CoE202 Final Project Report

새내기과정학부 20210337 신명진

2021-06-14

Abstract

The main purpose of this project is to get high mIoU using neural network. First, I used the same unet which the professor offered. Using that model, I got 0.24 mIoU maximum. Then, I tried some variations on the model and I got 0.28 mIoU with the modified unet. Then, I tried data augmentations and I got 0.43 mIoU. I realized that I have to use another model to get mIoU higher than 0.5. So I used the ASPP module and unet decoder for decoding and used the structure of deeplab for the encoder. Finally, I got 0.5784 mIoU. This is the main routine of my effort. I will also refer about the changed optimizer, hyperparameters, and talk more precisely about the data augmentation and the model. cf) I perchased pro-colab and trained the model.

1 Data augmentation

I tried Data augmentation on 4-layer unet. The augmentation which got the best mIoU was selected and used on other models that I choosed. So I will only talk about how the augmentation improved the performance on 4-layer unet. Obviously for validation set I didn't apply augmentations and used the original image.

1.1 Randomcrop

Rather than Centercrop I used Randomcrop. Since we can't afford the full size image for train, we have to cut or resize the image. I croped the image to the size and I got 2 images from the original image by randomcroping and used it for training. I thought that this could get more information from the original image. My prediction was right and The final valid mIoU was increased 0.28 to 0.32. Randomcrop prevented overfitting very well. Using Centercrop the overfitting occured when the train mIoU was 0.21 but after using Randomcrop the overfitting occured at 0.33.

1.2 RandomFilp

By filping the image I thought that it can prevent overfitting than before. The train set will have more variations and noise. The valid mIoU was increased 0.32 to 0.35.

1.3 Color Transform

Bus, airplane, boat, ... can have various colors. So I applyed color transform to attain various colors from train set. But some objects like people, motorcycle have simliar color in the same object. So I applyed this transform only at 0.3 probability. The valid mIoU was increased 0.35 to 0.37.

1.4 Normalization and Affine Transform

I read articles and blogs about unet and they said using Normalization, Affine Transform increased the performance. So I applyed it and the performance was highly increased. The valid mIoU was increased 0.37 to 0.43. It prevented overfitting. Before overfitting occured when train mIoU was 0.35 but now it occured at 0.42.

2 Model

2.1 3-Layer Unet

First I tried the same model which the professor offered us. I only changed the channel numbers. It didn't increased the performance very well. I tried 4 times with this model and I felt that I should chagned the model because overfitting was occuring too fast. With this model the max valid mIoU was 0.24.

2.2 4-Layer Unet

Since, overfitting occured too fast I thought that parameters were too few. So I put one more layers to the decoder and encoder. But the this didn't fix the overfitting issue. The mIoU was only increased 0.24 to 0.28. Next, I tried data augmentation at this point. This fix the overfitting problem significantly. And the valid mIoU was increased to 0.431. At this point, I chagned the activation function ReLU to LeakyReLU. There was no reason, I just wanted to experiment the result so I tried. The mIoU was increased to 0.442. LeakyReLU can return negative values so I think it was the reason the mIoU increased. It was mimic change so I could have been the effect of randomness in data augmentation. So I repeately train the model 3 times, but the results were always slightly high than using ReLU. So I choosed LeakyReLU from this point.

