Semantic Segmentation

Yang Tiancheng

March 12, 2023

1 Background

A hot topic in contemporary computer vision is semantic segmentation. In image analysis, semantic segmentation aims to assign each pixel in an image to its corresponding semantic category. Semantic segmentation provides finer annotation for each pixel than image classification, which only needs to recognize the entire image. This makes it ideal for many computer vision applications, including autonomous driving, medical image analysis, robotic navigation, and more. This article will introduce semantic segmentation and its common algorithms and implementation steps.

2 Introduction

2.1 What is semantic segmentation

Semantic segmentation is an image segmentation technique whose goal is to associate each pixel in an image with its corresponding semantic category. This means that for a given image, we will assign a label or category to each of its pixels, which will help us better understand the content in the image. Unlike traditional image segmentation methods, semantic segmentation focuses on the semantic meaning of each pixel segmented.

Traditional image segmentation simply divides an image into several different regions, without considering the content within each area. In semantic segmentation, we require that the segmentation result be consistent with the semantics in the image. For example, for a road image, we need to assign each pixel in the image to different semantic categories such as "pavement", "sidewalk", "building", "car", etc.

2.2 Common Algorithms

2.2.1 FCN

FCN (Fully Convolutional Network) is one of the most well-known deep learning algorithms in semantic segmentation. It is an end-to-end semantic segmentation method based on convolutional neural networks. Unlike traditional convolutional neural networks, FCN does not use a fully connected layer in the last layer, but instead uses a convolutional layer and an upsampled layer to obtain a dense output of the original input image. Specifically, FCN uses convolution and deconvolution operations across multiple resolutions to produce a dense pixel-to-pixel map. In addition, FCN uses skip architecture to fuse feature maps at different scales to obtain finer segmentation results.

2.2.2 U-Net

U-Net is an encoder-decoder structure based on convolutional neural networks specifically for biomedical image segmentation. The encoder part of U-Net uses a convolutional layer similar to the VGG network for extracting image features. The decoder section uses a deconvolutional layer for upsampling to obtain finer segmentation results. In addition, U-Net uses skip architecture to fuse the feature maps of encoders and decoders to improve segmentation accuracy and detail retention.

2.2.3 SegNet

SegNet is an encoder-decoder structure based on convolutional neural networks for semantic segmentation. The encoder part of SegNet adopts the structure of VGG network to extract image features. The decoder section uses a deconvolutional layer for upsampling. SegNet uses maximum pooled indexes to record each maximum pooling position in the encoder so that these indexes can be used in the decoder for upsampling. This process preserves more detail and improves the accuracy of segmentation results.

2.3 SegNet

SegNet network is the beginning of the clear definition of the ecoder end and the decoder end, its ecoder end is the use of Vgg16, a total of 13 convolutional layers of Vgg16, compared to the use of more typical ecoder to extract image features, SegNet improvement focuses on the well-designed decoder end, the decoder end will pool indexes technology applied to max The pooling process connects the output of the encoder to do a nonlinear upsampling, and according to the description of SegNet, using SegNet strikes a good balance between computer consumption and accuracy of predictive classification. An important improvement over fully convolutional segmentation networks by SegNet is in pooling indices, embodied in encoder and decoder processes, as exemplified below.

2.4 Principle of SegNet

SegNet network is the beginning of the clear definition of the ecoder end and the decoder end, its ecoder end is the use of Vgg16, a total of 13 convolutional layers of Vgg16, compared to the use of more typical ecoder to extract image features, SegNet improvement focuses on the well-designed decoder end, the decoder end will pool indexes technology applied to max The pooling process connects the output of the encoder to do a nonlinear upsampling, and according to the description of SegNet, using SegNet strikes a good balance between computer consumption and accuracy of predictive classification. An important improvement over fully convolutional segmentation networks by SegNet is in pooling indices, embodied in encoder and decoder processes, shown as Figure 1 [1].

