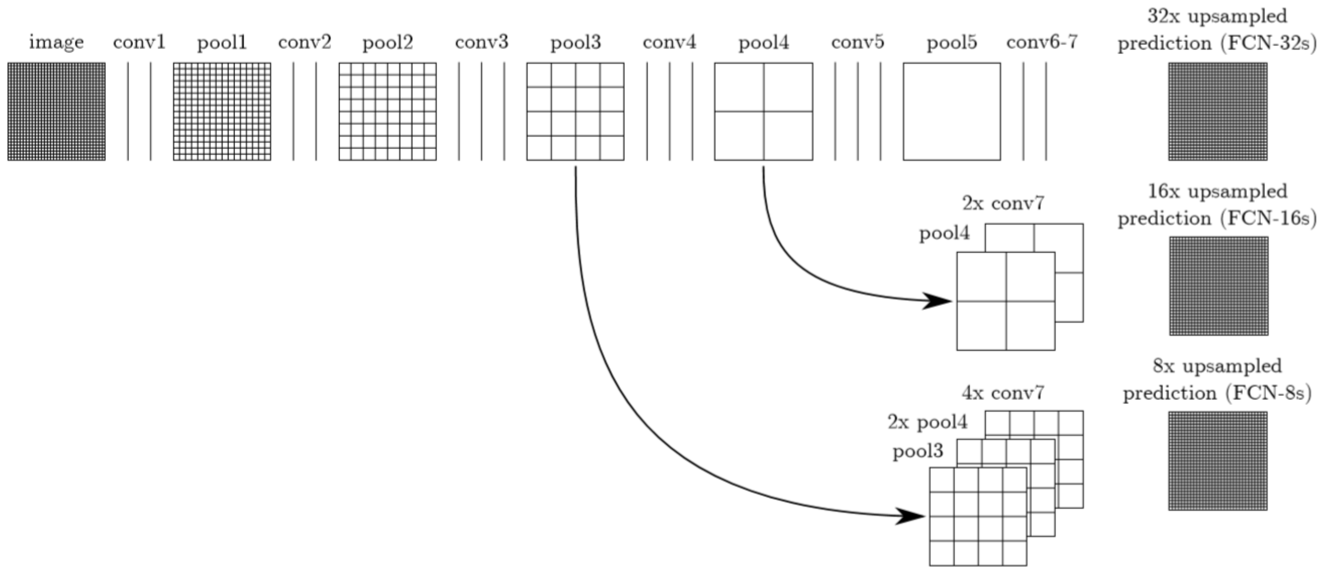


image
conv1
pool1
conv2
pool2
conv3
pool3
conv4
pool4
conv5
pool5
conv6-7
32x upsampled prediction (FCN-32s)
2x conv7
pool4
16x upsampled prediction (FCN-16s)
4x conv7
2x pool4
pool3
8x upsampled prediction (FCN-8s)

Fig. 4: Skip level architecture of Long et. al. with VGG-16 [23]

1) Data I/O: One of the challenges we encountered with this network architecture was that it quickly overwhelmed our local GTX1060 and GTX1070 based desktop machines. Consequently, we moved our training to Google's Colab environment. One of the benefits of the service is that it interfaces with standard Google Drive folders, providing a pleasant user interface and PC-like experience with Jupyter notebooks. Additionally, it provides access to high-powered K80 GPUs with 12 GB of memory.

For dataflow into and out of our network, we loaded all training images and their associated labels into a shared Google Drive folder. Our code gives the user the option of dictating which subset of the 18 cities the network is to be trained on. The training image paths are then read into memory, and a dictionary is created mapping the training images to the corresponding annotated images. Since the testing image annotations weren't available for download, we also read in the validation dataset to use for model testing and performance. Separately, we confirmed an annotated image exists for each of the training and validation images to avoid training interruptions.

