EE599 Project: Fully Convolutional Network With Edge Labels For Semantic Segmentation

Yi Zheng, Ruyi Zhang, Yifan Wang
{zhen030, ruyizhan, wang608}@usc.edu

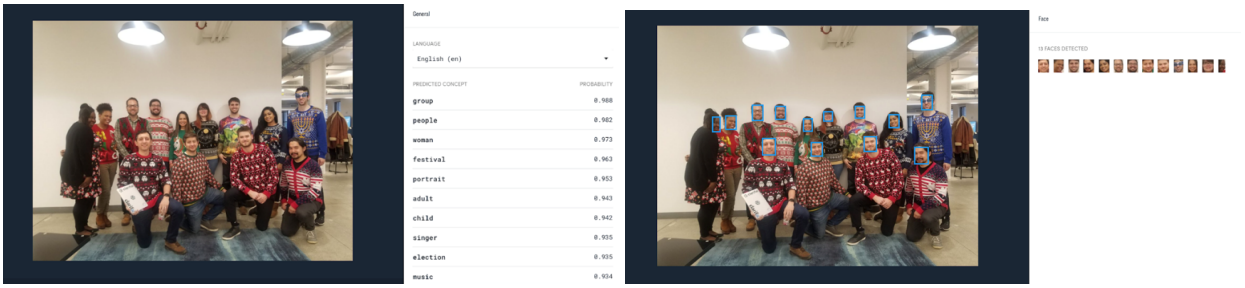
May 7, 2019

Abstract

Image semantic segmentation has been a hot research topic in computer vision field. Fully convolutional neuron network [4, 5] has achieved astonishing performance since its publish. In this project, we aimed on to increase it's performance by modify the loss function with both pixel segmentation result and edge labels based on ResNet50 architecture with atrous convolution. After some training on both original network and the one with additional edge loss on both Pascal VOC [6] and some images taken from streets, we observe increase in segmentation results.

1 Introduction

Before starting image semantic segmentation, we first have a brief comparison at image classification, object detection and image semantic segmentation.



(a) (b)

Figure 1: Example of object detection

Image classification is a basic technology in computer version, which is referred to assign an image to a certain label or some labels with probabilities from a set of labels. Classification just tells what the picture is at a global view and it does not focus on the specific part of an image. Below we can see an example of image classification in $Figure1(a)$ [1]. The image is labeled as group with probability 0.988 and as people with probability 0.982, etc. This is also the main part we have done in the EE599 class this semester.

Object detection is focused on the distinguishable objects in an image. Usually, objection detection will give us results including labels, probabilities and bounding boxes. We can the difference between classification and detection in the same picture with different technologies. Below in $Figure1(b)$ is the same picture shown in $Figure1(a)$ using a facial detector [1]. As we can see, this time the technology focused on the faces in the image. It labels all the faces with bounding boxes.

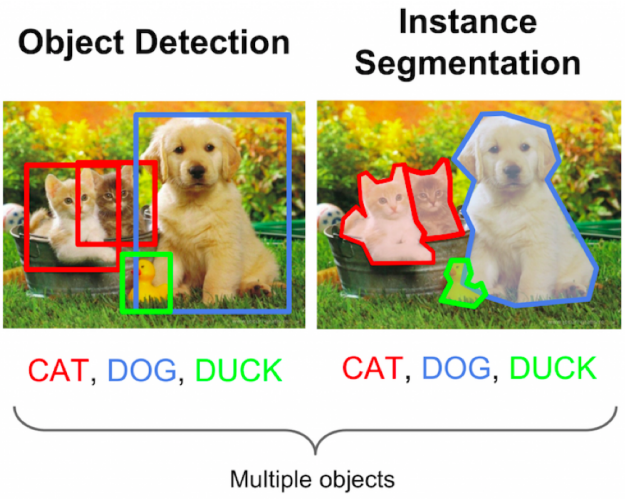


Figure 2: Comparison of detection and semantic segmentation

Image semantic segmentation is a technology focused on the pixelwise classification. It assigns each pixel in an image to a certain label in a set of labels. To some degree, we can treat semantic segmentation as a kind of detection with more accurate bounding boxes. Below we can see the difference in an image by using both technologies in $Figure2$ [1]. We can see segmentation can have more accurate edges with different objects because segmentation labels every pixel in the image.

