Image Segmentation using SegNet and Analysis

Murugan Viswanathan
Georgia Institute of Technology
Atlanta, GA
mviswathana3@gatech,edu

Paresh Upadhyay
Georgia Institute of Technology
Atlanta, GA
pupadhyay8@gatech.edu

Abstract

Image segmentation is a pivotal and one of the most extensively studied area in image processing and computer vision. Its applications span across diverse fields including enhancing robotic perception, video surveillance and facilitating augmented realty experiences. This work explores semantic image segmentation using the SegNet model[1]. We leverage the widely adopted Pascal VOC dataset and employ a transfer learning strategy with pretrained VGG16 weights as described in the SegNet original paper. We investigate two training approaches: freezing encoder weights for overfitting mitigation and fine-tuning both encoder and decoder for improved adaptation, with a focus on mitigating challenges related to computational resources, limited dataset characteristics, and model training. Using special hardware GPU, we conduct multiple trainings and hyper-parameter tunings. Despite resource limitations, we achieved a reasonable pixel accuracy. We also compare research studies on other models such as FCN[3]. Finally, we highlight the insights learned in this Deep Learning exploration.

1. Introduction

1.1. Semantic Segmentation using SegNet:

SegNet is a Deep Convolutional Encoder-Decoder architecture designed for semantic pixel-wise segmentation[1]. In this project we implement the SegNet model for the Semantic Image Segmentation task. Our primary objective was to tackle the challenge of precisely identifying and segmenting objects within images using this model. SegNet is designed to be efficient both in terms of memory and computational time during Inference[1]. It is also smaller in number of trainable parameters compared to other competing architectures. Given this, we wanted to explore the effectiveness of training a Deep Learning model such as SegNet models in a resource constrained environment and have a feel for the nuances of training complex Deep Learning models.

1.2. Existing Models and Limitations:

Current practices in Semantic Segmentation often rely on deep neural network architectures tailored for the specific task. The most recent techniques used for semantic segmentation are ViTs[7], Multi-scale Attention mechanism and multi-model fusion techniques[8] to name a few. However, these existing models can be computationally very expensive with high memory and resource consumption, limiting their applicability, especially in real-time scenarios. The limitations of existing approaches include the trade-off between accuracy and computational efficiency, making it challenging to deploy sophisticated models on resource-constrained devices such as Embedded devices.

1.3. SegNet Applications:

SegNet offers a unique solution by utilizing max-pooling indices for efficient upsampling, resulting in lower parameters and hence memory efficiency. Also, it's fast inference time is suitable for real-time applications, Hence, the SegNet model holds significant importance in the fields of computer vision for resource constrained devices such as embedded systems, real-time inference systems such as Medical imaging and Autonomous Vehicles. Precise semantic segmentation can revolutionize autonomous vehicles' perception systems. Additionally, industries relying on object detection, such as robotics and surveillance, can benefit from enhanced scene understanding. Therefore, we believe that, our project aimed at training SegNet in our resource constrained environment would be of a significant importance and would provide us with a great learning experience.

1.4. Pascal VOC dataset[4]:

SegNet has been empirically validated on various datasets. For our specific implementation, we chose the Pas-

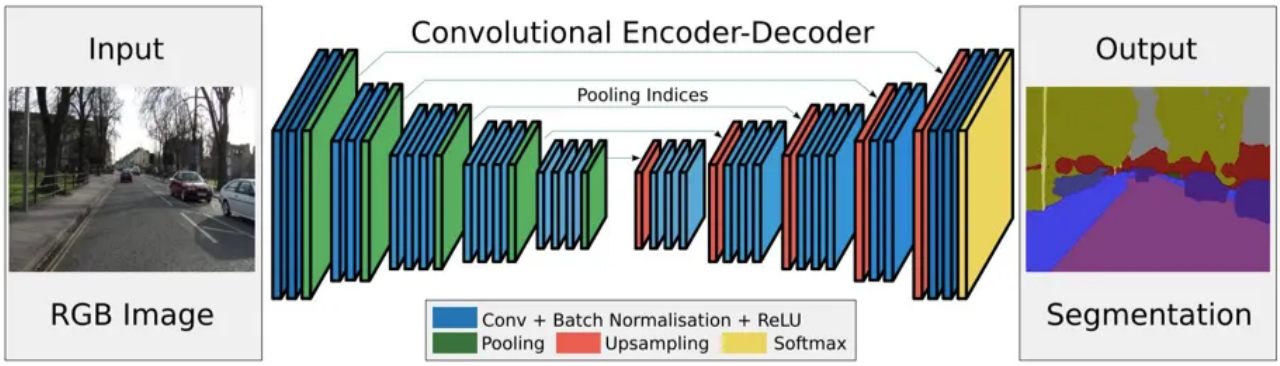

Figure 1. SegNet: Architecture (image from: https://doi.org/10.48550/arXiv.1511.00561)