Since the images are quite large and our base layer classification network was based on much smaller images, we downsize each image to 25% of it's original size, of 256 x 512 pixels. The annotated images are translated into classes based on pixel color. In all, we evaluated five classes plus the background: cars, people, the road, traffic lights, and vegetation. The training images and classified labels are then converted into arrays for feeding into the network.

The user defines a batch size in the network's TensorFlow training function. At the start of each epoch, the training function shuffles the training image and label paths, processes ”n” images where ”n” is the batch size, and feeds them into the network. This continues until all training images have been fed into the network, completing the epoch.

2) Network Structure, Weights, and Training: To successfully employ transfer learning of a convolutionized classification network (VGG16 in this case), it was important for us to find a source of pretrainined weights. Le's work provided a TensorFlow-based example that we incorporated into our work [39]. It downloads the pretrained VGG16 weights, loads the TensorFlow graph and operations, and builds the additional FCN-8 layers into the graph. TensorFlow is more difficult to work with than Keras, it's high-level wrapper. The graph and its operations are symbolic, and only exist when a TensorFlow session is created. To support this structure, the model training and data-grabbing were written in the form of functions.

3) Network Evaluation: The final component of implementing the neural network was monitoring its performance. We accomplished this by storing epoch metrics in a csv, storing a small subset of validation images with segmentation mask overlays produced by the network, and writing an Intersection of Union (IoU) function to evaluate how well our masks for each class overlapped the ground truth image.

Unfortunately, a critical weakness of our network implementation was it's inability to be loaded after a server session ended. We were able to implement a TensorFlow saver function and save model checkpoints, but we struggled to load the model from these checkpoints and resume training. This weakness and the lessons learned from it will be discussed below. It's significance in evaluating the network's performance is that we needed to evaluate the network as it was training - we were unable to load the weights of a trained network make predictions. As such, we evaluated a small subset of 33 validation images (five from each of the validation cities) and saved them to the Google Drive after each epoch. Similarly, we added provisions to evaluate the accuracy of the goodness of our network on all validation images after every ”n” epochs. This number can be set depending on how much calculations the system which is training the network can handle.

4) Intersection over Union: The most effective measure to calculate the performance of the trained network is the Intersection over Union (IoU) method. This method is the usual choice for tasks like object detection, semantic segmentation and instance segmentation.

In semantic segmentation, each pixel is classified into one of the many classes. We need to check all the false positives(FP), false negatives(FN), and true positives(TP). The IoU calculation is done by the following equation-

$$IoU = TP/(FP + TP + FN) \tag{1}$$

The IoU calculation is done for each class to find the performance of the network specific to every class. TP are the number of pixels which are classified correctly, FP are pixels which are not of the class type undr consideration but are classified as that class and FN are the pixels which belong to the class but are not classified as one of the class members. The name IoU comes from the concept of venn diagrams.For an intuitive understanding, consider two sets. The first set has all pixels which belong to a class as given by the ground truth/labels. The second set is the pixels predicted to belong to the class as estimated by the network. IoU is the ratio of intersection of these two sets to the union of these two sets.

IV. RESULTS

The comparison of the network performance is shown between the network trained before the mid-review progress report and the network trained after that. The training dataset varies.

A. *Pre-Progress Report Results*

The old network trained locally on relatively low number of training images. We used about 325 images for training. We had to run the training for about 100 epochs to get a considerable good segmentation results. This was done using Google colab. Another experiment was performed by running the training on local machine for 35 epochs. Our path forward was to improve upon these results with - more training data - saving epoch results to a csv for visualizing The preliminary results obtained using the locally trained network are shown in Figure 6. Figure 5 presents the training progression for a locally-trained implementation of the network described above over 35 epochs. Figure 6 presents a corresponding output of the trained network. Although CityScapes provides thirty classes, this early implementation was performed using just road, car, person, and traffic light (and the fifth being background).