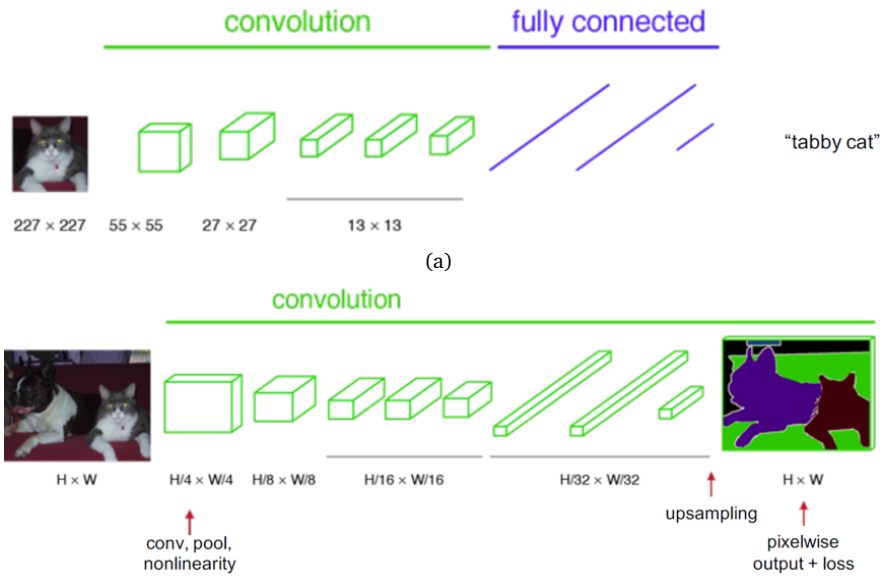


Figure 3: Structure of CNN and FCN

Image segmentation has wide application in field like auto-piloting cars which has been a hot topic for recent years and many other fields. Attention has been used in segmentation tasks to reduce disturbance from background [14] as well.

This part we will see more details of FCN. To have a better understand, we can compare CNN we learned in class with FCN. In CNN, we usually do the image classification by using convolutional layers and pooling layers to downsample the image and to extract the deep features in the image. After that, we concatenate fully

connected layers to finish the classification. Finally, the output will be a label or some labels with probabilities. In $Figure3(a)$, we can see a simple example of CNN [2]. Input is a picture and output is a label. And because of the dense layer, the size of input is fixed.

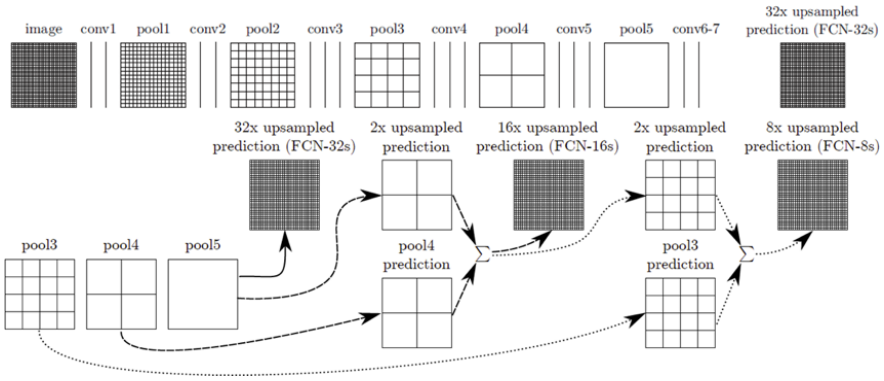


Figure 4: Basic structure of skip

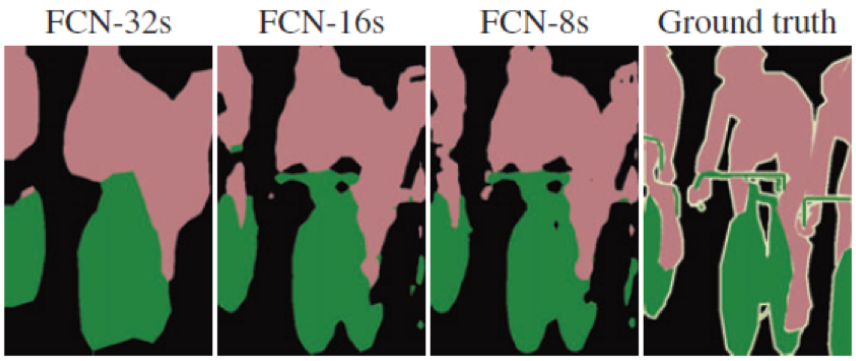


Figure 5: Results of skip

As for FCN, in simple terms, we replace the dense layers with convolutional layers so all layers in the network are now convolutional layers. After that, we concatenate upsample layers to keep the size of output same with input. This is the basic structure of FCN. And compared with the CNN, similarly we extract the deep feature by using convolutional layers and pooling layers with loss of spatial coordinates. Besides, we also use upsample layers to recovers the spatial information. Finally, the output of FCN is a pixelwise prediction. Further, without dense layers, the size of input in FCN can be any size and the number of parameters can be significantly reduced. Below in $Figure3(b)$ we can see an example of FCN. Input is a picture and output is a pixelwise mapping in same size.

However, although we use the upsample layers like bilinear interpolation, the loss of spatial information can still not be fully recovered. Thus, generally the output is dissatisfyingly coarse. To deal with limitation of details, we also use a skip structure to fuse the layer outputs. In general knowledge, when we go deeper in convolutional layers, we can get deeper features and lose more spatial information. In other words, the shallower layers keep more spatial information. Thus, we can combine the outputs from deeper layers and shallower layers, which is the key concept of skip structure. Below in $Figure5$ is basic structure of skip and $Figure5$ is the results. The upsample output without any skip structure is called FCN-32s. we upsample pool 5 (1-by-1 convolutional output) with stride 2 and combine it with pool 4 (2-by-2 convolutional output in shallower layer) by pixelwise addition. After that we upsample the sum by stride 16, which is called FCN-16s. Similarly, combining more shallower layers we can get FCN-8s. From the results in $Figure5$, we can see that after skip the results have a significant improvement.

