New Approach to Image Segmentation and Semantic Edges

Tyler Higgs
MIT
thiggs@mit.edu

Abstract

One way to improve current image segmentation methods is to encode information about shape of segmentation in a separate CNN since shape information could be significantly different from color and texture information. It may be difficult to store color, shape and texture information in a single network. The network encoding color and texture information feeds its information into the network encoding shape information in past works. Here, I propose an improvement on the current state-of-the-art two-stream CNN by also feeding the network encoding color and texture information data from the network encoding shape information. In this paper, I compare the results from adding this connection and omitting this connection. I also compare the performance of my network compared to the current state-of-the-art two-stream CNN (called GSCNN). I had two findings. The first is that the state of the art GSCNN was trained on 8 GPUs and the code for this does not run on a single GPU machine. So I modified their net to use a lighter net (Resnet 18). Secondly, I found that adding the extra connection did not improve accuracy in terms of mask (mIoU) or boundary (F-Score) quality.

1. Introduction

Much research is being done to solve the problem of semantic segmentation. It is an important field with many applications such as self driving vehicles and image generation. Recently it has become standard to use a fully convolutional network (FCNN) to encode semantic information about an image and then decode this information to get pixel-wise predictions as required by the segmentation problem.

One way to improve this method is to actually encode this information into two FCNNs using gates. One will encode information about the shape and edges of the segmented image and the other can focus on other semantics such as color and texture. I propose a network that allows each of these networks to take outputs from certain layers of the other as inputs to some of its own layers. This is so that the networks can help each other out with their encoded semantic information.

2. Related works

There has been recent success using fully convolutional networks for segmentation by Long *et al*. [5]. These networks take pretrained classification nets like AlexNet [4] and VGGnet [1] and converts the final few fully connected layers to convolutional layers so that the output are downsampled images which encode semantic information about the input image. They are "heat maps" that give a low quality representation of where objects are in the image. They then upsample to obtain the same level of detail as the input image. To help the network learn how to upsample, skips are introduced so that in addition to the regular inputs to the network, earlier layers are also taken as inputs.

Huikai Wu *et al*. [7] improved on this by compressing the last half of this FCNN which decodes and upscales the segmentation image into a single Joint Pyramid Upsampling (JPU) module which also takes in the skip connections.

In this paper we will be using Resnet by He *et al*. [3] as our pretrained classification net.

Another more recent work by Takikawa *et al*. [6] introduced a Gated Shape CNN (GSCNN) which uses two networks for the talk of segmentation. One of these is a "normal" fully connected CNN for encoding semantic information about the image. The second is specifically meant to encode information about the edges and shape of the images. The information encoded by these networks are combined to create the final segmentation prediction. Upscaling is done with a JPU module.

3. Data Set

I use the Cityscapes dataset [2] for training and evaluation of the proposed network as done in the GSCNN paper [6]. This dataset contains images taken on roadways along with their segmentations. Segmentations distinguish between 19 classes and give their category as a typical classification net would. The categories are listed in Table 1.

Specifically, I train, validate, and test my models using

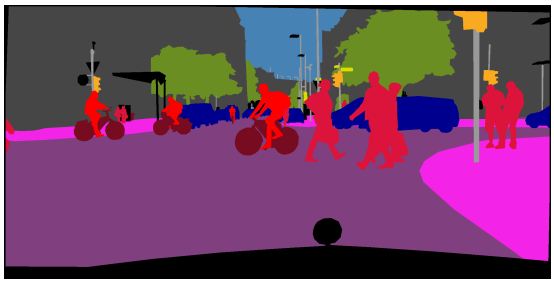

Figure 1. Example segmentation from the Cityscapes dataset.

Name	Category
Road	Flat
Sidewalk	Flat
Building	Construction
Wall	Construction
Fence	Construction
Pole	Object
Traffic Light	Object
Traffic Sign	Object
Vegetation	Nature
Terrain	Nature
Sky	Sky
Person	Human
Rider	Human
Car	Vehicle
Truck	Vehicle
Bus	Vehicle
Train	Vehicle
Motorcycle	Vehicle
Bicycle	Vehicle

Table 1. Names are the segmentation classes in the Cityscapes dataset along with their associated general categories.

the **leftImg8bit** dataset from Cityscapes.

The data goes through many transforms before ever entering the network. These transformations are described in the original GSCNN paper [6] and I have not altered these in my own model.

4. Approach

4.1. Fusion Module

All of the nets proposed in this paper will use the same fusion module from the GSCNN [6]. It is a way to combine the results from the two streams as can be visualized in **figure 2**.

