Semantic segmentation using full convolutional network

Fangzhou Ai
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093
faai@eng.ucsd.edu

Yue Qiao
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093
yuq021@eng.ucsd.edu

Zunming Zhang
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093
zuz008@eng.ucsd.edu

Abstract

TBA

1 Introduction

2 Related Work

3 Methods

3.1 Baseline

3.2 Experimentation

4 Results

4.1 Baseline model

4.2 Improved baseline model

4.2.1 Dataset Augmentation

From the baseline model, we tried to do some data augmentation on the training dataset to see whether the performance could be approved. Since training is quite time-consuming, in this report, we implemented and tested two different data augmentation algorithm and tried to compare the performance with each other.

Preprint. Under review.

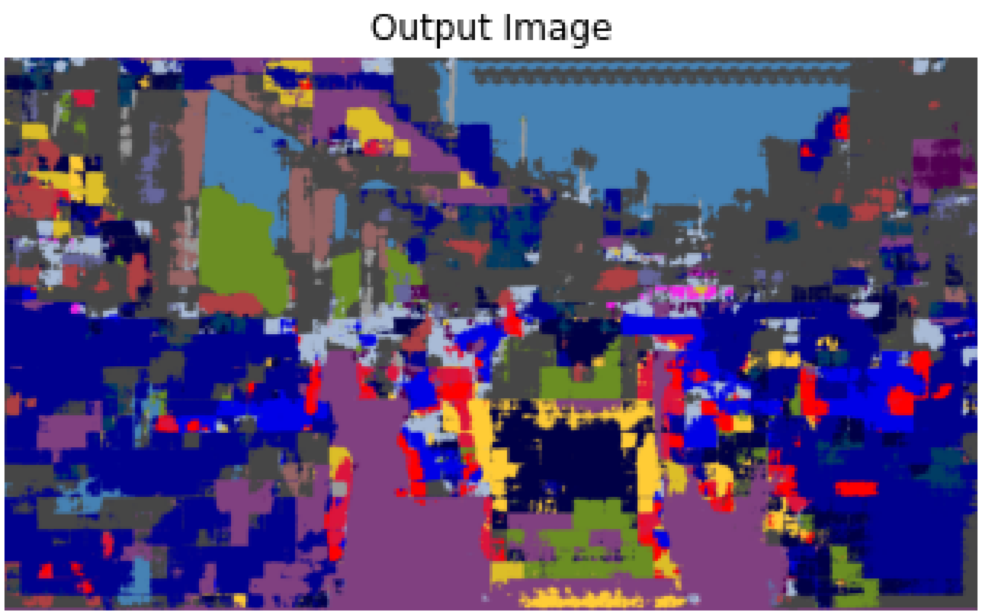

Figure 1: Visualization of segmented output for Mirror flip augmentation method

The data augmentation methods used are:

- Mirror flip: for the training dataset, we mirror flipped the dataset and fed them into the model
- Image Rotating: rotate the images slightly (for a certain degree), in this report, we tried to rotate the images by 1 degree and then fed them into the model

Table 1 shows the model performance of different augmentation method. From the table, we could see that the performance was improved by 10% simply by rotating the training dataset for a small angle.

Table 1: Comparison of performance of different augmentation methods

	Mirror flip	Rotation
Validation set pivel accuracy	0.6953	0.7936
Average IoU	0.2313	0.3393
IoU for road(0)	0.7477	0.8577
IoU for sidewalk(2)	0.0874	0.2483
IoU for car(9)	0.2262	0.4857
IoU for billboard(17)	0.2542	0.2755
IoU for sky(25)	0.8105	0.9445

The visualization of the segmented output for the first image in the test.csv overlaid on the image for mirror flip and rotation method could be seen in Figure 1 and Figure 2, respectively.

4.2.2 Imbalanced class problem

For the imbalanced class problem, what we usually do is to pressuring the network to categorize the infrequently seen class. We proposed to use the weighted loss method to accommodate this issue, which will weight infrequent classes more.

For the weighted loss, the idea is to set the weight of the loss of each class inversely proportion to the frequency of each class in the training dataset. For the data pre-processing, we calculated the total

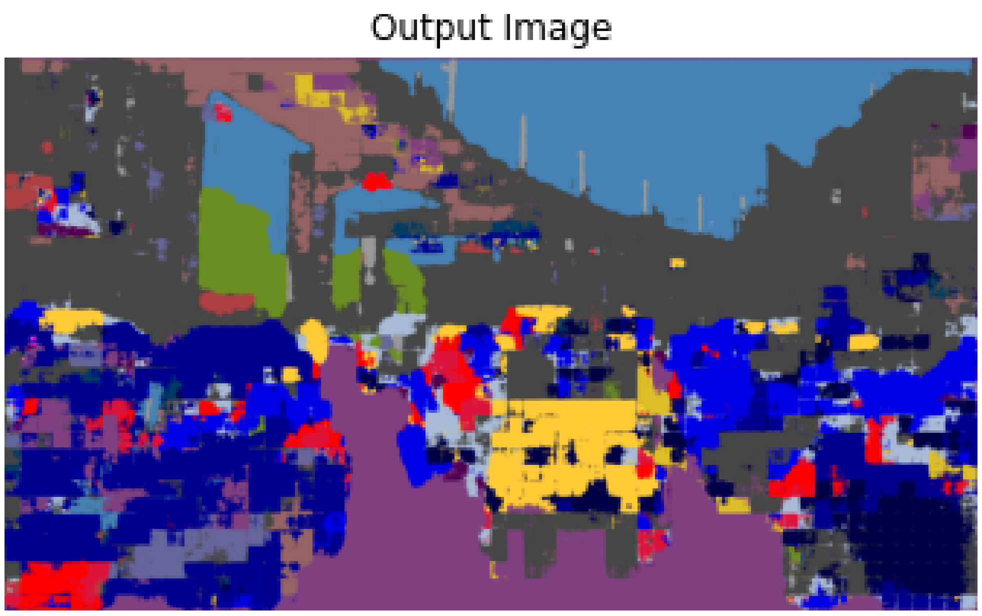

Figure 2: Visualization of segmented output for Rotation augmentation method

pixel count of each class in all the training dataset and then use this information to help us set the weight of the loss.