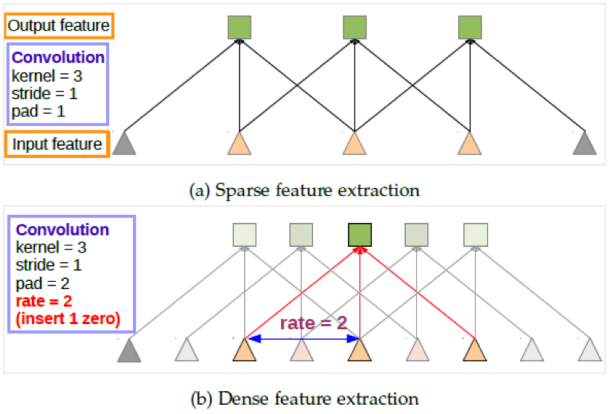


Figure 6: 1D example of artous convolution

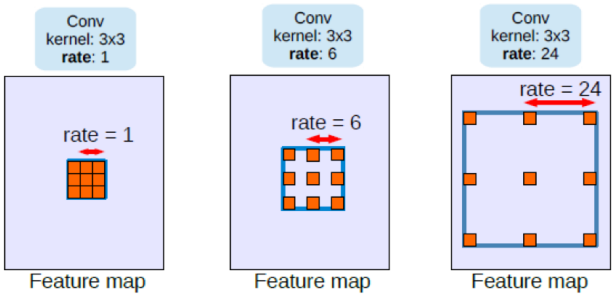


Figure 7: 2D example of artous convolution

In this part we will introduce atrous convolution. In the basic FCN, we use standard convolution to get feature map. In general, to get a good feature map we need to downsample the image to a very small size. Because the downsample process is a loss compression and upsample with stride 32 is really aggressing, the results are not that ideal. Although we can use the skip, it is not enough to solve the problem perfectly. Thus, in our project, we use atrous convolution in our FCN. The basic formula is shown below, where r is the rate [3]. When $r = 1$, it is the standard convolution formula; when , it is the artous convolution.

$$y[i] = \sum_k x[i + rk]w[k]$$

Here we can see 1D and 2D examples in $Figure6$ and $Figure7$ [3].

In 1D case, when kernel $size = 3$ and $stride = 1$: in standard convolution, we just need to pad 1 zero in each side and the size of output is 3, same with the input; however, in artous convolution $rate = 2$, we need to pad 2 zeros each side and insert 1 zero as well and the size of output is 5, larger than the input. Similarly, in 2D case, we can also get larger output size by using artous convolution. In other words, by adjusting rate, we can increase the FOV (filed-of-view) of filter to capture more content. Further, when calculating, we just need to consider the non-zero values of filter. Thus, the number of parameters is same compared with standard convolution. In $Figure8$, we will see the performance of artous convolution and standard convolution.

The 1st row shows the standard convolution: we downsample the image with stride 2 and do the convolution with kernel $size = 7$ and after that we upsample the output with stride 2. Finally, we can see the loss in the output. The 2nd row shows the artous convolution: we use the same kernel size and do artous convolution with $rate = 2$, which is similar with the procedure that we first upsample the kernel with stride 2 and do standard convolution. We can see that the performance of artous convolution is much better than the standard convolution.

Image segmentation has wide application in field like auto-piloting cars which has been a hot topic for recent years and many other fields. Generally, it would face several problems like segmentation accuracy, background disturbance, misclassification etc. Fully convolutional neuron network is first introduce in 2014 [5] with huge improvement in accuracy on semantic segmentation tasks. It uses some convolutional layers to replace fully connected layers in network like VGG16 [9]. Due to the fact that convolutional layers can keep spacial information then using a bilinear interpolation to reconstruct segmentation labels for each pixel. This idea is quickly applied on other network structures like ResNet [10]. Attention has been used in segmentation tasks to reduce disturbance from background [14] as well.