4.2. Regular stream and CPUs

The GSCNN proposed by Takikawa *et al.* [6] uses a WideResNet38 [8] and was trained on 8 NVIDIA GPUs,

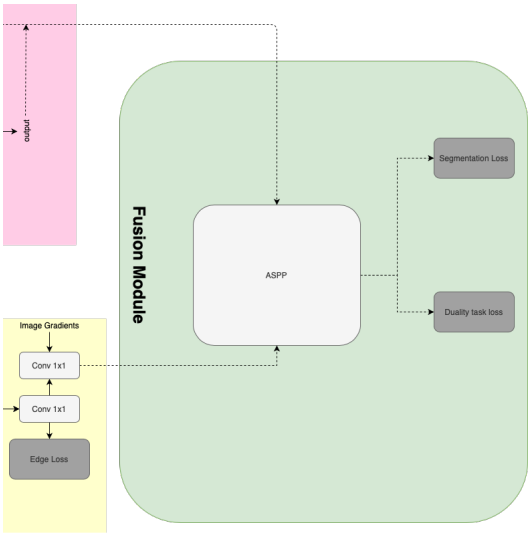

Figure 2. The fusion model, taken directly from the original GSCNN model [6].

and as such, training is an issue with on a single GPU. As far as I can tell, there does not exist a publicly accessible implementation of this network that instead uses a lighter net for the regular stream, such as ResNet-18 [3]. So the first thing that I set out to do was to implement this and make it publicly accessible. It might be that someone else who also lacks 8 GPUs can modify my implementation of the GSCNN to create a network with similar performance to the original GSCNN, but one that bears less of a burden to the machine and can train on a single GPU. **Figure 2** shows my proposed network.

The ResNet-18 architecture has a four residual blocks and this will be the structure of our regular stream. The input to the first residual block in the regular stream is used as the input to the first residual layer of the shape stream after bringing it through a 1×1 convolution which will bring the number of channels in its output to 1. The shape stream will remain the same as the original structure's, with three residual blocks. In between the residual blocks in the shape stream and after the final residual block are gates which combine the output of the previous layer of the shape stream with the output of one of the layers of regular stream. The first gate takes in the output of the first residual block of the shape stream and the second residual block of the regular stream (convolved 1×1 so that it only has a single channel). the gate's output is taken as input to the second residual block of the shape stream. The second gate takes in the output of the second residual block of the shape stream and the third block of the regular stream (convolved as before). The third gate takes in the output of the third residual block of the shape stream and the fourth block of the regular stream. The input to the Fusion model will then be the out-

