Peer Review

Comments by group 33

Positive:

- Presentation is really audible and clear, and really easy to understand and easy to follow.
- The graphs of the models are really clear, and combine really well with the presentation.

Critiques:

- For the result section, since you have mentioned that only small dataset is applied, are there ways to improve the accuracy of your model when still using the small dataset?
- In the result section, you have listed the mIoU metric from SOTA, but in your model measurement you did not show your mIoU values for us to compare the results. Since mIoU is so crucial, we think it is important to see your mIoU value as well.
- Have you used the SOTA model to train on the same dataset to see differences in results?
- The models you have listed are from a few years ago, is there any new change in model in recent years on the topic? If there are new models in recent years, why did you not choose them instead? Is there a particular reason for you guys to use the current model?

Response:

- Thank you for liking our presentation ability and overall aesthetic.
- As mentioned in the presentation, skip connections, fewer parameters, transfer learning, and data augmentation can be used to mitigate the issues with small datasets. This is further detailed in Section-4 of the paper
- As mentioned, the dataset we were using did not have categorical labels, so we used pixel accuracy and Manhattan score to gauge the performance of our model initially. IoU and labeling of data have been implemented for the paper and mentioned in Section-5 and Section-6.
- We did not train the SoTA as the pre-trained model was not publicly available, and we did not have the resources to implement and train this ourselves. The results were publicly available for the full dataset and this is what was used as a comparison.
- Implementing SoTA was not a requirement for this project. U-net++ came out in 2018 and is still widely used as a semantic segmentation architecture. Again, due to limitations in time and computational resources, we chose not to do additional newer models.

Comments by group 32

Positive:

- The problem addressed is an important one, not only for autonomous vehicles but also for other problems in the intersection of robotics and computer vision.
- The idea behind the problem is addressed clearly and the group does a good job of presenting the solution as well.
- The models used were explained well in the context of the problem.

Critiques:

- The group says that deep learning doesn't require much data, which seems counter-intuitive. Is it possible for the group to use a larger subset of Cityscapes to prove that the metrics do not improve much on increasing the data?

- There could have been more discussion on the metrics used. While the group members explained why intersection over union was not used, they did not explain why they used Manhattan Score or Pixel accuracy, or what these were. Also, they used a different metric for comparison with the state of the art method, which makes it difficult to understand how the models used by the group performed. It will be great if the same metrics can be used across all models.
- The discussion about mean squared error on the slide about observations seemed out of the blue. There was no discussion about MSE before, and it wasn't used as a metric in any of the models. Some more context regarding MSE in the report would be perfect.
- The group mentioned that training for too long led to the kernel dying and they had to start training from scratch again. Did the group try using checkpoints to save the model after every few epochs?

Response:

- Thank you for understanding the scope of the problem and how we address it.
- We mentioned that deep learning does not *necessarily* require much data. Our project was to showcase that even with a subset of the data in a larger problem, you can achieve decent results with deep learning by using different methods. Deep learning is a data-driven approach and does require data to be successful.
- Section-4 goes into detail on Pixel Accuracy and MSE, while Section-5 and Section-6 discuss cross-entropy loss, labeling, and IoU.
- Checkpoints are a good suggestion. Often the kernel would die even on the initial epoch, but this would have helped for successful runs. We moved to Google Colab which helped some of our issues as well.

Comments by group 2

Positive: Group 10 introduces some of the mainly used models for Semantic Segmentation. Our project also involves the use of the semantic segmentation model. Therefore, from their slides, we can see that group 10 did some thorough research towards different semantic segmentation models. Moreover, their video demonstrated the intention of why they chose this field to explore as their final project, and we fairly agreed with what they demonstrated. Furthermore, they used different evaluation metrics when comparing the performance of different models, which we think is comprehensive. Overall, they did a great job on explaining the different semantic segmentation models.

Critiques:

- Personally, we think that the entire project lacks new ideas and novel approaches. It's mainly a comparison of different models.
- It is true that they mentioned state-of-art models for semantic segmentation. However, they did not compare the most recent state-of-art implementations. As a project mainly focusing on the comparison between different models, it is important to add the comparison with the state-of-art models.
- For more comprehensive comparison results, it's recommended to test the models on different datasets instead of a single dataset.

Response:

- Thank you for recognizing the breadth of the topic we covered.
- It was not listed in the requirements to implement novel approaches, and we thought it would provide a better learning experience to us to compare the benefits of different models. U-net++ [1] was released in 2018 and is a common advanced approach that is lightweight and used in the field.
- While we included reported results, we did not discuss the model architectures of the state-of-the-art in detail. This is now in Section-2 with comparisons to our models in Section-6.
- Our project was focused on differences in model architectures and how different models adapt to this specific problem. We limited the scope to a single dataset to ensure we had the time to complete the different implementations and testing.