1) Hyperparameters: Our preliminary version of the VGG-16 FCN was run with

- 174 nos of images trained
- 25% resizing of input images
- batch size of 16
- 35 epochs

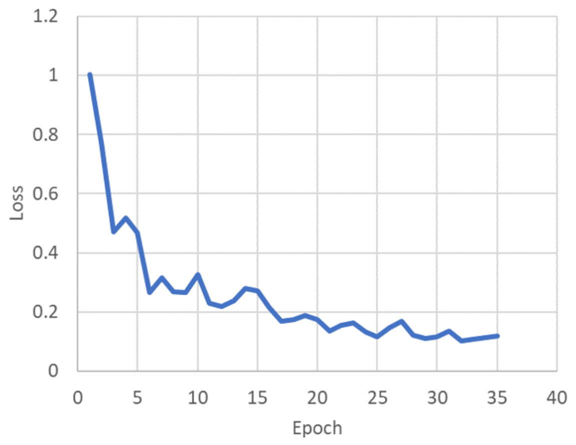


Fig. 5: Training progression for local VGG-16 implementation on 3% of CityScapes data

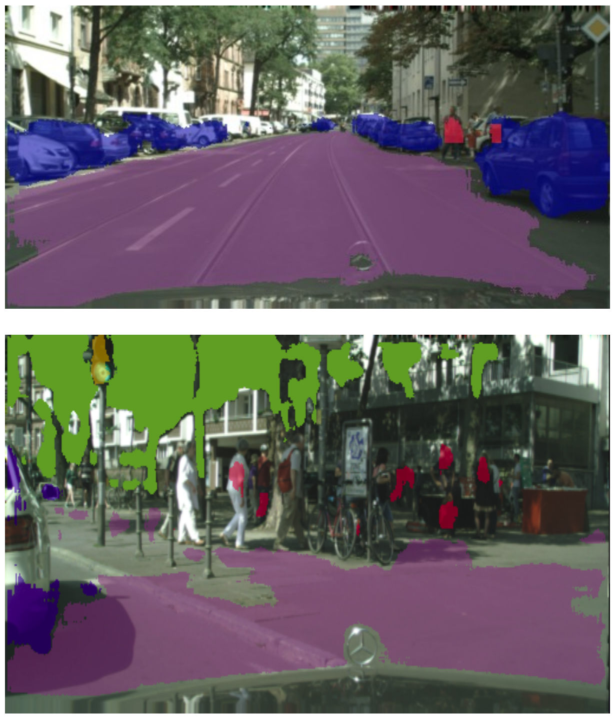


Fig. 6: Output masks of limited local VGG-16 implementation

- Adam optimizer for learning rate
- learning rate 0.0001

B. *Post-Progress Report Results*

1) Baseline: Initially the training used to crash several times due to the resources allocation from google. Resetting the runtimes and limiting the unnecesary calculations made it possible to run the network for 35 epochs on the new extensive data. In this case, we added a new major class of vegetation. All the trees and shrubs are considered in this class. The hyper parameters are listed below.

- 2774 nos of images trained

Class	IoU
road	0.91
car	0.7
person	0.65
traffic lights	0.33
vegetation	0.82

TABLE I: Best IoU of the network

- 25% resizing of input images
- batch size of 16
- 35 epochs
- Adam optimizer for learning rate
- learning rate - 0.0001

The loss for this training goes to almost 0.04. The trend of loss over epochs is shown in Fig.7.

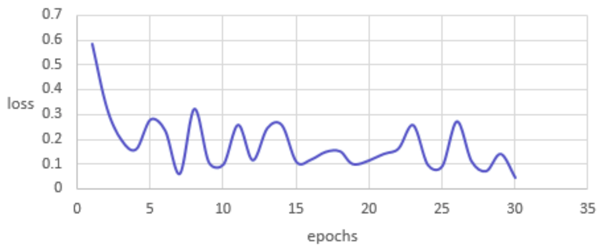


Fig. 7: epochs vs loss graph

The output of this network is extremely good. It classified every class with a high IoU. Even the moving people which are less in number in the whole dataset were classified with great accuracy. Figure 8. shows the output of the network. The mask of segmentation is overlayed on the image.The best IoU of the trained network gives in Table 1. The less IoU of traffic lights is because less number of traffic lights are available in the dataset.