cal VOC dataset due to several key considerations:

Widespread Adoption in Research: Pascal VOC stands as a widely recognized benchmark dataset in the computer vision research community. Its extensive use makes it a standard for evaluating the performance of semantic segmentation models.

Diversity of Object Classes: The dataset encompasses a broad spectrum of object categories, including people, animals, and vehicles. This diversity makes Pascal VOC well-suited for semantic segmentation tasks where the objective is to classify and segment various objects within an image.

Pixel-wise Annotations: Pascal VOC provides detailed pixel-wise annotations, precisely delineating segmentation masks for each object instance in the images. This feature is crucial for training pixel-wise segmentation models like SegNet, ensuring accurate object boundary delineation.

Real-world Applicability: The objects present in Pascal VOC represent common everyday items, imparting real-world relevance to the dataset. However, it's important to note that images in Pascal VOC are relatively simpler compared to real-world scenarios, potentially limiting the generalizability of models to more complex environments.

Challenges with Limited Training Data: Pascal VOC is comparatively small in size when compared to some other datasets. While this presents a challenge, it also mirrors a realistic scenario. In our model, we implement transfer learning to capitalize on knowledge acquired from a larger dataset (e.g., VGG16 trained on a more extensive dataset), thereby enhancing performance on Pascal VOC.

2. Approach

2.1. SegNet model implementation:

Our project is focused on Semantic Segmentation of Images. For this we implemented the *SegNet* model architecture, as detailed in the paper[1].

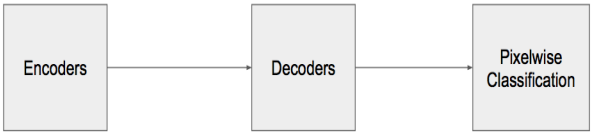

Figure 2. SegNet in a nutshell
(Image from: https://saytosid.github.io/segnet/)

We selected SegNet because it is specifically designed for semantic segmentation tasks. Its effectiveness has been established through research studies, making it a reliable choice for segmentation tasks. Furthermore, we were confident in making this model work for our Semantic segmentation task due to its widespread community support, discussions, and the availability of resources.

SegNet comprises an encoder network, a corresponding decoder network, and a final pixelwise classification layer, offering an effective solution for high-resolution image segmentation tasks.

The SegNet architecture, briefly summarized[1], involves an encoder network with 13 convolutional layers, resembling the initial layers of the VGG16 model. To initialize the encoder weights, we employed a transfer learning strategy, leveraging a pretrained VGG16 model[6] trained on a large dataset to transfer knowledge to our segmentation model.

2.2. Transfer Learning strategy

In our SegNet architecture, as described in the original paper[1] we employed Transfer learning, Transfer Learning involves using pre-trained models on a different but related task to initialize our model's weights. In our implementation we used the transfer learning approach using a pretrained VGG16 model from the PyTorch library[2] to initialize the Encoder layers' weights. We considered the following two approaches:

Approach 1: Freezing Encoder Weights: In this approach, we froze the encoder weights post-initialization with pretrained VGG16 weights. This prevented the encoder weights from being updated during training, enabling the model to focus on learning specific features pertaining to Semantic Segmentation task. This strategy proved advantageous, especially considering our use of the Pascal VOC training dataset, which is relatively small. Freezing encoder weights helped mitigate the risk of overfitting.

Approach 2: Training both Encoder and Decoder: We also explored fine-tuning both the encoder and decoder. This approach aimed to allow the model to adapt the learned features from VGG16 to the specifics of the segmentation task. However, we observed an increase in training time over Approach 1. But we found the pixel accuracy improved over the time using this, hence we decided to go with this approach. Using multiple GPUs, we tried to mitigate this issue of increased training time, while being cautious about overfitting to the small VOC training dataset. We mitigated this issue by performing image augmentation. Over several days of training, we were able to achieve a reasonably good accuracy using this strategy.