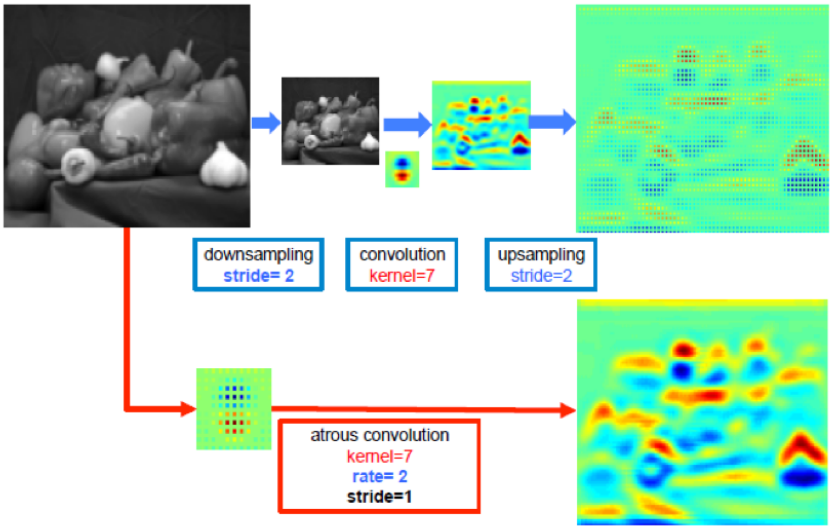


Figure 8: Performance between standard convolution and artous convolution

Up until now, the state of art result is achieved by deeplab [11–13] from Google and UCLA. In this series of work, they used probability to represent likelihood of each pixel belongs to particular class, then using a iterative CRF to generate the final segmentation mask. They made some improvement on convolutional layers as well. In previous FCNN, in order to have a large perceptive field, input images are reduced into very small scale before they are being upsampled to input shape. However, the reduce produce is a loss compression where some information useful to final pixel-wise prediction is lost. They used a atrous convolution to enlarge perceptive field. In that case, input images do not need to be reduce to small scales while maintain perceptive filed. Less information is lost in this process, so that this network give better performance than previous work.

In this project, we based on a ResNet50 architecture aimed to design a end to end structure which can improve segmentation accuracy especially on object boundaries. Train and test is using Pascal VOC [6], some additional images taken by ourselves are used as well to find out edge loss's effect.

2 Methods

In Pascal VOC dataset [6], there are 20 classes of object and background, besides, for better visualization of each object each boundaries are labeled as white.

In Pascal VOC dataset [6], there are 20 classes of object and for each pixel indices, it corresponds to one of these 20 classes in alphabetical order(1=aeroplane, 2=bicycle, 3=bird, 4=boat, 5=bottle, 6=bus, 7=car , 8=cat, 9=chair, 10=cow, 11=diningtable, 12=dog, 13=horse, 14=motorbike, 15=person, 16=potted plant, 17=sheep, 18=sofa, 19=train, 20=tv/monitor). In addition, the background is labelled as 0 and each boundaries are labeled as white with index 255. The example training data are as below,

When using this dataset in training, these boundaries are ignored without using them for any computation which means any prediction in these pixels would not affect the value of loss function. To some extent it

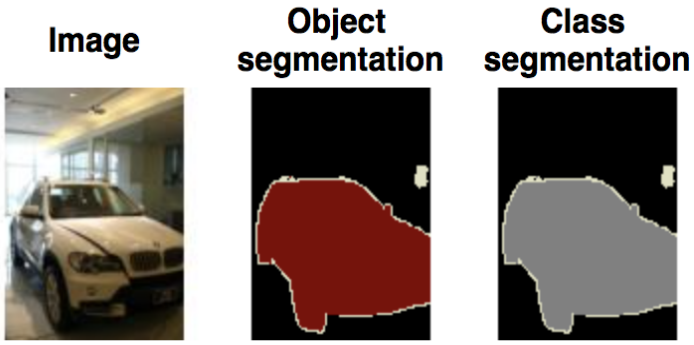


Figure 9: Example training image.

means some uncertainty would occur on boundaries. To increasing accuracy of segmentation result, we here includes these edge information into our loss function as a fine tuning method to get accuracy segmentation result along with traditional sparse loss which ignores last label (L_{spase}).

To compute edge loss, it is relative easy to get edges from training labels, they are the last class of one-hot labels (22 one-hot class = 20 objects + 1 background + 1 edges). Let's name true edge labels for image I as $E_{I_{true}}$ For a predict mask, suppose input image I has height H, width W. Activation mask a_I has shape $(H, W, 21)$. By simply choose the largest activation index as label for this pixel, a prediction mask P_I can be derived from it which has shape $(H, W, 1)$. Where one represents edges and zero means non-edge. Edge mask is computed by whether a pixel's both 4-connected and 8-connected neighbours have same label as it does. If all pixel in neighbourhood are the same, set edge mask for this pixel to be zero else one. After this process, we can get a predicted binary edge mask $E_{I_{pred}}$ for input image I. Edge loss is computed by:

$$L_{edge} = \frac{E_{I_{true}} \odot (1 - E_{I_{pred}})}{\sum E_{I_{true}}} \tag{1}$$

This edge loss is always restricted between zero and one with a hard decision. Compare with total loss during the beginning training stage, this loss is much smaller than L_{sparse} so that it would not affect much. More loss can be added to the network since lots of error occurs on edges as well. When training for a long time, L_{sparse} would get much smaller, so that L_{edge} would help to fine tune prediction mask with relative large loss if edges not matches well. Besides, notice that in our case, $E_{I_{true}}$ is slightly thicker than $E_{I_{pred}}$ we generate that means several edge prediction of same image with slightly pixel different is acceptable. This kind of uncertainty is aimed to add some noise during training stage to avoid stuff like over-fitting from happening.