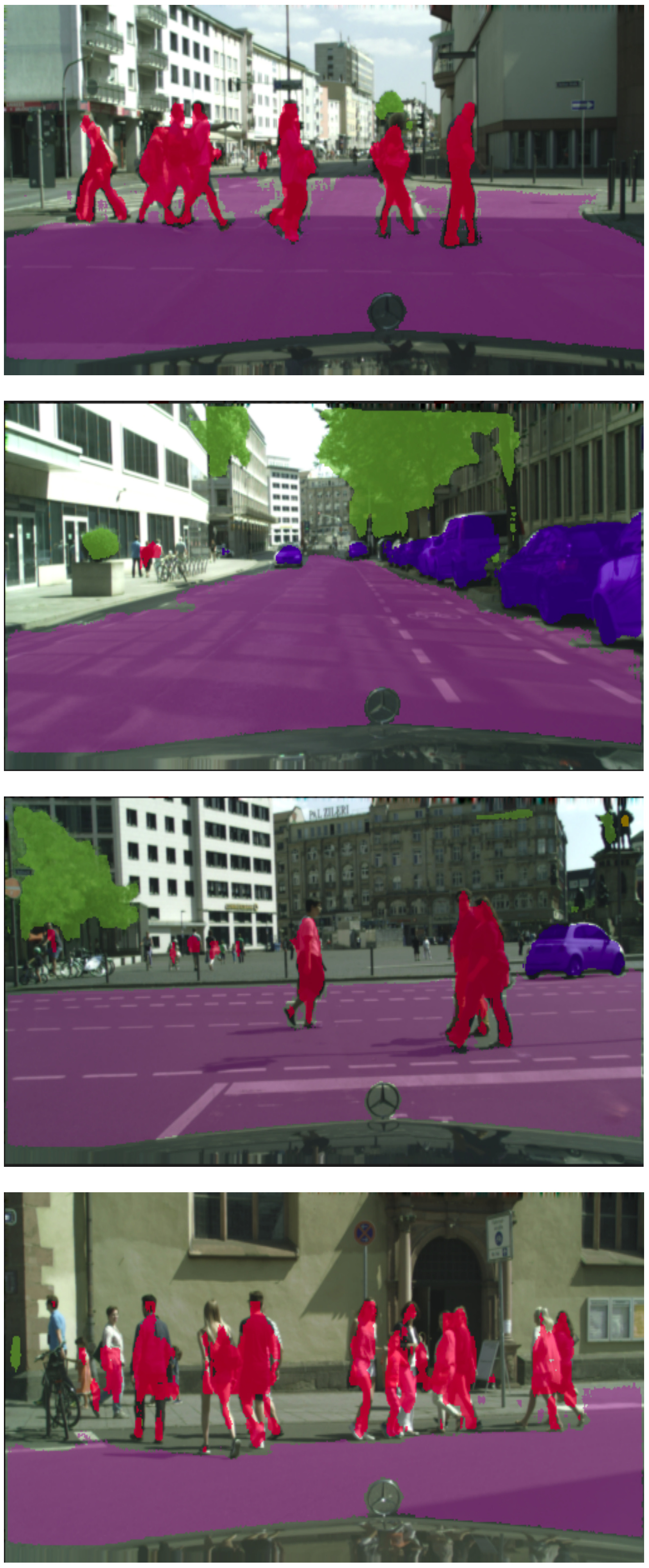
2) Modifying Learning Rate: Our baseline training was performed with a learning rate of 1 e-5. While this slow learning rate trained results quickly enough with a small training set, it proved frustratingly slow with a larger dataset. Thus, we experimented with smaller and variable learning rates to produce inferences from the model more quickly.