GROUP 10: SEMANTIC SEGMENTATION FOR AUTONOMOUS VEHICLES

Samuel Cowin[⋆†], *Jianghong Wan*[⋆†], *and Vivaswat Suresh*[⋆‡]

[⋆]University of California San Diego, La Jolla, CA 92093-0238,
[†]Department of Electrical and Computer Engineering, [‡]Department of Mathematics

ABSTRACT

Semantic segmentation through deep learning is a novel area of advanced computer vision used to solve problems in automotive, medical, and other industries. There are a multitude of methods used for segmentation, each with its own advantages and disadvantages. In this paper, different models for segmentation including UNet, UNet++, and SegNet, are implemented from scratch to understand which model gives the best results with limited data. In addition, other methods for handling small data in the scope of this problem will be introduced and tested.

1. INTRODUCTION

The autonomous vehicle industry has one of the highest projected growth rates of any industry with a market that is expected to grow past 65.3 billion USD by 2027 [2]. Thus, large companies like Amazon, Google, and Tesla are all seeking to capitalize on autonomous vehicles. Much of this projected growth can be attributed to advances in machine learning technology which give autonomous vehicles a greater scene understanding. One such technology is semantic segmentation: the process of assigning every pixel in a given image with a different label, such as car, stop sign, or traffic light.

This project uses three segmentation models that employ different deep learning techniques to identify the objects in the urban scene: U-net, a U-net++, and SegNet. To train the models, inputs are taken from the Cityscapes dataset [3] with the input being a RGB image, and the output being the segmented image. To test the performance, the metrics chosen are Intersection-over-Union (IoU), Manhattan score, and pixel accuracy which are then compared to the reported state-of-the-art (SOTA) for this problem.

2. RELATED WORK

There are three high level groupings that semantic segmentation algorithms can be bucketed into: fully convolutional networks, adaptations to these layers, and regional methods. Examples of fully convolutional can include FCNs [4], U-nets [5] [1], and other encoder-decoder architectures [6]. Regarding layer adaptations, there are Altrous Convolution [7] and Joint Pyramid Upsampling [8] as examples. Finally, for regional methods there are Mask R-CNN [9], Gated-SCNN[10], HRNet [11], and transformer methods [12] [13] among others.

While all of these methods are used and can produce usable semantic segmentation results, recently the most profound advances have been in adapting the encoder-decoder framework to fit with the transformer model as mentioned in [12] and [13] above often using other models such as Deeplab (Altrous Convolution) or HRNet as a trained backbone. The issue with many of these approaches is they require significant data and are high in parameters in order to learn their impressive results. With this in mind, this paper tackles the "small" data problem with fully convolutional methods in order to keep the parameters low and the overhead for training at a minimum while showcasing possible results that can be achieved.

3. DATASET

A subset of the Cityscapes dataset[1] was used for this project. This was purposeful as the project is to examine different methods in handling a smaller dataset in the scope of a difficult problem. This dataset provides classes such as "bicycle" or "road" to be used in segmenting the image with examples included in Section-6. From the data given, there are 2975 training images and 500 validation images. The validation images were used for testing, and 10% of the training data was used instead as a validation holdout in order to form a training/validation/test split. The images were of size 256x256x3 and after labeling the ground truth images were 256x256x30.

The data provided was not categorical in the sense that the labels were pixel color instead of assigned a class. This meant that the normal semantic segmentation problem using IoU and cross-entropy needed to be replaced for the initial testing. After initial prototyping, labeling and new loss and metrics were used which are further detailed later in the paper. These labels would not work for other computer vision tasks such as instance segmentation, or object detection, and the

Project for ECE 228/SIOC 228: Machine Learning for Physical Applications, Spring 2021, Group 10.

[1]https://www.kaggle.com/dansbecker/cityscapes-image-pairs

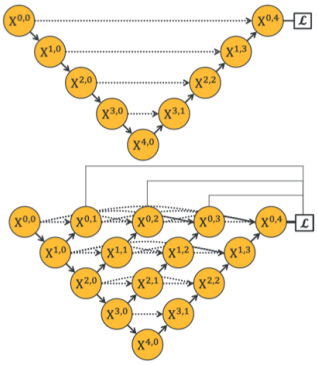

Fig. 1. U-net architecture compared with U-net++

labels from the full dataset would need to be utilized.