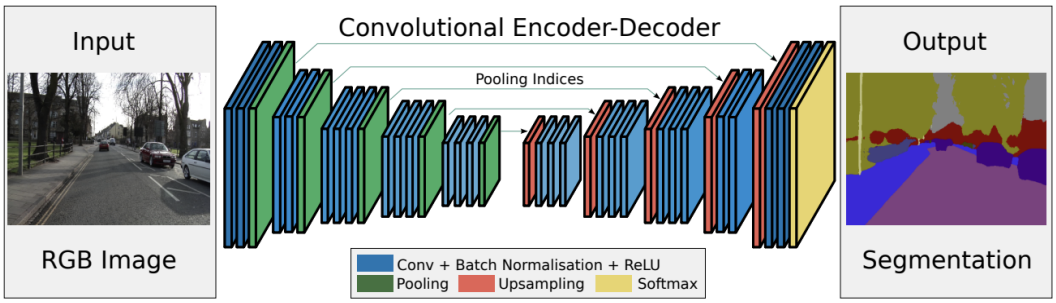

Figure 1: SegNet Architure

2.5 Illustration of SegNet

On the encoder side, the article records the use of a simple max pooling 2x2 window width, step size of 2, there is no doubt that the image after using max pooling will become smaller, and according to such parameters, it will become half of the original image, and at the same time, this will increase invariance, such as large-scale backgrounds, accompanied by a decrease in spatial resolution. There is no doubt that this decrease in spatial resolution is detrimental to the delineation of boundaries, and in the discussion of the article, the author believes that this boundary information needs to be stored and retained, such as the author recorded the maximum localtion in each pooling window in each feature map; In short, in addition to the regular conv, bn, and maxpooling on the encoder port, the author recorded the maximum location in the pooling window. On the Decoder side, the normal convolution, BN, and REU operations are also used, the difference is that a softmax is added at the

end to organize the probability graph of the output k channels, and k is the specified number of segmentation categories. But the difference is that SegNet uses the pooling indices obtained on the encoder side in the upsampling process, and uses this to guide the upsampling process, rather than directly using a convolutional process for upsampling like the same convolutional network, so that each pixel position after upsampling is a weighted average of the input image before upsampling. The details are as Figure 2.

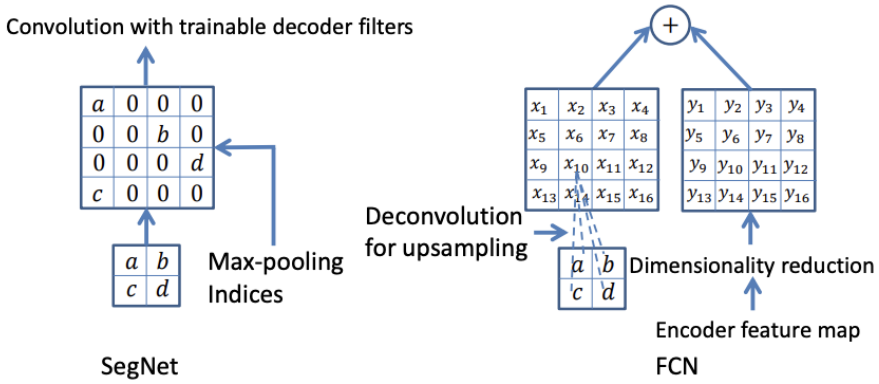

Figure 2: SegNet Illustration

2.6 Implementation

2.6.1 Steps

When implementing a semantic segmentation algorithm, the following steps are generally required:

1) Data preparation: Collect and tag semantic segmentation training and test sets to ensure correct labeling.

2) Network design: Select the network structure that suits the task and make adjustments.

3) Model training: Train the network with a training set and validate with a validation set.

4) Model evaluation: Use test sets to evaluate the performance of the model and make necessary adjustments.

2.6.2 Attention

As we implement each step, we need to be aware of the following:

1) When preparing data, care should be taken to collect representative data and label it correctly.

2) When designing a network, we should choose a network structure that is suitable for the task and make necessary adjustments.

3) When training a model, we should choose the appropriate optimizer and loss function, and train with appropriate hyper-parameters.

4) When evaluating a model, we should use multiple evaluation metrics and use visualization tools for visual analysis.

5) When applying deployment, the trained model should be deployed to the real scene, and necessary testing and optimization should be performed.