V. CHALLENGES AND LESSONS LEARNED

The value of an educational project lies in the lessons learned. The team is satisfied in that while the results of the trained model weren't everything we hoped for, the experience has provided several beneficial lessons that can be applied forward to future Deep Learning work. If we were to repeat this project, adapting to all or at least of subset of these lessons would greatly improve our results.

A. Transitioning to Keras

The first update we would make to our network implementation would be adapting the model from TensorFlow to Keras. Besides aligning the model with the upcoming TensorFlow 2.0, Keras is more intuitive and allows for



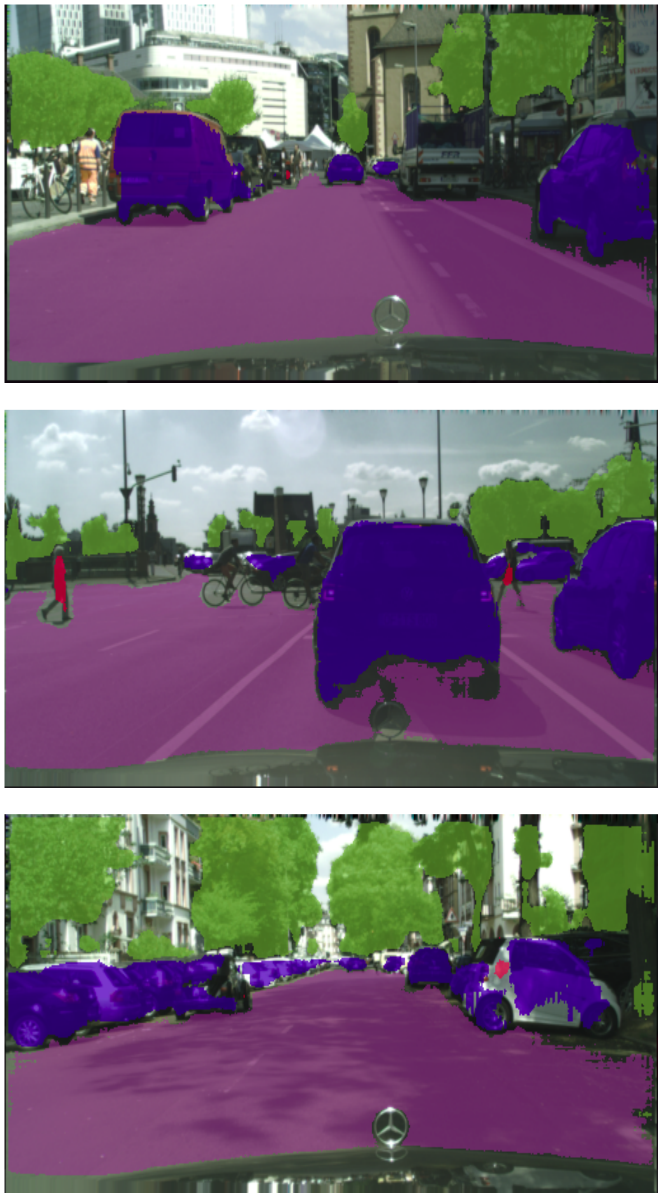


Fig. 8: Training progression for VGG-16 implementation on 2774 images of CityScapes data

quicker changes to model architecture and to perform basic model tasks like training and validation. We are confident we could rectify our challenges with loading tensorflow models as the Keras implementation of this simply requires loading an hdf5 file of the saved weights, not loading and initializing graphs and operations in TensorFlow. Given that we had numerous issues with Google Drive/Colab behavior, this would provide a significant benefit. Additionally, we would be able to train the network over many more epochs than allowed in the 12 hour time limit for a Colab session.