2.3. Novelty of SegNet:

2.3.1 Max-Pooling Indices for Efficient Upsampling:

The Novelty of SegNet resides in the unique approach used in the decoder to upsample lower-resolution input feature maps. Instead of learning to upsample, it uses the indices of the max-pooled values from the encoder. This approach is memory-efficient because it avoids the learning for upsampling filters. We had a limited computational resources in terms of GPU memory and speed. So we found this model to be more appropriate for our implementation.

2.3.2 Low Inference Time and Memory Efficiency:

Segnet is also designed to be efficient both in terms of memory and computation time during inference. We found this attractive since it is adaptable to real time domains such as medical imaging, autonomous vehicles, and robotics. It also has significantly small number of trainable parameters compared to other image segmentation architectures.

2.4. Risks and Mitigation:

After selecting the SegNet model with Transfer Learning using VGG16 for implementation and choosing Pascal VOC dataset for training, we identified the following Risks and Mitigations:

2.4.1 Computational Resources:

Training deep neural networks, especially with fine-tuning, can be computationally intensive and time-consuming. We were very concerned about our ability to perfect the model especially with the limited compute resources we had.

We tried using the cloud computing resources with GPUs to accelerate training, but we were unsuccessful in getting the configurations working properly for the cloud recourses and were bottlenecked without much support.

We were able to mitigate this issue by locally setting up a high-powered computer with a Nvidia Quadro RTX 4000, 8GB GPU. We also optimized the batch size to balance between model performance and training efficiency.

2.4.2 Limited Training Data:

PASCAL VOC 2012 is relatively small for deep learning standard. Training extensively on a small dataset may lead to overfitting to the training data. We were mainly concerned about our model overfitting to this training data and not generalizing well.

We encountered this problem initially during our training. We observed that the pixel accuracy improved for the training data when trained for a longer duration, but performed poorly in the validation images.

We were able to mitigate this issue and train better by using data augmentation techniques such as rotations, flips, and scaling of the images. Additionally, we employed transfer learning with pretrained VGG16 weights (using PyTorch[2] library) to leverage knowledge from a larger dataset.

2.4.3 Choosing Training Strategy:

Deciding whether to freeze the pretrained VGG16 encoder weights or fine-tune both the encoder and decoder during training was challenging. We were aware that freezing the encoder may help prevent overfitting on a small dataset such as pascal VOC, but it might not adapt well to the specific characteristics of the segmentation task on PASCAL VOC.

On the other hand, fine-tuning allows the model to adapt to specific segmentation task characteristics, but risks overfitting, especially to a smaller data such as Pascal VOC.

We experimented with both the strategies and using careful hyperparameter tuning, Image augmentations and considering of the compute resources such as GPU and mem-

ory considerations we were able to experimentally arrive at an optimal result for our task.

2.4.4 Class Imbalance Issues:

We were also concerned about class imbalance issue where some classes in the Pascal VOC may be underrepresented and may not perform well in segmentation tasks. We were ready to implement class-aware loss weighting or other techniques to address the class imbalance issue. But we did not encounter this issue during our experiments.

2.4.5 Evaluation Metrics:

We expected some challenges choosing appropriate evaluation metrics for semantic segmentation, such as mIoU (mean Intersection over Union). We regularly evaluated the model's performance on pixel accuracy on validation data and used mIoU to gain insights into segmentation quality.

3. Experiments and Results

3.1. Setup

Pre-trained Model (VGG16): We leveraged the transfer learning capabilities of pretrained VGG16 architecture from PyTorch[2]. This strategic choice enabled us to capitalize on the knowledge acquired by the VGG16 from a diverse and large set of images, he downside being the trained dataset is on a a widely different set of classes. We experiment with both freezing the Encoder to capitalize on learned features from the VGG16 and well as fine-tuning the encoder to allow the model to learn Image Segmentation features.

