Image Semantic Segmentation

In computer vision, **image segmentation** is the process of partitioning a digital image into multiple segments [6]. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

### Applications

Some of the practical applications of image segmentation are [6]:

- **Object detection**
  - Face detection
  - Pedestrian detection
  - Locate objects in satellite images (roads, forests, crops, etc.)

- **Medical imaging**
  - Locate tumors and other pathologies
  - Measure tissue volumes
  - Diagnosis, study of anatomical structure
  - Surgery planning
  - Virtual surgery simulation

- **Machine vision**
- **Traffic control systems**

Image Segmentation is a wildly used application in many fields, such as Artificial Intelligence and modern medicine.

An **autonomous car** (figure 1) is a vehicle that is capable of sensing its environment and navigating without human input [7]. Autonomous cars can detect surroundings using a variety of techniques such as radar, lidar, GPS, odometry, and **computer vision**. Advanced control systems interpret sensory information to identify appropriate navigation paths, as well as obstacles and relevant signage. Autonomous cars have control systems that are capable of analyzing sensory data to distinguish between different cars on the road, which is very useful in planning a path to the desired destination. Image semantic Segmentation is the key technology of autonomous car, it provides the fundamental information for semantic understanding of the video footages, as you can see from the photo on the right side, image segmentation technology can partition the cars, roads, building, and trees into different regions in a photo. These information are sent to the advanced control systems, which helps the controller to make right decisions during the driving.

**Medical imaging** (figure 2) is the technique and process of creating visual representations of the interior of a body for clinical analysis and medical intervention, as well as visual representation of the function of some organs or tissues. Image segmentation can extract clinically useful information from medical images using the power of convolutional neural networks. It can be applied to complex medical imaging problems, such as quantifying damage after traumatic brain injury or organ injury.

---

**Figure 1:** Image segmentation during autonomous car driving

**Figure 2:** Image segmentation for medical imaging
Technical methods

There are multiple methods to do image segmentation, some of them are listed in the table below:

- Thresholding
- Clustering method
- Compression-based methods
- Histogram-based methods
- Convolutional neural networks

This wiki page will focus on the method of convolutional neural networks. In machine learning, a convolutional neural network (CNN, or ConvNet) is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex. And in computer vision, CNNs are powerful visual models that yield hierarchies of features, based on CNNs there are several ways to do image segmentation, some of them are listed in the following table:

- Fully Convolutional Neural Networks [1]
- Deep Convolutional Encoder-Decoder Neural Networks [2], [3]

Short Introduction

Fully Convolutional Neural Networks

Fully convolutional indicates that the neural network is composed of convolutional layers without any fully-connected layers usually found at the end of the network. To semantic segment a image, a "fully convolutional" neural networks (figure 4) can take input of arbitrary size and produce correspondingly-sized output with efficient inference and learning, both learning and inference are performed whole-image-at-a-time by dense feedforward computation and backpropagation. In-network upsampling layers enable pixelwise prediction and learning in nets with subsampled pooling.

Deep Convolutional Encoder-Decoder Neural Networks

A convolutional encoder-decoder neural network (figure 5) is composed of a stack of encoders followed by a corresponding decoder stack which feeds into a soft-max classification layer. The decoders help map low resolution feature maps at the output of the encoder stack to full input image size feature maps. This addresses an important drawback of recent deep learning approaches which have adopted networks designed for object categorization for pixel wise labelling.

Fully Convolutional Networks for Semantic Segmentation

Introduction

A fully convolutional indicates that the neural network is composed of convolutional layers without any fully-connected layers usually found at the end of the network. To our knowledge, the idea of extending a convnet to arbitrary-sized inputs first appeared in Matan et al [4], which extended the classic LeNet to recognize strings of digits, because their net was limited to one-dimensional input strings, Matan et al. used Viterbi decoding to obtain their outputs. In the follow wiki text, a fully convolutional network presented by UC Berkeley is introduced [1], the net is trained end-to-end, pixel-to-pixel on semantic segmentation. And also this was the first work to train FGNs end-to-end (1) for pixelwise prediction and (2) from supervised pre-training.

Adapting classifiers for dense prediction
Typical recognition nets, including LeNet, AlexNet [5], ostensibly take fixed-sized inputs and produce non-spatial outputs. The fully connected layers of these nets have fixed dimensions and throw away spatial coordinates. However, these fully connected layers can also be viewed as convolutions with kernels that cover their entire input regions. Doing so casts them into fully convolutional networks that take input of any size and output classification maps. This transformation is illustrated in the figure on the right (figure 6).

For example, for the normal convolutional network deal with classification tasks, there will be some fully connected layers at the end of network, the output of fully connected layers pass through the soft-max function, several probabilities are generated, the input image will be classified into the class which has highest probability. For a fully convolutional network, fully connected layers are replaced by convolutional layers, the spatial output maps of these convolutionalized models make them a natural choice for dense problems like semantic segmentation.

**Upsampling is backwards strided convolution**

A way to connect coarse outputs to dense pixels is interpolation. For instance, simple bilinear interpolation computes each output from the nearest four inputs by a linear map that depends only on the relative positions of the input and output cells. In a sense, upsampling with factor \(r/s\) convolution with a fractional input stride of \(r/s\). So long as \(r/s\) integral, a natural way to upsample is therefore backwards convolution (sometimes called deconvolution) with an output stride of \(r\). Note that the deconvolution filter in such a layer need not be fixed, but can be learned.

**Segmentation Architecture based on VGG**

VGG neural network did exceptionally well in ImageNet Large Scale Visual Recognition Competition 2014. The fully connected layers of VGG are replaced by convolutional layers, after the input image pass through the original 5 convolutional layers of VGG, it will be upsampled by 2 deconvolutional layers to generate a output image with the same size as input image. For each pixel, it will be inputted to the soft-max function, every pixel will be classified into the class with highest probability, and illustrated with different color in the output image.

But the output segmented image is not accurate enough compared to the ground truth, and it is known, after pass through convolutional layers and pooling layers, the size of the image and also resolution is reduced. In order to increase the accuracy of image segmentation, coarse, high layer information are combined with fine, low layer information. So not only the output layer is upsampled, but also the pooling layers before, and the results is combined with the output layer, which shows a better performance. Pass through the original 5 convolutional and pooling layers of VGG network, the resolution of the image is reduced 2, 4, 8, 16, 32 times. For the last output image, it should be upsampled 32 times to be the same size as the input image (Figure 7). More details are summarized in the following table [1]:

- First row (FCN-32s): the single-stream net, upsamples stride 32 predictions back to pixels in a single step.
- Second row (FCN-16s): Combining predictions from both the final layer and the pool4 layer, at stride 16, lets the net predict finer details, while retaining high-level semantic information.
- Third row (FCN-8s): Additional predictions from pool2 at stride 8, provide further precision. Three upsampled ways are illustrated in the right figure, pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines.
The refined fully convolutional network is evaluated by the images from ImagNet, a example is illustrated in the (figure 8), from the left to the right are original images, output image from 32, 16,8 pixel stride nets and the ground truth. As you can see, the output image combine with $pool4$ layer is loser to the ground truth compared to FCN-32s and FCN-16s. The success of fully convolutional network for semantic segmentation can be summarized in the following table:

- Use the architecture of VGG network and the pre-trained weights and biases for first five convolutional layers as fine-tuning.
- Replace the fully connected layers by convolutional layers to solve dense prediction problem like image segmentation.
- Use deconvolutional layers to upsample the output image, make it same size as the input image, combine the information of pooling layers before to get better performance.

**Literature**