We briefly investigated translating the model to Keras, but we struggled to find a pretrained source of convolutionized VGG16 weights with network dimensions matching our need. More research and experimentation is required on this topic.

B. Training Platform

Two weeks ago, we presented promising results obtained on a GTX1070 local machine and expressed interest in moving to a more capable server platform to train on more data. At that point in time, we were evaluating Google Colab and the WPI cluster as solutions. Ultimately, we selected Google Colab for its friendlier interface and greater online prescence (tutorials, examples, etc.). Unfortunately, we encountered several recurring challenges implementing our code in the Colab environment.

The first inconsistency involved saving files from the code. For some reason, given the same code, one user would be able to save output images and csv files while no files would be deposited to Google Drive for the other. Workarounds were attempted including disabling extensions, resetting the chrome browser on each machine, and even upgrading storage space in each user's Google accounts to no avail. Frustratingly, the teammate was able to successfully save files in a standalone notebook with example code, but it would not work in the main project notebook.

The second inconsistency was equally frustrating. Despite the Colab environment working for long periods of time for other class assignments, it began routinely disconnecting from the cloud GPUs after only thirty minutes of training, a crippling problem for an implementation lacking the ability to load weights and restore the latest progress. Even worse was that this problem presented itself for the user who was able to save csvs and output files successfully. In other words, across multiple users of the Colab environment, multiple users encountered unique and challenging performance quirks, suggesting it may not be the right tool for tasks of this nature.

VI. Future Work

While we didn't have the opportunity to enact these experiments and improvements, the following section will discuss topics and strategies that would constitute valuable future effort to see if they would improve the model's performance.

A. Dataset Preprocessing

Given the challenges we encountered with Colab stability and continuing training across multiple instances, preprocessing our training and validation images could greatly help reduce our required resources. Rather than uploading all roughly 10 GB of training and validation images and labels to the cloud and performing image resizing and translation into batches within each epoch, we would preprocess the images into compressed hd5 files. This would reduce our data presence on the cloud and reduce the computational cost of the training loop.

B. Validating Network During Training

We previously defined the parameter 'n'. This defines after how many epochs should the validation images be tested and IoU calculations be done. After every n epochs, the IoU calculations will run on the 33 validation images and the average IoU for each class will be displayed. Having this calculations done during training will increase the time of training but we can get the evolution of IoU during the training process. This will specifically help when an adaptive learning rate is used. The online IoU calculations will show which learning rate is best affecting the network performance.

C. Further Hyperparameter Experimentation

Given more time, we would like to continue experimenting with the FCN-8 architecture by tweaking other model hyperparameters. For our results, we experimented with batch size and learning rate. In the future, we would like to adjust regularization and evaluate its impact on overfitting. Additionally, we'd like to experiment with optimizers other than ADAM.

D. Architecture Experimentation

For this project, we did not stray from the FCN-8 architecture. Modifying the architecture of the network and evaluating its impact presents a large source of future work. Implementing the model in Keras would allow for rapid minor adjustments of the model (adding and removing layers for example). Additionally, it would be interesting to modify more foundational model parameters such as CNN stride.

E. Generating More Data for Training

With the goal of creating a very generalized model for deployment, we could look into introducing image translations, rotations, and flips to increase our training image count. Another good exercise would be to evaluate the model on other semantic segmentation networks.

F. Comparing Impact of Different Base Networks

Finally, it would intersting to repeat this exercise with a different convolutionized classification network such as AlexNet or GoogLeNet. While accuracy would be interesting to compare to the VGG16 model, evaluation time would be an equally interesting metric given semantic segmention's applicability to in-the-loop control laws for autonomous vehicles. A larger undertaking would be creating our own version of the classification base layers (such as modifying CNN stride of the VGG16 model as mentioned above) and training our own model from untrained, initialized weights.