However, after some analysis, we found if change $Equation(1)$ to be following one would help to make more sense:

$$L_{edge} = \frac{(1 - E_{I_{true}}) \odot E_{I_{pred}}}{\sum (1 - E_{I_{true}}) + \epsilon} \tag{2}$$

Where ϵ is to avoid denominator to become zero. In previous cases, it does not care about whether prediction is

loss		ground truth	
		is edge	none edge
prediction	is edge	0	1
	none edge	1	0

Table 1: Loss at different condition

right on none edge pixels which means that if prediction is that all pixels belongs to edge, loss computed from $Equation2$ would be zero. That is not what we want to see. Some slightly changes are made on $Equation2$

(a) true edge labels (b) predict edge labels (c) pixels used to compute edge loss

Figure 10: Illustration of edge loss

to make it become $Equation2$. According to this equation, all that wrong prediction on none edge part would be punished. One example is shown in $Figure10$ where (a) shows $E_{I_{true}}$, (b) is $E_{I_{pred}}$ and (c) shows wrong predicted pixels that used to compute loss which is numerator of $Equation2$. In this case, this loss would not be restricted between zero and one, if lots of wrong prediction occurs, it would become relatively large. Value of different loss is shown in $Table1$.

3 Experiment

Our project uses a fully convolutional network modified from ResNet50 using atrous convolution to replace part of convolutional layers. Final layer is a bilinear upsampling layer $16s$ to reconstruct the segmentation result as input size. Pascal VOC dataset is used for training and validation. Loss function uses traditional pixel-wise loss and the one with the additional edge loss are both test to compare new loss's effect with the same amount of training time. There is an intuition that relative large loss would introduce more unstable factor while training. While our test result shows it do have some side effect, however, final result remains pleasing. When combining our new loss with old loss, no weight is used. This could be hyper-parameter that can be explored more.

Test results shows using edge loss can slightly increase meanIOU by about 1.5% ($Table2$). We are expected to see a greater improvement in these numbers if training more epochs or find some suitable weight. Since this edge loss mainly serves as a fine tuning method which might give a much better performance after more training. In original code, they trained it for 500 epochs. However, due to the fact that we have made lots of other trials on other ideas and methods, we can not afford both the time and cost to train this network for a longer time for this project. But this result remains appealing.

	Without edge loss	With edge loss
meanIOU	0.5613	0.5760
pixel acc	0.8911	0.8970

Table 2: Accuracy when train both network for 30 epochs


(1)


(2)


(3)


(4)

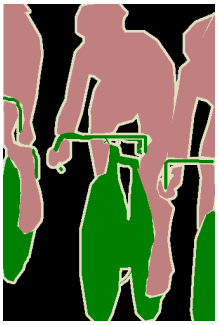

(5)


(6)


(7)


(8)


(9)


(10)


(11)


(12)


(13)


(14)

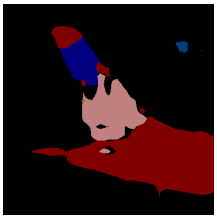

(15)


(16)

(17)


(18)


(19)


(20)


(21)


(22)


(23)

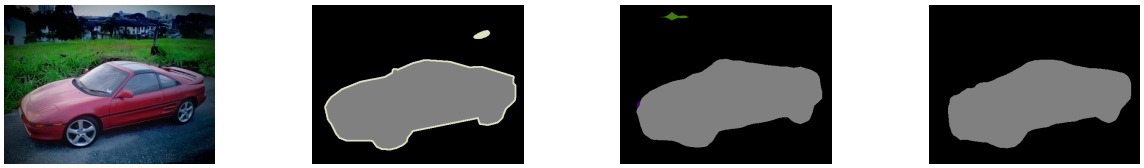

(24)


(25)


(26)

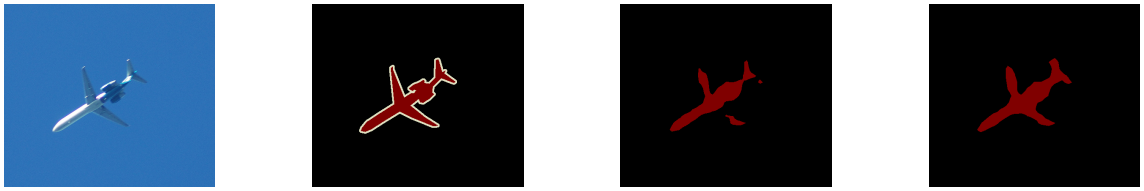

(27)


(28)



(29) (30) (31) (32)

(33) (34) (35) (36)



(37) (38) (39) (40)

(41) (42) (43) (44)

(45) (46) (47) (48)

(49) (50) (51) (52)

(53) (54) (55) (56)

(57) (58) (59) (60)

Figure 11: Segmentation result of original model and the one equipped with edge loss
$1^{st}col$: original image; $2^{nd}col$: ground truth; $3^{rd}col$ column: without edge loss; 4^{th} column: with edge loss

Some of success result are shown in $Figure11$. Segmentation results are more accurate with the help of edge loss. More detail separation can be derived. For example, image with two person on motorcycle ($Figure11(49-52)$). In original network's result, the driver's hand is not well segmented, their bodies are connected. With the help of edge loss, a slightly better result is produced on bodies and hands. However, there remains some issues on mirror which have super thin connection with motorcycle and the back of motorcycle are not segmented well. Other cases like whole airplane, segmentation result is much better. Most successful results have following characteristic in common:

- relative simple boundaries;
- no complex texture on objects;

Both networks work nicely when dealing with image like $Figure11(25, 29, 33, 37)$ where only a single object appear on a clear background. The one with edge loss can produce segmentation result with more accuracy. For example, the edge of airplane becomes more resemble to ground truth. Some of pixels on objects not recognized in old network are classified with the help of new loss.