Dataset (Pascal VOC2012): For the purpose of our experiment, we opted for the Pascal VOC2012 dataset[4], consisting of a modest number of training and validation images (1464 and 1449, respectively)[4]. This deliberate choice facilitated a more focused training approach, addressing challenges associated with limited data and available compute resources

Computational Resources and GPU Setup: Addressing the crucial need for a higher compute resource, we obtained a high-powered computer with NVIDIA GPU Quadro RTX4000, 8GB installed. This helped to address the challenges associated with prolonged training times, making it possible for us to fine-tune the hyper-parameters.

3.2. Measurement Parameters and Techniques used

3.2.1 Metrics (Pixel Accuracy, mIOU):

We selected Pixel Accuracy and Mean Intersection over Union (mIOU) as our primary metrics for evaluation. These segmentation-specific metrics provided us with comprehensive insights into both the overall training and validation performance of our model

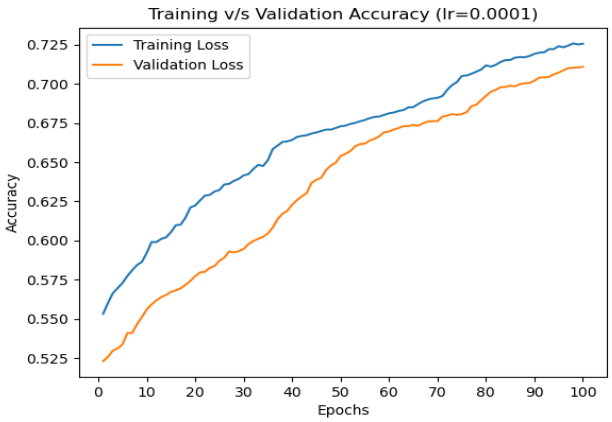

Figure 3. Pixel Accuracy (Learning Rate: 1e-4)

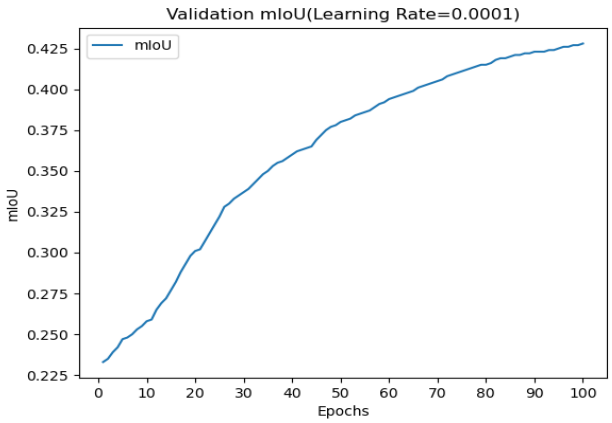

Figure 4. mIoU (Learning Rate: 1e-4)

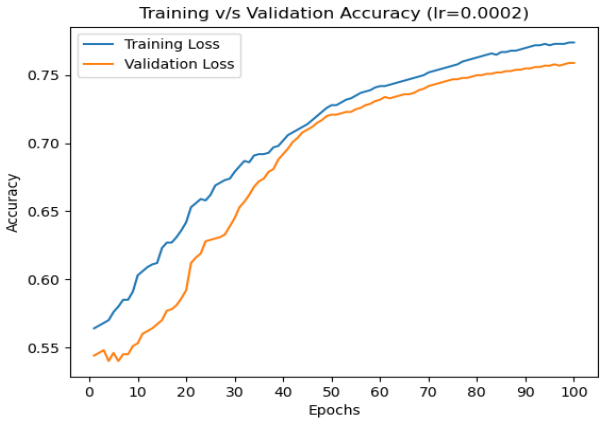

Figure 5. Pixel Accuracy (Learning Rate: 2e-4)

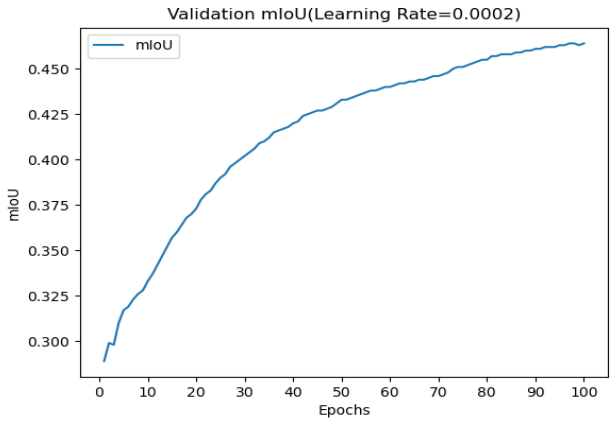

Figure 6. mIoU (Learning Rate: 2e-4)