VII. CONCLUSIONS

The purpose of this project was to enable the team to explore a challenging problem facing the autonomous vehicle industry today. To that end, the project has been very successful. The team performed a detailed an in-depth literature review. It implemented a state-of-the-art semantic segmentation network into the Google Colab cloud-based servers and trained it on the CityScapes dataset. Finally, the team learned from the numerous challenges that were encountered, and discussed the many opportunities for future work following this semester.

REFERENCES

[1] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. G. Rodríguez, "A review on deep learning techniques applied to semantic segmentation," *CoRR*, vol. abs/1704.06857, 2017. arXiv: 1704.06857. [Online]. Available: http://arxiv.org/abs/1704.06857.

[2] A. T. Alentin Bazarevsky. (2018). Mobile real-time video segmentation, [Online]. Available: https://ai.googleblog.com/2018/03/mobile-real-time-video-segmentation.html (visited on 03/28/2019).

[3] A. Takeki, T. T. Trinh, R. Yoshihashi, R. Kawakami, M. Lida, and T. Naemura, "Combining deep features for object detection at various scales: Finding small birds in landscape images," *IPSJ Transactions on Computer Vision and Applications*, vol. 8:5, 2016. [Online]. Available: https://doi.org/10.1186/s41074-016-0006-z.

[4] K. Chen, K. Fu, M. Yan, X. Gao, X. Sun, and X. Wei, "Semantic segmentation of aerial images with shuffling convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, pp. 173–177, Jan. 2018. DOI: 10.1109/LGRS.2017.2778181.

[5] M. Kampffmeyer, A. Salberg, and R. Jenssen, "Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2016, pp. 680–688. DOI: 10.1109/CVPRW.2016.90.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. arXiv: 1505.04597. [Online]. Available: http://arxiv.org/abs/1505.04597.

[7] 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," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[8] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010, ISSN: 0920-5691. DOI: 10.1007/s11263-009-0275-4. [Online]. Available: http://dx.doi.org/10.1007/s11263-009-0275-4.

[9] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. arXiv: 1405.0312. [Online]. Available: http://arxiv.org/abs/1405.0312.

[10] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, pp. 88–97, 2009.

[11] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 3234–3243. DOI: 10.1109/CVPR.2016.352.

[12] C. Zhang, L. Wang, and R. Yang, "Semantic segmentation of urban scenes using dense depth maps," in *ECCV*, 2010.

[13] P. Arbelfffdfffdez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik, "Semantic segmentation using regions and parts," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 3378–3385. DOI: 10.1109/CVPR.2012.6248077.

[14] J. Carreira, R. Caseiro, J. P. Batista, and C. Sminchisescu, "Semantic segmentation with second-order pooling," in *ECCV*, 2012.

[15] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. arXiv: `1311.2524`. [Online]. Available: `http://arxiv.org/abs/1311.2524`.

[16] Y. Tian, P. Luo, X. Wang, and X. Tang, "Pedestrian detection aided by deep learning semantic tasks," *CoRR*, vol. abs/1412.0069, 2014. arXiv: `1412.0069`. [Online]. Available: `http://arxiv.org/abs/1412.0069`.

[17] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," *CoRR*, vol. abs/1512.04412, 2015. arXiv: `1512.04412`. [Online]. Available: `http://arxiv.org/abs/1512.04412`.

[18] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," *CoRR*, vol. abs/1611.07709, 2016. arXiv: `1611.07709`. [Online]. Available: `http://arxiv.org/abs/1611.07709`.

[19] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015. arXiv: `1511.07122`. [Online]. Available: `http://arxiv.org/abs/1511.07122`.

[20] L. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," *CoRR*, vol. abs/1511.03339, 2015. arXiv: `1511.03339`. [Online]. Available: `http://arxiv.org/abs/1511.03339`.

[21] T. Pan, B. Wang, G. Ding, and J.-H. Yong, "Fully convolutional neural networks with full-scale-features for semantic segmentation," in *AAAI*, 2017.