The final aspect of the data that needed to be handled was the pre-processing. The input images and labels were attached as one image, so the images needed to be divided up and filtered into different categories. From there, the images were normalized from 0-255 to 0-1. Additionally, data augmentation was applied by flipping the dataset images horizontally in addition to the items within the dataset to expand the dataset and provide more robust training. Further augmentation was avoided to ensure that the input images would still correlate to real world data.

4. METHODS

As mentioned in Section-2, there are many algorithms in which to choose for tackling the semantic segmentation problem. Many of these architectures are extensions of U-net or FCN. U-net, being specifically designed for a small data biomedical imaging problem, was thus chosen to see the extensions that could be made into autonomous vehicles. Transfer learning was considered, but as mentioned in Section-3, the labels provided were not categorical, and thus labeling would need to be done across the entire dataset to in order to match the pre-trained models for this problem. This was not done until after initial prototyping and testing. Finally, this study is to showcase the differences in methods to handle the small data problem, and while transfer learning is one way around this, this project focuses on model complexity, size, and configurations for evaluating differences in addition to data augmentation.

This will be shown in the classic U-net [5], extensions into the newer U-net++ [1], and alternative encoder-decoder structures in SegNet [6] models. Semantic segmentation state-of-the-art models were not chosen as this project is developing and training all of these models from scratch. Deploying the more complicated and higher parameter count state-of-the-art models were out of scope for the time-frame and computational power available in the project.

Now that the rationale has been laid out, the details in the models need to be discussed. As each model has the encoder-decoder structure, this will be detailed prior to the extensions each model uses. The high-level overview of the encoder-decoder architecture can be seen with Figure-1 by ignoring the skip connections (dotted arrows) - there are convolutional and pooling layers (solid downward arrows) in the contracting path followed by convolutional and upsampling layers (solid upward arrows) in the expanding path. Contracting refers to the narrowing of the information that is kept in that layer of the model, which showcases that the end of the contracting path is the bottleneck for this architecture. This bottleneck serves as a dimensionality reduction in the sense that there is less information that is being stored, but the most important information is maintained to be fed through the expanding path. Each convolutional layer has a non-linear activation function to enable complex learning, and specifically the non-linear function used for this example is ReLU with the output function being sigmoid. These were chosen as the image pixels are non-negative, thus ReLU can be used without degrading the output, and sigmoid outputs 0 to 1 values which aligns with the normalized pixel images after preprocessing. The equations for these activation functions are included below:

$$\text{ReLU} = \max(0, \mathbf{x}), \ \text{Sigmoid} = \frac{1}{1 + e^{-\mathbf{x}}}$$

Dropout is used in order to regularize the model following these activation functions, making sure to scale the weights by the inverse of the keep probability to compensate between training (with dropped nodes) and testing (with all nodes). Finally, average pooling is commonly used which refers to taking the pixel values for a specified region, calculating the average value among them, and then storing this single value in place of the larger block of numbers moving forward while default upsampling is just copying the input values a number of times to fit the desired dimensionality.

With the foundation of these models outlined, the U-net model modifications are next. This model takes the core architecture and adds skip connections. These connections are from the outputs of the convolutional layers in the contracting path, and connect to the output of the convolutional layers corresponding to the same spatial dimension in the expanding path. This allows the model to preserve knowledge of the spatial importance in the image that was lost in the bottleneck contraction of the encoding architecture and the concept is showcased in Figure-1 along with U-net++. For U-net++, the main difference is in the number of concatenations and nodes in the network.

The other model that was implemented was the SegNet model. For this, the main encoder-decoder architecture remains completely in tact, but the pooling and upsampling operations are modified. Instead of average pooling, the operation that is completed is max pooling (taking the max value not average) while tracking the pixel indices in which the

pooling applies to. Instead of upsampling, those indices that were used in the max pooling are used as fill-ins for upsampling (or unpooling) while the remaining indices are left as zero. This is then passed through convolutional layers to train the filters. In this way, boundary information is well maintained when compared with a simple autoencoder since the locations of maximum features are maintained, and the reconstruction (and corresponding segmentation) is much better.