Suppose, the number of samples for each class is stored in nSamples, then we use $normedWeights$ = [1 - (x / sum(nSamples)) for x in nSamples] as the normalized weight of the loss for each class.

For this dataset, we calculated the number of samples for each class is:

$nSamples = [2198453584, 519609102, 28929892, 126362539, 58804972, 59032134, 94293190, 2569952,$

$101519794, 163659019, 105075839, 47400301, 30998126, 133835645, 135950687, 41339482, 15989829,$

$104795610, 8062798, 450973, 94820222, 341805135, 557016609, 71159311, 1465165566, 1823922767, 2775322]$

Then we could be able to set the weight of the loss for each class according to nSamples.

The performance of the model could be seen in Table 2 and the visualization of the segmented output for the first image in the testdataset overlaid on the image could be seen in Figure 3

Table 2: Performance of model after introducing the weighted loss for the imbalanced dataset

	weighted loss w/ rotation
Validation set pivel accuracy	0.6953
Average IoU	0.2313
IoU for road(0)	0.7477
IoU for sidewalk(2)	0.0874
IoU for car(9)	0.2262
IoU for billboard(17)	0.2542
IoU for sky(25)	0.8105

4.3 Custom model

Inspired by this paper, we replace the encoder part in FCN model with the RRCNN layer instead, the RRCNN represent better abstraction ability, thus we would expect better IoU and precision. However the RRCNN is really memory-intensive, we have to crop the image to only 256 *256 size and delete

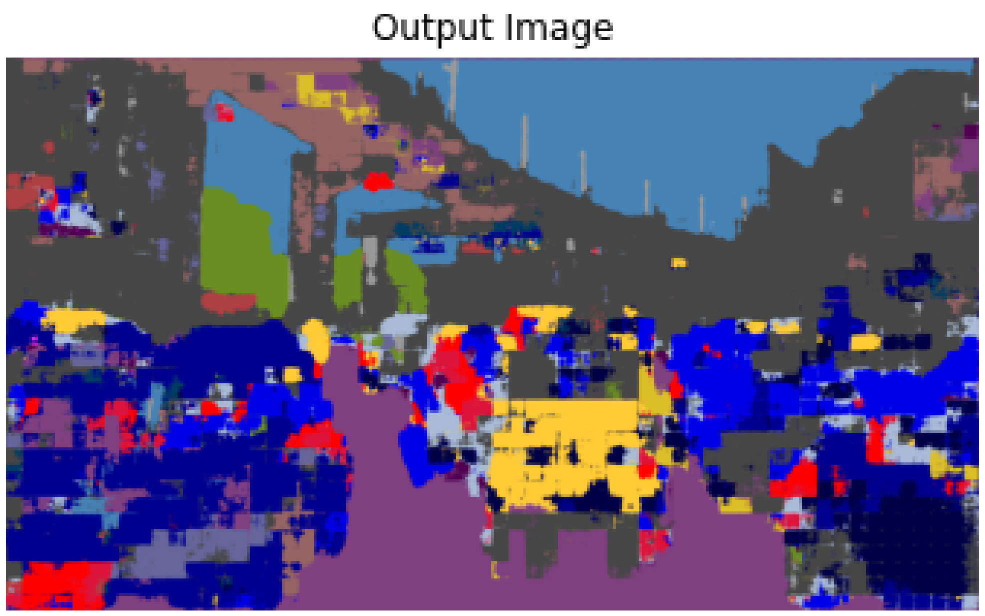

Figure 3: Visualization of segmented output after introducing weighted loss method

3 layers from original model's encoding part, which severly influence the performance of our model. What's worse is the RRCNN encoding part contains maxpooling operation after each RRCNN block, which means the images size would shrinks even more, since we have already started from a relative small size, maxpool will almost ruin all the details in the feature map.

4.4 Transfer learning

In this part we employ the DEEPLABV3-RESNET101 model, which is a default pre-trained net work in PyTorch, from the official document we know that this model is constructed by a Deeplabv3 model with a ResNet-101 backbone. The pre-trained model has been trained on a subset of COCO train2017, on the 20 categories that are present in the Pascal VOC dataset. We modifier the classifier part so that it could match the number of our feature, and we freeze the parameters of the pretrained part to accelerate the whole procedure.

4.5 U-Net

Folowing the routine demostrated in the U-net paper, we build the U-net by our self. The obvious difference is the U-net convolution block contains 2 convolution of the same size, while the FCN model only contains 1 convolution operation. We notice that the U-net model is also memory intensive while the datahub only give us a GTX 1080 ti with 11GB memory, hence there's a trade-off of between batch size, image size and model's completeness. We decide to crop the image and shrink the batch size to run the whole model, though this might not be the best solution, but this could give us a flavor of how the complete network looks like.

5 Discussion

6 Individual contributions to the project

Fangzhou Ai Custom model, Transfer learning and U-Net model.

Yue Qiao Evaluation metrics, Dataset loading and Baseline Model.

Zunming Zhang Improved baseline model with mirror flip, rotate and weighted loss for imbalanced classes