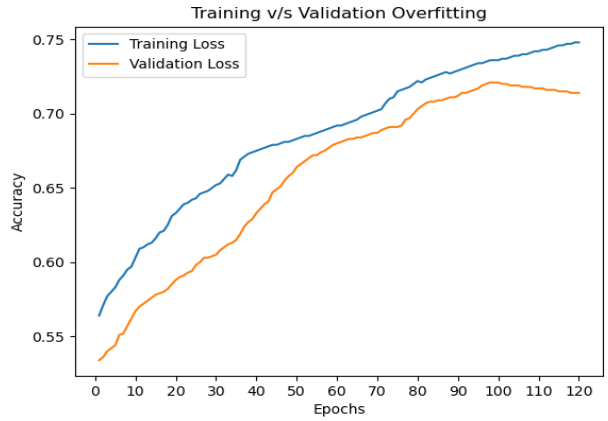

Figure 7. overfitting to training data on longer training with limited dataset

3.2.2 Overfitting Mitigation:

Since we used a small dataset [Pascal VOC 2012][4], we implemented a strategy of image augmentation to avoid overfitting. When we trained for many epochs with the training data we saw the accuracy kept increasing (though marginally). Bur yielded poor results during Validation and Tests. We mitigated this issue by implementing image augmentation (flips, rotations and skewing the image) to generate more training dataset and carefully watching the learning plots for training and validation and effectively fine-tuning the hyper parameters.

3.2.3 Batch Size Adjustment:

Balancing the compute resource, hardware constraints and the need for efficient training processes we carefully adjusted the training batch size. With a single GPU of limited memory, we ran into issue of memory overflow when training on the GPU device. When training on the CPU we

```
10/3660 || Lr: 0.000200 || Loss: 1.4516 || Cost Time: 0:02:09 || Estimated Time: 13:08:06
20/3660 || Lr: 0.000199 || Loss: 1.5399 || Cost Time: 0:02:12 || Estimated Time: 12:08:13
30/3660 || Lr: 0.000199 || Loss: 1.8574 || Cost Time: 0:02:15 || Estimated Time: 11:27:03
40/3660 || Lr: 0.000198 || Loss: 0.9759 || Cost Time: 0:02:18 || Estimated Time: 11:02:12
50/3660 || Lr: 0.000198 || Loss: 0.4545 || Cost Time: 0:02:20 || Estimated Time: 11:01:06
```

Figure 8. Training time of 12 hours with one GPU *Nvidia Quadro RX4000, with batch size 16

encountered very slow training time. We tried to balance the issue by training with the batch size of 16 on the GPU device which was the optimal batch size for our experiment.

3.3. Reference to External Data Points:

Leveraging External Knowledge: In crafting our experimental design, we drew upon insights from relevant research papers.

1) Fully Convolutional Networks for Semantic Segmentation [[https://doi.org/10.48550/arXiv.1411.4038]: We did a comparative analysis of our model performance with the Fully Convolutional Network for Semantic Segmentation model, especially comparing notes with the implementation of the FCN[3] in https://github.com/Tramac/awesome-semantic-segmentation-pytorch We found that even with the limited compute power and resources we were able to achieve comparable pixel accuracy and mIoU.

2) GitHub [https://github.com/Tramac/awesome-semantic-segmentation-pytorch]: We referenced the implementation of various competing models in the Image Segemnation task to gain insights on the metrics used, performance of the competing models such as FCN[3] and gain insights on the Compute and Hardware resources we may require.

3) Image Segmentation Using Deep Learning - A Survey[5] [https://arxiv.org/pdf/2001.05566.pdf]: We also referenced several external studies conducted by researchers to compare the effectiveness of our SegNet model. We would like to draw attention especially the following comparative study: Image Segmentation Using Deep Learning: A Survey [https://arxiv.org/pdf/2001.05566.pdf]. We found that our model implementation yielded a moderate result when compared to other model performances surveyed in this research paper.

4) GitHub [https://github.com/say4n/pytorch-segnet]: We also referenced and reused the Pascal VOC 2012 dataset loading implementation for the above GitHub resource.