With the models detailed, the next step is explaining the training and validation metrics. Based on the filter and kernel sizes for the models, the parameters for each were 600k for U-net++, 550k for U-net, and 280k for SegNet corresponding to the relative training times needed. As mentioned prior, uncharacteristically for this problem, the labels were provided as pixel colors and not categorical values. This means the standard Intersection-over-Union score cannot be applied for either training or evaluation. Instead, a regression-like problem with the mentioned sigmoid activation is applied for the pixels to match the label output by using Mean Squared Error for training to optimize against matching the color closely. The formula is provided here:

$$MSE = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$$

Given this training objective and the non-categorical labels, the commonly used Intersection-over-Union was abandoned for Pixel Accuracy and Manhattan score (L1 distance per pixel) as detailed below:

$$\text{Pixel Accuracy} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{I}(Y_i = \hat{Y}_i),\ L1 = \frac{1}{n}\sum_{i=1}^{n}|(Y_i - \hat{Y}_i)|$$

5. EXPERIMENTS

As mentioned in previous sections, the experiments are conducted over 2975 images in the training set, 10% of which are used as validation data instead, and the test set containing an additional 500 images. For all of the models, the first convolutional layer learns from 8 filters, and after each pooling layer, the number of filters doubles, and the number of filters is halved after each upsampling layer. The changes of filter numbers ensure the capture of patterns at each layer. The reason why 8 filters are used as initial filter size is to showcase a smaller network that would be easier to train with limited data. For the models, after each pooling layer, dropout with 0.5 dropout rate is performed. The reason for 0.5 dropout rate is to overcome the problem of over-fitting, and the ratio is chosen based on experiment results.

All models are trained with batch size 32 for 10 epochs. An additional 5 epochs with batch size 32 are trained over the augmented training data, i.e. the horizontally flipped images, which is used to overcome disadvantages of having a small dataset. Due to the fact that the pixels are not categorical, mean square error is used as the metrics for all three models.

Furthermore, to potentially improve the results and be able to use IoU (Jaccard Index), another experiment on categorized data is conducted using U-net++. Images are categorized into 30 classes based on binning pixel values changing the output shape of the model to be 30 classes per pixel instead of the 3 color channels. From there, U-net++ is trained with the same hyper-parameters mentioned above, except categorical cross entropy loss with softmax activation (class scores) are used. These equations are below followed by IoU:

$$S(\mathbf{z}) = \frac{e^{z_c}}{\sum_{j=1}^{C} e^{z_j}},\ CE = -log(S(\mathbf{z}))$$

$$J(\mathbf{A},\mathbf{B}) = \frac{|\mathbf{A}\cap\mathbf{B}|}{|\mathbf{A}\cup\mathbf{B}|} = \frac{|\mathbf{A}\cap\mathbf{B}|}{|\mathbf{A}|+|\mathbf{B}|-|\mathbf{A}\cap\mathbf{B}|}$$

6. RESULTS AND DISCUSSION

Figure-2 shows the loss curve of U-net on training set with data augmentation. The training was done in stages so only the final epochs are shown. In the plot, the issue with overfitting exists despite the use of dropout.

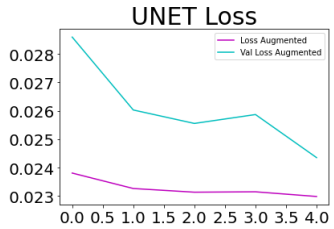

Fig. 2. U-net Loss Curve w.r.t. epochs

Fig. 3. U-net output example with original (left), ground truth (middle), and prediction (right) shown

U-net achieves 45% pixel-wise accuracy, and 0.21 Manhattan Score per pixel. Figure-3 shows an example of the output prediction by U-net. On the left is the original image, and the ground truth label is shown on the middle, whereas the prediction is shown on the right. As shown in the figure, the output prediction captures the essential features of the original image, but with lower resolution and loss of details.

Figure-4 show the loss curve of U-net++ on the training set with data augmentation. The overfitting is reduced, indi-

cating this model is better suited for the task and is able to generalize learning to new data better than the original U-net.

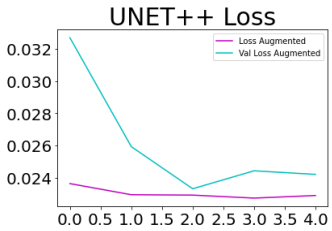

Fig. 4. U-net++ Loss Curve w.r.t. epochs

U-net++ achieves 62% pixel-wise accuracy, with 0.17 Manhattan Score per pixel. As Figure-5 shows, U-net++ reconstructions reserve more details of the scene than U-net, which agrees with the metrics that U-net++ outperforms U-net.

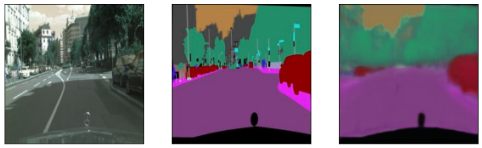

Fig. 5. U-net++ output example with original (left), ground truth (middle), and prediction (right) shown

The categorized U-net++ (after labeling the data) output achieves 77% pixel accuracy, as opposed to 62% mentioned prior. The mIoU score over the test set in this case is 44%. Figure-6 shows an example of the output. On the left is the masked image of prediction, and on the right is the ground truth masked image.