However, if object with complex texture on image like the tail of air plane ($Figure11(17-20)$). The network trends to separate them into other objects. Especially when such texture can be regard as another object class, wrong classification happens. Otherwise, it would be regard as background. After adding edge loss, these part can be separated even though some wrong classifications would happens as well. We also make some fancy test like this in the end of our experiment.

Another case when the network trend to fail is when objects have lots of boundaries like $Figure12(1, 5)$, they can be regard as a kind of failure in both original network and network with edge loss. The plant or chair in these images is so dedicated that they can not be segmented well. Other objects like the can or human can be found have some increase accuracy as above mentioned. This shows that both networks can not perform well on object with lots of thin part which verified our observation from the successful segmentation result. Only object with relative simple shape and texture can be segmented well.

It is really a failure when dealing with cases like $Figure12(13)$. Even though there is one object on image, it have super fancy texture which is almost the same as the background. Both network can not perform well especially on neck part which is regarded as background in both cases. Network with edge loss even performs worse. It happens the same in $Figure12(17)$ where the bicycle have lots of detail ground truth, both network do not work well. They even regard bridge as another objects even though it is not listed on ground truth.

We run both trained network on images we taken in our daily life, as well. Some results are shown in $Figure13$. These test confirms our initial motivation that by adding edge loss we can have more accurate segmentation results. For example, in $Figure13(34-36)$, it can be found three pigeons are separated in original image. The one with edge loss can almost separate these birds nicely. On the other hand, no edge loss would connect the two pigeons which are quite closed together. This can be found in many other cases where more accurate result is achieved. However, when comes to trees $Figure13(4-6)$ both network may not perform well. They only give predication to part of tree which is the same case as the above test from Pascal dataset. Besides, image quality also matters. If using images with blurred objects, segmentation result would drop dramatically. This case trends to happen when performing real time video segmentation in low light conditions.

Some fancy cases like how network world perform when a object have texture of another object are test in our project. From $Figure14$ where we take photos of two bottles and computer screen which shows the