4. Summary

4.1. Achievements:

Our experiment yielded a comparable pixel accuracy of above 0.79 and mean Intersection over Union (mIoU) of 48, reflecting the success of our model in accurately classifying pixels within the segmentation task. This achievement, within the context of limited resources, demonstrates the efficacy of our chosen methodologies.

4.2. Limitations:

We recognize the limitations tied to compute resources and training time. To mitigate we adopted a pragmatic stance, acknowledging the real-world challenges associated with Deep Learning experiments.

4.3. Potential Improvements:

Looking forward, we recognize the opportunity for improvement through the selection of more diverse training data and an increase in the number of training epochs, enabled through more compute power and time for training.

4.4. Key Learnings:

Dataset Selection: We understand the importance of selecting a proper dataset with balanced and diverse class objects. We understand the foundational role that data plays in the success of deep learning models. We had a practical understanding of how a small dataset can lead to overfitting and the practical approach to address this issue.

Resource Considerations: Considering the resource implications, we grasp the challenges associated with training Deep Learning models on large datasets. This project not only afforded us a glimpse but a firsthand understanding of the challenges inherent in the training of sophisticated deep models.

Training Observations: This experiment provided us with a comprehensive understanding and perspective on various parameter tuning techniques, offering us an in-depth experience in managing the challenges associated with the tuning process.

References

[1] Vijay Badrinarayanan. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. 2016. 1, 2

[2] PyTorch Consortium. Pytorch library. 2, 3, 4

[3] Trevor Darrell Jonathan Long, Evan Shelhamer. Fully convolutional networks for semantic segmentation. 2015. 1, 5

[4] Univ of Oxford. Pascal voc 2012 dataset. 2012. 1, 5

[5] Fatih Porikli Antonio Plaza Nasser Kehtarnavaz Shervin Minaee, Yuri Boykov and Demetri Terzopoulos. Image segmentation using deep learning: A survey. 2020. 5

[6] Karen Simonyan. Very deep convolutional networks for large-scale image recognition. 2015. 2

[7] Hans Thisanke. Semantic segmentation using vision transformers: A survey. 2023. 1

[8] Jianhua Yang. Cmf: Cascaded multi-model fusion for referring image segmentation. 2021. 1

Student Name	Contributed Aspects	Details
Paresh Upadhyay	Pascal VOC2012 Data identification	Identoified suitable dataset for the model
Paresh Upadhyay	Data Loader and parser implementation	Implemented code
Murugan Viswanathan	SegNet Model implementation	Implemented code
Paresh Upadhyay	SegNet Model implementation	Implemented code
Murugan Viswanathan	Training, Validation code implementation	Implemented code
Paresh Upadhyay	Metrics (Pixel Accuracy, mIOU) implemenataion	Implemented code
Murugan Viswanathan	Setup of GPU, Hardware infrastructure	Setup GPUs, Hardware for speeding up training
Murugan Viswanathan	Performing Training on Train and Validation Datasets	Training and validation of the model
Paresh Upadhyay	Work on balancing Long Training time and Model Accuracy	Monitor training progress and adjust parameters train
Murugan Viswanathan	Parameter Tuning and working on improving Model Accuracy	Workied on tuning model parameters for optima mance
Paresh Upadhyay	Parameter Tuning and working on improving Model Accuracy	Workied on tuning model parameters for optima mance
Murugan Viswanathan	Mitigating Overfitting	Mitigate overfitting by applying various data au tion
Murugan Viswanathan	Researching on other models for comparative analysis	research for comparitive analysis
Paresh Upadhyay	Researching on other models for comparative analysis	research for comparitive analysis
Murugan Viswanathan	Paper writeup	Contributeed to create draft, final writeup
Paresh Upadhyay	Paper writeup	Contributeed to create draft, final writeup
Murugan Viswanathan	Generating Inference Images	Contributeed to create inference images for t writeup
Paresh Upadhyay	Generating Metrics Plots	Contributeed to create plots for the final writeup
Murugan Viswanathan	Code review	Review code for potential issues and bugs
Paresh Upadhyay	Code review	Review code for potential issues and bugs
Murugan Viswanathan	Reviewing Paper writeup	Review draft and final Writeup
Paresh Upadhyay	Reviewing Paper writeup	Review draft and final Writeup
Murugan Viswanathan	Error Analysis and mitigating bottlenecks in implemenation	Contributed to debugging
Paresh Upadhyay	Error Analysis and mitigating bottlenecks in implemenation	Contributed to debugging

Table 1. Contributions of team members.