2.6.3 Code and Explanation

In the encoder part, multiple layers of convolution and pooling are used to gradually extract the feature information of the input image. In the decoder part, the feature map output by the encoder is restored to the original image size by deconvolution and upsampling operations, and pixel-level classification is performed [2].

Specifically, first we have to denfine a SegNet class, which inherits from nn. Module class. In the constructor of the class, we need to specify the number of input channels and the number of output channels.

Next, an encoder with multiple convolutional layers, batch normalization layers, and ReLU activation functions is defined. Each convolutional layer is followed by a batch normalization layer and a ReLU activation function. After each pooling layer, a maximum index of the pooling layer is also returned for upsampling of the decoder portion. The final layer of the encoder section is a pooling layer that reduces the input image size to 1/32 of the original size. Next, a decoder containing multiple deconvolution layers, batch normalization layers, and ReLU activation functions is defined. Each layer of the decoder section is a combination of a deconvolutional layer and a ReLU activation function. Before each deconvolutional layer, a batch normalization layer is also added. When deconvolving and upsampling, the feature map size needs to be restored based on the maximum index returned by the encoder partial pooling layer.

Finally, a forward function is defined for forward propagation. In this function, the encoder part of the input image is first calculated to obtain the feature map output by the encoder. Then, according to the maximum value index returned by the pooling layer of the encoder part, the deconvolution and upsampling operations of the decoder part of the feature map are performed, and finally the predicted segmentation image is obtained.

Code is shown Here .

2.6.4 Result

The result is shown as Figure 3.

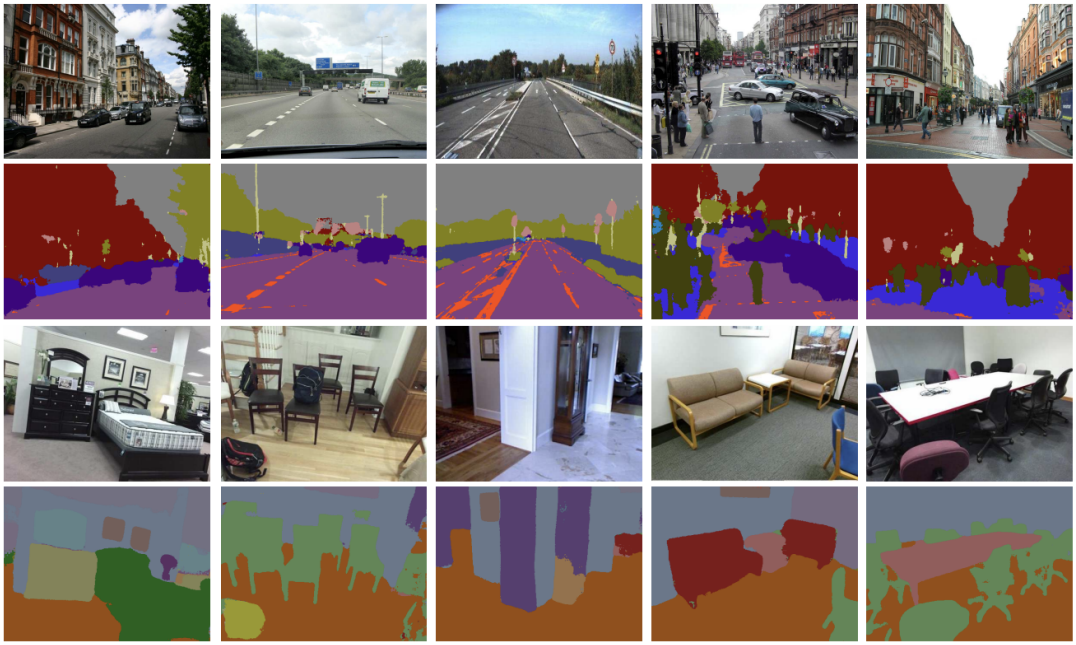

Figure 3: Semantic Segmentation Result

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[2] xiaoxifei. Segnet, 2019. https://blog.csdn.net/xiaoxifei/article/details/88710506.