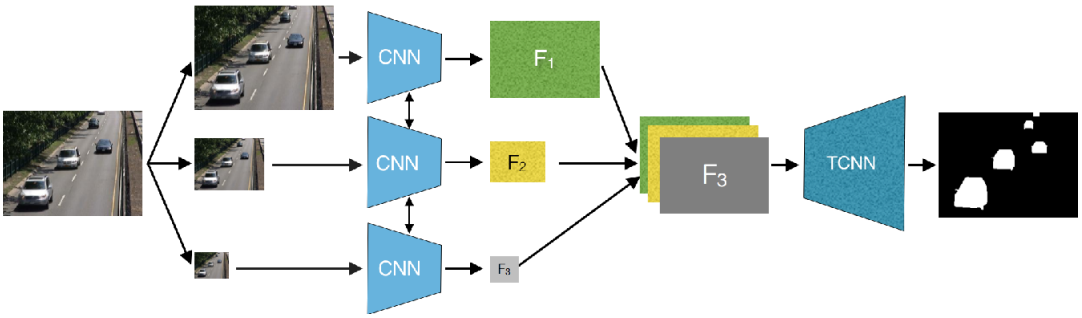

Figure 1: MultiScale CNN

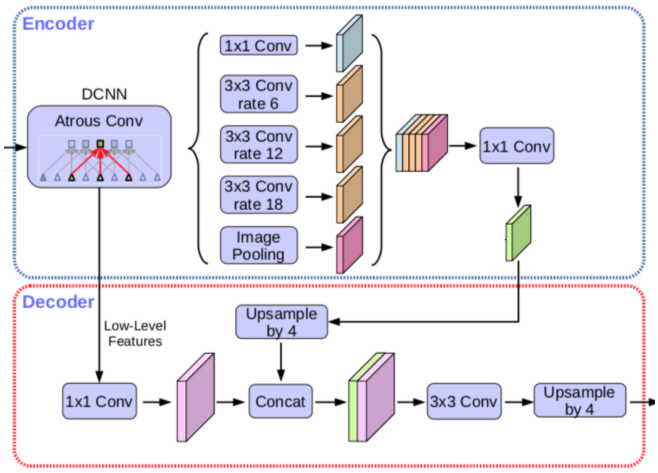

Figure 2: ASPP

2.3 Multi Scale Unet

By searching the internet, I found that there was a technique called multi scale cnn. I used this technique to unet and below is the structure. Let's say that the original image size is 256. We put image the size of 256, 128, 64 as input into separable decoders. The output of three decoders are concatenate and put into the encoder. The decoders and encoder is same as unet. Sadly, the performance was not that improved. The valid mIoU was increased 0.44 to 0.46. But the training speed was too slow. It was slower 2 times than the original unet. So tried an another model.

2.4 ASPP

Short for Atrous Spatial Pyramid Pooling, ASPP is a module used in the model deeplab. We can code this by adding dilation in nn.Conv2D. This can improve the field of view so It improves the performance very well. I used the

same decoder and encoder of unet. The structure is like the Figure 2. This structure is the model deeplab. I changed some parameters. By experiment the dilation rate of the ASPP was best with 3,6,12 for my model. And for the last decoder I set the dilation rate 2. Another thing that I changed was the activation function. I used LeakyReLU because my experiment result was good than ReLU. Finally, I got valid mIoU 0.5744 from this model. Not only for mIoU the model was very light than unet. The model was 121MB by using unet but using deeplab it was 31MB. This mean that the parameters were less than unet. And the training speed was almost 3 times faster than unet. And overfitting occured on unet when train mIoU was 0.42 but in this model it occured at 0.55. I could realize important of a model.

2.5 Depthwise Separable Convolution

In the final model which I submitted I didn't use this method. Because the performance dropped to 0.563. But I think this is only because of the randomness of data augmentation. In my opinion, the result should have been better. Depthwise Separable Convolution does Depthwise Convolution then 1x1 Pointwise Convolution. This lower the parameter numbers and increases the speed of training than the original Convolution. Actually, the speed was increased about 1.2 times for 1 epoch.

3 Optimizer

First, I used the Adam optimizer which was in the code that professor gave. The Final valid mIoU was about 0.55. The adamW is known to be more good at generalizing the L2 regularization and weight decay, especially on image nerual networks. So I used the AdamW optimizer and the valid mIoU increased to 0.57.

4 Hyperparameters

4.1 Image size and Batch size

First I set the Image size to 128 and the Batch size to 25. Than I realized that higher the Image size is less overfitting occures. So I increased the size to 256. But cuda out of memory occured. I reduced the batch size to 12. Only because to resolve the cuda memory error. This is why I used 256 for croping training image size and 12 for batch size. For validation image I didn't changed anything.

4.2 Learning rate

When the train mIoU got higher I felt that the learning rate was too high. Because the train loss oscillated. But when I lower the learning rate the training was too slow or was in the local minimum too long. So I used the learning rate scheduler. I used the step-drop learning rate decay. I set the gamma which is the decay rate to 0.9. And the step size to 5.

5 Result

Finally I used deeplab structure for model. The decoder was same with unet and I changed the variables in ASPP and some Convolution layers. And changed the activation function to LeakyReLU. The encoder was same with unet too. I used AdamW for the optimizer and learning rate scheduler. I also applyed various data augmentations too. by this efforts, the final valid mIoU was 0.5744 and test mIoU was 0.5785. My model detected aeroplane and motorcycle very well. But was poor at detecting people and bicycles.