[22] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," *CoRR*, vol. abs/1505.04366, 2015. arXiv: `1505.04366`. [Online]. Available: `http://arxiv.org/abs/1505.04366`.

[23] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014. arXiv: `1411.4038`. [Online]. Available: `http://arxiv.org/abs/1411.4038`.

[24] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *CoRR*, vol. abs/1412.7062, 2014. arXiv: `1412.7062`. [Online]. Available: `http://arxiv.org/abs/1412.7062`.

[25] ——, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *CoRR*, vol. abs/1606.00915, 2016. arXiv: `1606.00915`. [Online]. Available: `http://arxiv.org/abs/1606.00915`.

[26] L. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille, "Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform," *CoRR*, vol. abs/1511.03328, 2015. arXiv: `1511.03328`. [Online]. Available: `http://arxiv.org/abs/1511.03328`.

[27] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017. arXiv: `1706.05587`. [Online]. Available: `http://arxiv.org/abs/1706.05587`.

[28] Y. He, M. Keuper, B. Schiele, and M. Fritz, "Learning dilation factors for semantic segmentation of street scenes," *CoRR*, vol. abs/1709.01956, 2017. arXiv: `1709.01956`. [Online]. Available: `http://arxiv.org/abs/1709.01956`.

[29] C. Liu, L. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," *CoRR*, vol. abs/1901.02985, 2019. arXiv: `1901.02985`. [Online]. Available: `http://arxiv.org/abs/1901.02985`.

[30] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich, "Feedforward semantic segmentation with zoom-out features," *CoRR*, vol. abs/1412.0774, 2014. arXiv: `1412.0774`. [Online]. Available: `http://arxiv.org/abs/1412.0774`.

[31] S. Hong, H. Noh, and B. Han, "Decoupled deep neural network for semi-supervised semantic segmentation," *CoRR*, vol. abs/1506.04924, 2015. arXiv: `1506.04924`. [Online]. Available: `http://arxiv.org/abs/1506.04924`.

[32] G. Lin, C. Shen, I. D. Reid, and A. van den Hengel, "Efficient piecewise training of deep structured models for semantic segmentation," *CoRR*, vol. abs/1504.01013, 2015. arXiv: `1504.01013`. [Online]. Available: `http://arxiv.org/abs/1504.01013`.

[33] S. Gidaris and N. Komodakis, "Object detection via a multi-region & semantic segmentation-aware CNN model," *CoRR*, vol. abs/1505.01749, 2015. arXiv: `1505.01749`. [Online]. Available: `http://arxiv.org/abs/1505.01749`.

[34] V. Badrinarayanan, A. Handa, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling," *CoRR*, vol. abs/1505.07293, 2015. arXiv: `1505.07293`. [Online]. Available: `http://arxiv.org/abs/1505.07293`.

[35] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *CoRR*, vol. abs/1606.02147, 2016. arXiv: `1606.02147`. [Online]. Available: `http://arxiv.org/abs/1606.02147`.

[36] P. Luc, C. Couprie, S. Chintala, and J. Verbeek, "Semantic segmentation using adversarial networks," *CoRR*, vol. abs/1611.08408, 2016. arXiv: `1611.08408`. [Online]. Available: `http://arxiv.org/abs/1611.08408`.

[37] X. Li, Z. Liu, P. Luo, C. C. Loy, and X. Tang, "Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade," *CoRR*, vol. abs/1704.01344, 2017. arXiv: `1704.01344`. [Online]. Available: `http://arxiv.org/abs/1704.01344`.

[38] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. W. Cottrell, "Understanding convolution for semantic segmentation," *CoRR*, vol. abs/1702.08502, 2017. arXiv: `1702.08502`. [Online]. Available: `http://arxiv.org/abs/1702.08502`.

[39] J. Le. (2018). How to do semantic segmentation using deep learning, [Online]. Available: `https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef` (visited on 03/28/2019).