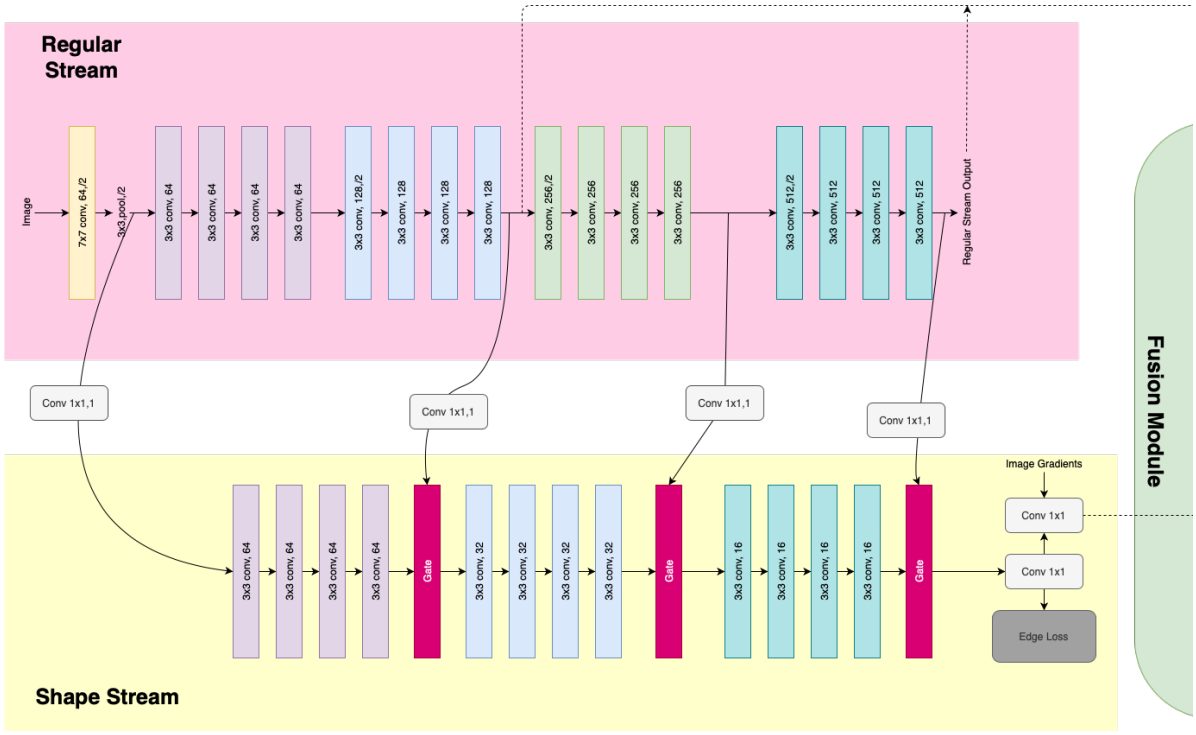

Figure 3. Here is the structure of the regular and shape streams of my proposed ResNet-18 [3] version of the GSCNN [6]. Each box is a convolutional layer with the specified kernel size and number of channels.

puts of the second and fourth residual blocks of the regular stream and the output of the third gate of the shape stream as described by the original GSCNN.
Resnet 18 GSCNN code: https://github.com/tylerhiggs/Res18GSCNN

4.3. Two directions of connection

Layers of the shape stream of the GSCNN take as additional input the layers from the regular stream. This is because segmentation information is encoded in the layers of the regular stream and that information is useful for detecting the segmentation edges. It would make sense that information about semantic edges should also be useful for determining semantic information itself. So in addition to the links from the regular stream to the shape stream as proposed by Takikawa *et al.*, I also introduce a link from the shape stream to the regular stream in this proposed network. **Figure 4** shows my proposed network.

There are two streams and the first is the **"regular" stream**. This will mimic the regular stream of the net proposed in section 4.2. The only difference is that the fourth residual block will have to take in the additional input from the shape stream. To do this, I bring the output from the third residual layer through a 1×1 convolutional layer that outputs 128 channels ($128 \times h \times w$). We also expect an input from the shape stream of dimensions $128 \times h \times w$. Then the layers are concatenated along the dimension that represents the number of channels, resulting in a tensor of size $256 \times h \times w$ which is the kind of input that the next layer of the regular stream expects. This output is also what the second gate in the shape stream expects so this is where we make a connection to that gate.

The shape stream gains a duplicate second residual block such that there are two identical residual blocks between the first and second gate. The output of the first of these two blocks is sent to the top layer to be concatenated as stated in the previous paragraph. Before it can do this, that output goes through a convolutional layer so that it has dimensions $128 \times h \times w$ and is compatible for concatenation.

Modified Resnet 18 GGSCNN code: https://github.com/tylerhiggs/Res18GSCNNmodified

5. Results

I trained each of the proposed networks (section 4.2 and 4.3) for 20 epochs and using the Cityscapes dataset. This dataset was also used for testing and validation. All of the hyperparameters were kept the same as the original GSCNN [6]. Here I will share the outcomes of each of these.