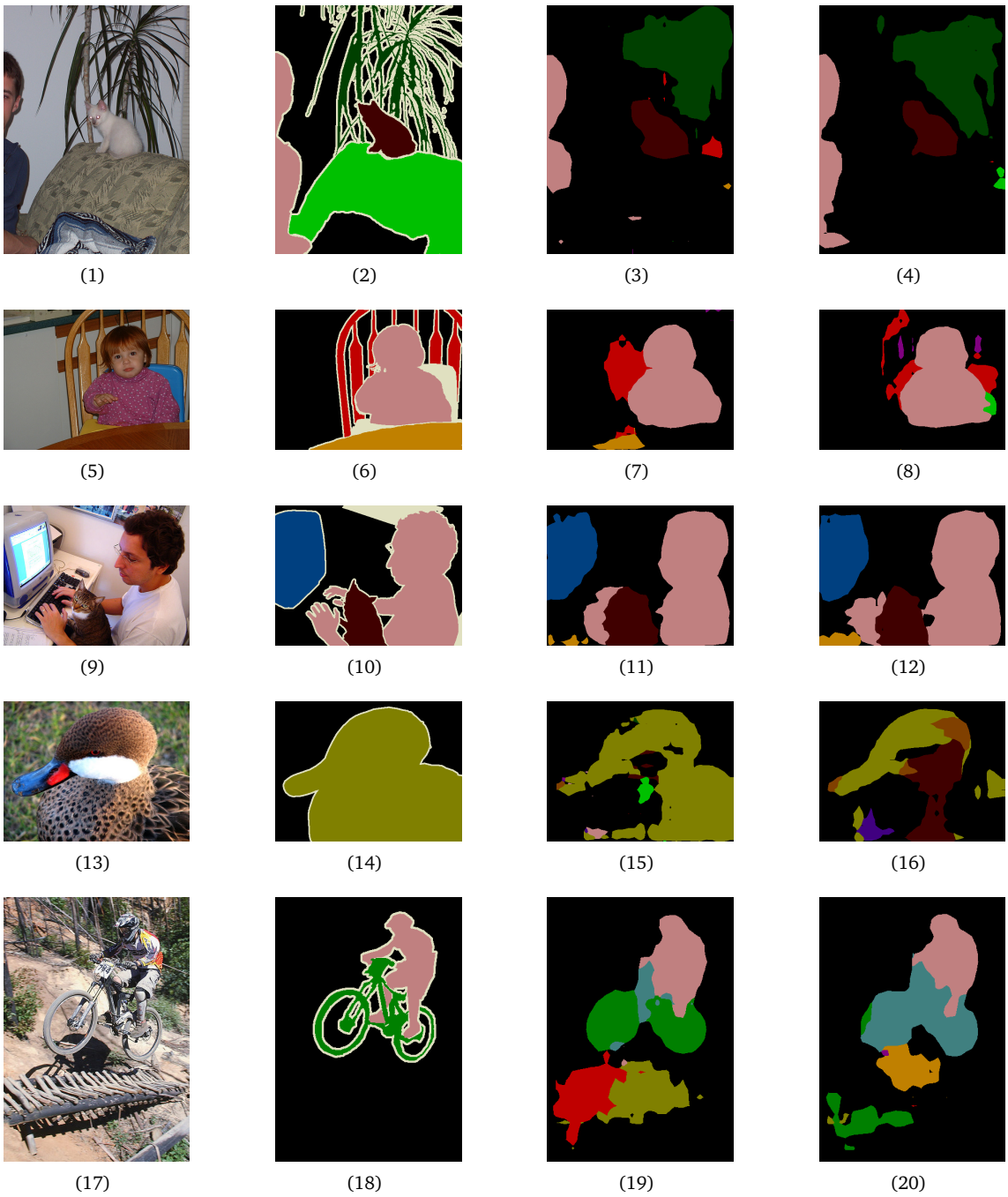


Figure 12: Other segmentation results of original model and the one equipped with edge loss
$1^{st} col$: original image; $2^{nd} col$: ground truth; $3^{rd} col$ column: without edge loss; 4^{th} column: with edge loss

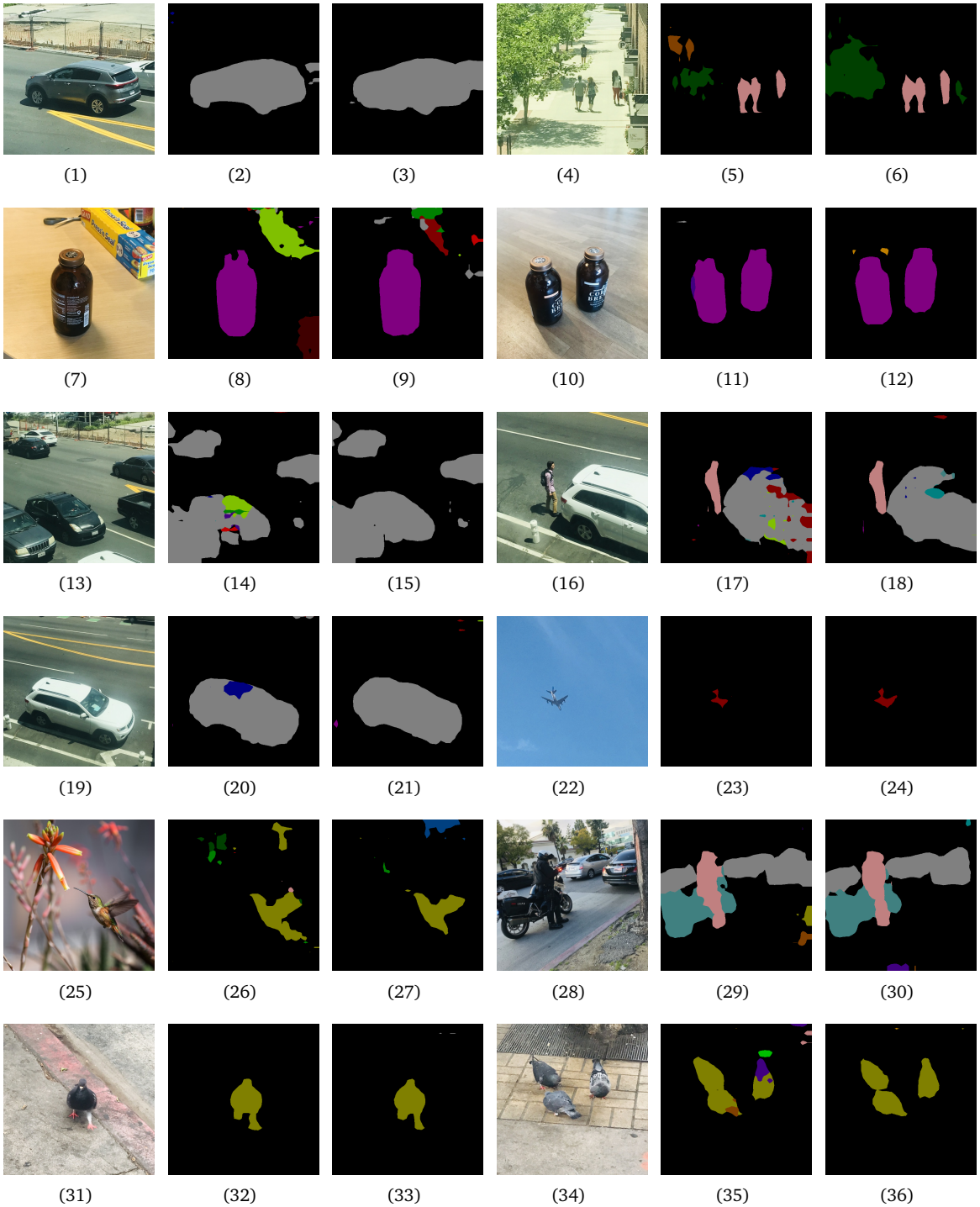
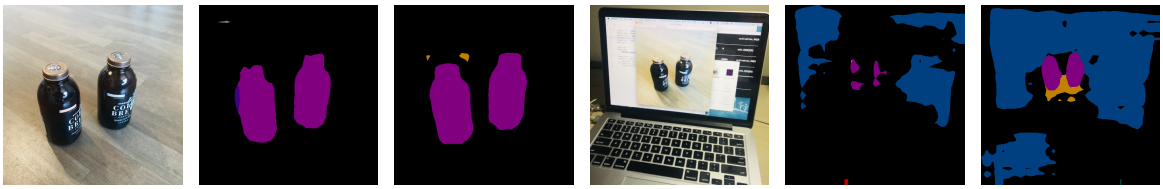