APPENDIX

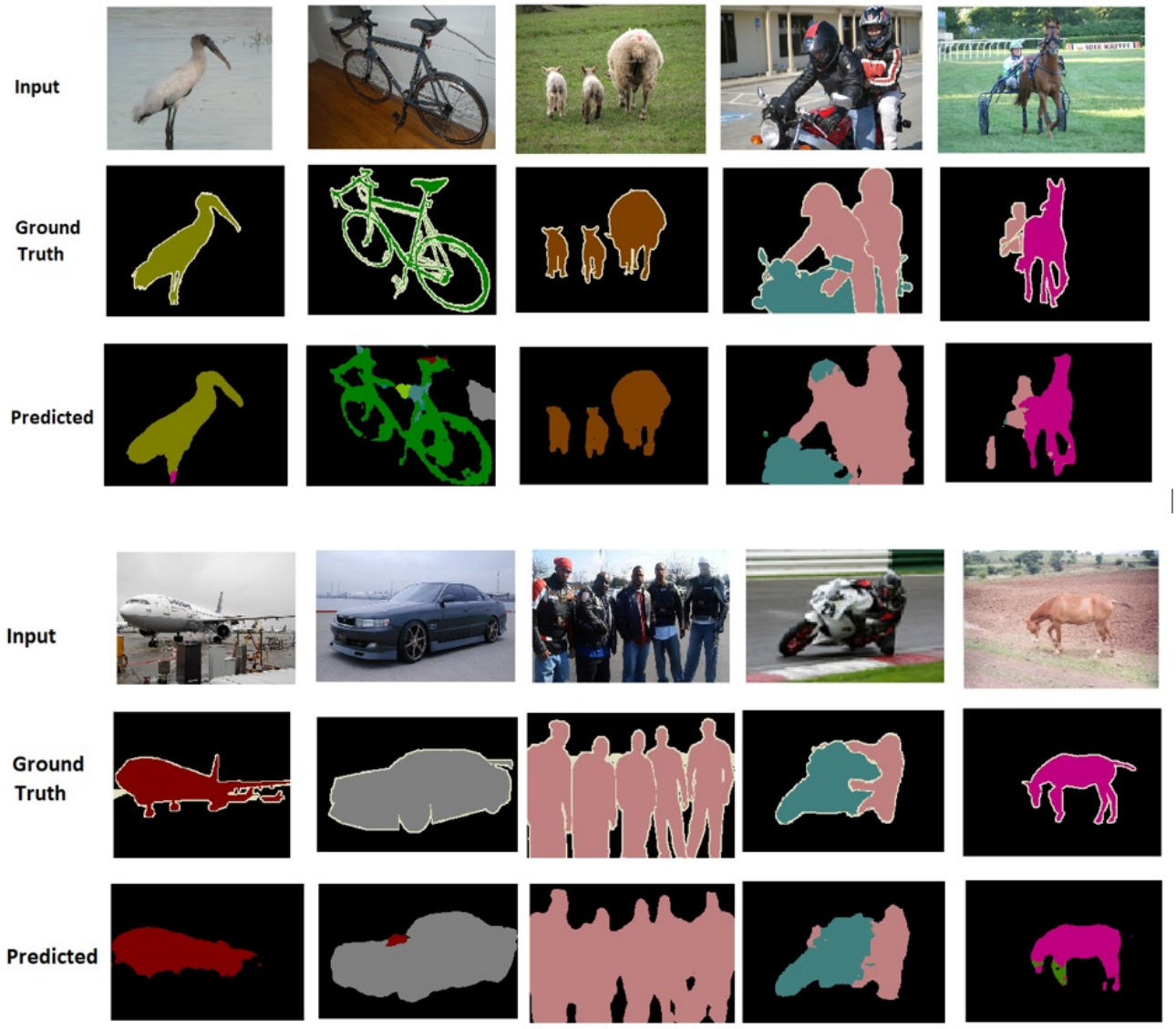

Figure 9. Images, Ground Truth, Prediction