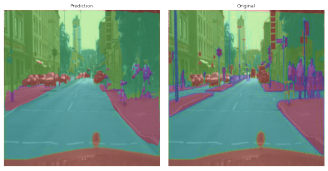

Fig. 6. U-net++ output example on categorized data with masked input by prediction (left) and ground truth (right) shown

Figure-7 shows the loss plot of SegNet. The loss behaves similar to U-net, where overfitting occurs. Clearly, the simplification in U-net and SegNet is causing issues as they are able to match the training data well without having the capacity to use what is learned on new examples. In this way, the U-net++, being more complicated, is more applicable for this problem. SegNet exhibits the worst performance out of three models, with 0.349 Manhattan Score per pixel, and 22% pixel

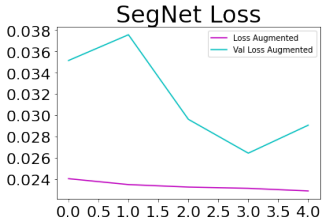

Fig. 7. SegNet Loss curve w.r.t. epochs

accuracy. As shown in Figure-8, a large amount of details are lost during prediction. In general, this shows skip connections having a profound effect on the output resolution.

Fig. 8. SegNet output example with original (left), ground truth (middle), and prediction (right) shown

Final comments regarding the results are how they compare with the SOTA for the Cityscapes dataset [2]. Using transformers as mentioned in Section-2, results of 85% mIoU across the test set are realized. This is almost double the performance achieved by the U-net++ implementation created in this paper. This is understandable, as the U-net++ architecture reported here did not train on the full dataset or extra data, used 600k parameters (compared to 10 million for the SOTA), and was trained for minimal epochs. With that being said, the results generated from [1] show U-net++ reaching 75% mIoU on the Cityscapes dataset. While still far off the SOTA, the results shown in this paper indicate that even with relatively low output performance, information is still extracted from the output reconstructions.

7. CONCLUSION

In this project, three semantic segmentation models were implemented from scratch: U-net, U-net++, and SegNet. In terms of performance, U-net++ performs the best among three models, but it has the most parameters which leads to longest training time. U-net performs slightly worse than U-net++, while SegNet performs the worst among the three, however, it is the fastest to train. Overall, even with a small version of a larger dataset, reasonable results are attainable through modifications of known networks and tuning hyperparameters. The limitations of this project stem from computational resources preventing further model exploration and tuning of performance.

[2]https://paperswithcode.com/sota/semantic-segmentation-on-cityscapes

8. REFERENCES

[1] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 3–11. Springer, 2018.

[2] M.R. Future. Autonomous Vehicle Market is Projected to Surpass USD 65.3 Billion by 2027 — North America to Command Largest Share in Global Autonomous Vehicles Industry, 02 2021.

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[4] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[6] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation, 2016.

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[8] Huikai Wu, Junge Zhang, Kaiqi Huang, Kongming Liang, and Yizhou Yu. Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation, 2019.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018.

[10] Towaki Takikawa, David Acuna, Varun Jampani, and Sanja Fidler. Gated-scnn: Gated shape cnns for semantic segmentation, 2019.

[11] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition, 2020.

[12] Yuhui Yuan, Xiaokang Chen, Xilin Chen, and Jingdong Wang. Segmentation transformer: Object-contextual representations for semantic segmentation, 2021.

[13] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers, 2021.

[14] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.

[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[16] Irem Ulku and Erdem Akagunduz. A survey on deep learning-based architectures for semantic segmentation on 2d images, 2021.

[17] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[18] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Contributions

An overview of the group member contributions are included here. Extraneous items to those listed were a group effort.

Samuel: Created the U-net and U-net++ model architectures, created the data extraction and pre-processing methods, developed a generator function to feed in data for training, augment the data, and ultimately label the data, created metric computation functions, ran model implementations, and finally worked on the presentation and paper (Section-3, Section-4).

Jianghong: Tested the FCN model and ran experiments on an existing FCN one-image-trainer, developed the labeling function, created the SegNet architecture, and finally worked on the presentation and paper (Section-5, Section-6, Section-7).

Vivaswat: Ran U-net/U-net++ model implementations, assisted with U-net++ model development and testing, and finally worked on the presentation and paper (Abstract, Section-1, Section-2).