Figure 13: Real world segmentation test results of original model and the one equipped with edge loss
$1^{st}\&4^{th}col$: original image; $2^{nd}\&5^{th}col$: without edge loss; $3^{rd}\&6^{th}col$: with edge loss



(1) (2) (3) (4) (5) (6)

Figure 14: Real world segmentation test results of original model and the one equipped with edge loss and the same image show on screen $1^{st}col$: original image; 4^{th} column: image on screen; $2^{nd}\&5^{th}col$: without edge loss; $3^{rd}l\&6^{th}col$: with edge loss.

same image. It can be found that both network can give nice performance when just process the bottle image (the one with edge loss perform slightly better). When processing screen image which showing bottle image, the prediction segmentation mask is much worse, the one without edge loss can not separate well, while with edge loss, network trend to recognize more part of screen. However, both network can not tell whether the bottle is shown on screen or they are real one placed in front of the screen. It can be found that this network can not deal with case like this which may restrict its application.

4 Conclusion

Using both edge label and class label would increase segmentation result on Pascal VOC dataset based on ResNet50 with atrous convolution. Segmentation result would be much nicer on boundaries. More training epochs are needed to find out how much improvement this edge loss can bring to final result.

In our project we used only a hard decision on whether pixel is on edge or not (loss is either $0/1$). It can be turned into a version with soft decision where each pixel has a prediction on the probability it is an edge pixel. By doing this seems would adding more detail loss and may produce a more accurate result. Some other methods should be developed to deal with cases when lots of complex edges occurs in one single image occurs. What's more, due to the fact that deep lab's segmentation method achieves super high accuracy, adding this method to other more advance architecture may generate a competitive result which is simply an end to end network.

Considering the fancy test like $Figure14$, other network structures that would collect depth information or trained by depth map would help this network develop further. For example, network can perform $3D$ segmentation, using training data collected by RGB-D camera. Since in real world, lots of images with cars, people would occur in large poster of screens, it of great importance to tell them apart from true objects to objects' image and know the size of such objects.

References

[1] N. Fletcher, "CLASSIFICATION VS DETECTION VS SEGMENTATION MODELS: THE DIFFERENCES BETWEEN THEM AND WHEN TO USE EACH," 2019.

[2] `https://towardsdatascience.com/review-fcn-semantic-segmentation-eb8c9b50d2d1`

[3] `https://towardsdatascience.com/review-deeplabv3-atrous-convolution-`semantic-segmentation-6d818bfd1d74.

[4] Shelhamer E , Long J , Darrell T . Fully Convolutional Networks for Semantic Segmentation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2016:1-1.

[5] Long J , Shelhamer E , Darrell T . Fully Convolutional Networks for Semantic Segmentation[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2014, 39(4):640-651.

[6] Pascal VOC 2011: `http://host.robots.ox.ac.uk/pascal/VOC/voc2012/`

[7] NYUDv2: https://cs.nyu.edu/~silberman/projects/indoor_scene_seg_sup.html

[8] GitHub: https://github.com/aurora95/Keras-FCN

[9] Simonyan K , Zisserman A . Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer Science, 2014.

[10] He K , Zhang X , Ren S , et al. Deep Residual Learning for Image Recognition[J]. 2015.

[11] Chen L C, Papandreou G, Kokkinos I, et al. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2018, 40(4):834-848.

[12] Chen L C , Papandreou G , Kokkinos I , et al. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2016, 40(4):834-848.

[13] Chen L C , Papandreou G , Schroff F , et al. Rethinking Atrous Convolution for Semantic Image Segmentation[J]. 2017.

[14] Chen L C , Yang Y , Wang J , et al. Attention to Scale: Scale-aware Semantic Image Segmentation[J]. 2015.