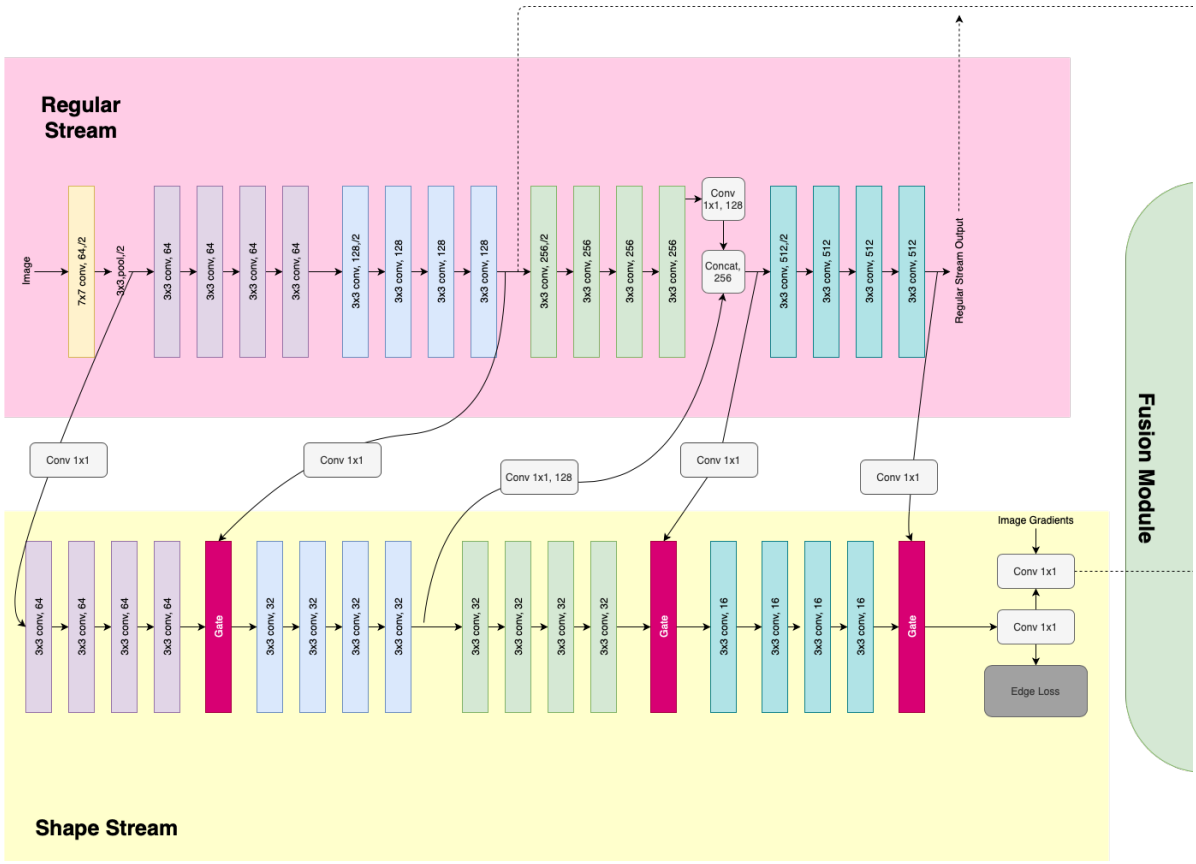

Figure 4. Here is my proposed regular and shape streams. They both individually follow the ResNet-18 [3] architecture. The overall architecture resembles very nearly the GSCNN [6]. These networks will act as inputs to the Fusion Module which also comes from the GSCNN. (A note to the grader of the progress report: feedback is greatly appreciated, I can easily edit this diagram and anticipate doing so, this is just what I have so far)

5.1. ResNet-18 GSCNN

The first comment I would like to make is that there is a warning on the original GSCNN code that says that their code will probably not run on a single GPU. I did try to run it on a single GPU and found that it did not run. I looked in the "issues" section of the repository and saw that some people had commented on this, looking for a version of the GSCNN that would run on a single GPU. The issue was clearly the WideResNet32 [8] in the regular stream. A lighter net could solve the problem. So I implemented the GSCNN using instead a ResNet-18 [3] for the regular stream. The best accuracy was achieved in epoch 18 and was **74.2 percent**. The original GSCNN achieved 82.9 percent accuracy using 175 epochs. While my network lags behind in accuracy, more machines are capable of using it to train. My hope is that having more people able to access this version of GSCNN, more people will play around with it, maybe even to the point of achieving a network with greater accuracy than the original state of the art GSCNN.

5.2. Modified ResNet-18 GSCNN

There is not much to say here. This network achieved an accuracy of 48.8 percent. It is clear that the original structure of the GSCNN was well thought out and tested. Altering this structure worsened accuracy significantly.

6. Conclusion

I have made publicly available a ResNet-18 version of the GSCNN which was requested by people on the internet so that they could run a GSCNN on a single GPU. I hope that some of these people can use this to construct a better network than I could. My experimental network which linked the shape stream to the regular stream made the net have significantly worse accuracy. My experiment was small which means that there could be a way to incorporate this strategy and somehow improve performance of the GSCNN, whether that be a different kind of connection, different number of connections, or different locations of connections. However, my experiments fail to show that

this is a meaningful path to go down.

7. Acknowledgments

I would like to thank Towaki Takikawa, David Acuna, Varun Jampani, and SanjaFidler for sharing their implementation of GSCNN [6].

I would also like to thank the staff of MIT 6.819/6.869 Advances in Computer Vision for teaching me many computer vision principles essential to the experiments done here.

References

[1] Christopher Bishop. Pattern recognition and machine learning. *Springer-Verlag New York*, page 229. 1

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 2, 3, 4

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012. 1

[5] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 1

[6] Towaki Takikawa, David Acuna, Varun Jampani, and Sanja Fidler. Gated-scnn: Gated shape cnns for semantic segmentation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 1, 2, 3, 4, 5

[7] Huikai Wu, Junge Zhang, Kaiqi Huang, Kongming Liang, and Yizhou Yu. Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation. *arXiv e-prints*, 2019. 1

[8] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 2, 4