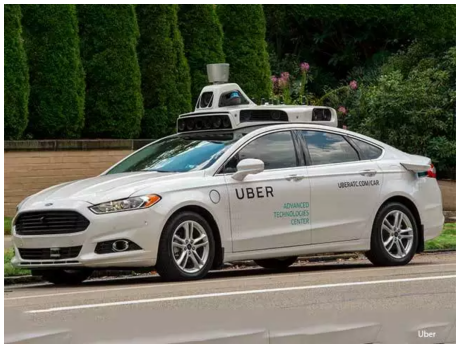
Semantic Segmentation and FCNs for Autonomous Vehicles

Manoj S
B.Tech Mechanical Engineering (Third year), IIT Madras

27 December 2020

1 What is an Autonomous Vehicle?

An Autonomous Vehicle (AV), also known as self-driving car, is a vehicle that is able to sense its surrounding environments using sensors like LIDAR, Sonar, etc. and act according to it so that the vehicle can move safely.



(a) Uber

(b) Google

Figure 1: Uber and Google's self-driving cars

An autonomous vehicle involves collection of a variety of data from its sensors which includes vision, sound, proximity, etc. Talking about the vision, the car needs to see what is around it so that the car can decide where to go and where not to. For this purpose, the vision data has to be collected using cameras and then they need to be processed. For this processing application, we use a method called **semantic segmentation**.

2 What is Segmentation?

Image Segmentation is the process of identifying different parts of the image and finding to which segment they belong to. The different segments here represent different objects. Thus segmentation can be considered to be a process which not only identifies the different objects present in the picture, but as well as represent the objects in the picture along with its pixel-wise boundaries.
Segmentation are of two types: Semantic Segmentation and Instance Segmentation.

2.1 Semantic Segmentation

Semantic Segmentation classifies all the pixels in the image into meaningful classes (according to real-world). For example, if two cats are present in a picture, semantic segmentation classifies the pixels present on both the cats to a single class.

2.2 Instance Segmentation

Instance Segmentation also involves classifying the pixels, but it classifies each and every instance of an object separately, even though they belong to the same class. For example, in the image with two cats, instance segmentation classifies the pixels in the cats separately as the two cats are two separate instances.

The following example shows the difference between instance and semantic segmentation.

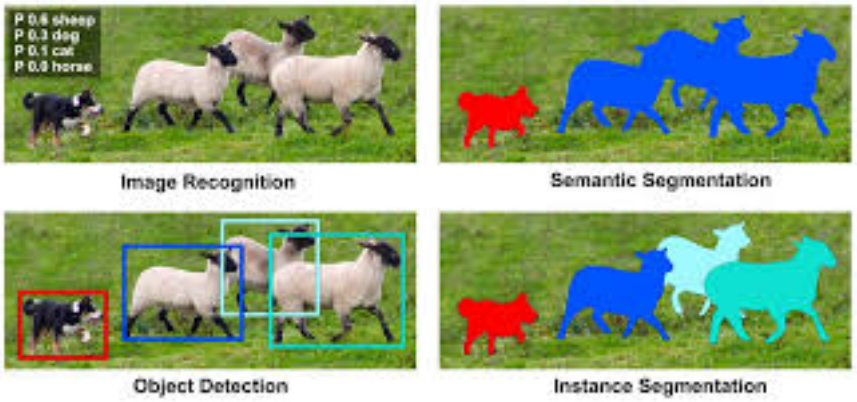


Figure 2: Segmentation Example

For autonomous vehicles, semantic segmentation is used as it will be able to detect objects like road, vehicles, etc. along with giving its position as well. We also need the model to give proper classes of the segments detected, not separate instances of the objects.

3 Performing Semantic Segmentation

There are many model architectures used for semantic segmentation, but all of them follow a similar basic concept. Semantic segmentation involves both image classification and object detection. To carry this out, an FCN (Fully Convolutional Network) architecture is used to **down-sample the image** so that the model can get information about the classes. The **low-resolution image is then up-sampled** (by using another FCN architecture in most of the cases) to get the output with original image resolution along with the masked regions of individual classes. Some of the famous model architectures used for semantic segmentation are as follows:

1. UNet
2. Mask R-CNN
3. DeepLab
4. FastFCN

In the following sections, the UNET and Mask R-CNN models are explained in detail.

4 UNet

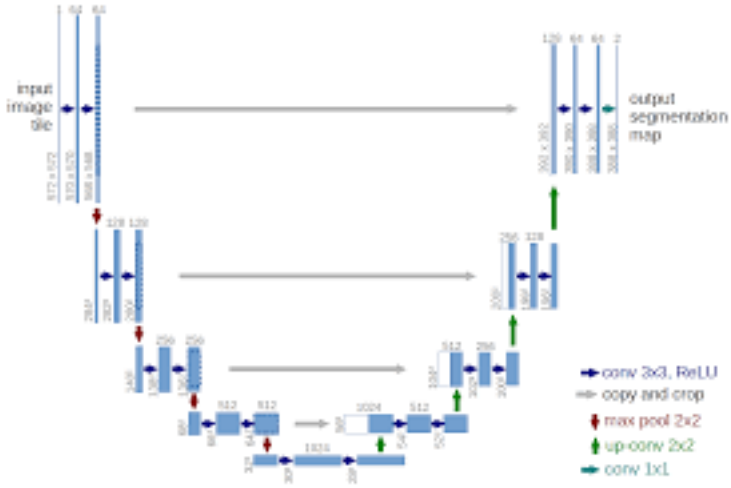


Figure 3: UNET Architecture

As said before, the UNet model involves down-sampling and then up-sampling the image using FCNs to perform semantic segmentation. The down-sampling part is known as the **encoder** and the up-sampling part is known as the **decoder**. The UNet model was first used in 2015 to process biomedical images. The model is famous for identifying and segmenting brain tumors using images. The architecture of the model is as follows:

1. Encoding using Convolutional Neural Network:
    - There are four encoder blocks involved in the standard UNet architecture.
    - Each encoder block consists two convolutional layers along with batch normalisation and Relu activation function, and then max-pooled to half the previous image size.
    - The filter sizes in the encoder blocks are 64,128,256 and 512 respectively.
2. Bottleneck layer:
    - The bottleneck layer is used to further enhance the encoded feature vector/tensor.
    - It has a convolutional layer with 1024 filters along with batch normalisation and Relu activation function.
    - Finally a transpose convolutional layer (1024 filter size) is present which up-samples the feature tensor to twice its size. Details about ConvTranspose2d layer in pytorch can be found here.
3. Decoding using Convolutional Neural Network:
    - The decoding layer again consists of four decoding blocks (same number as in encoding part).
    - A decoding block consists of two convolutional layers along with batch normalisation and Relu activation function, followed by a transpose convolutional layer which doubles the size of the input.
    - In order to perform the segmenting process effectively, the decoder block uses **skip connections**. These connections are present between every encoder-decoder block pair which are in the same level.

- The skip connections are used to directly concatenate the output from the corresponding levels in the encoder with the decoder. This further provides more information regarding the type and the location of a class, guiding the segmentation process.

By using the above processes specified, the model can segment the input image given. The model has to be trained with input image and its ground truth segmented images. After training, the model can be deployed for segmentation tasks on images and live videos as well.

5 Mask R-CNN

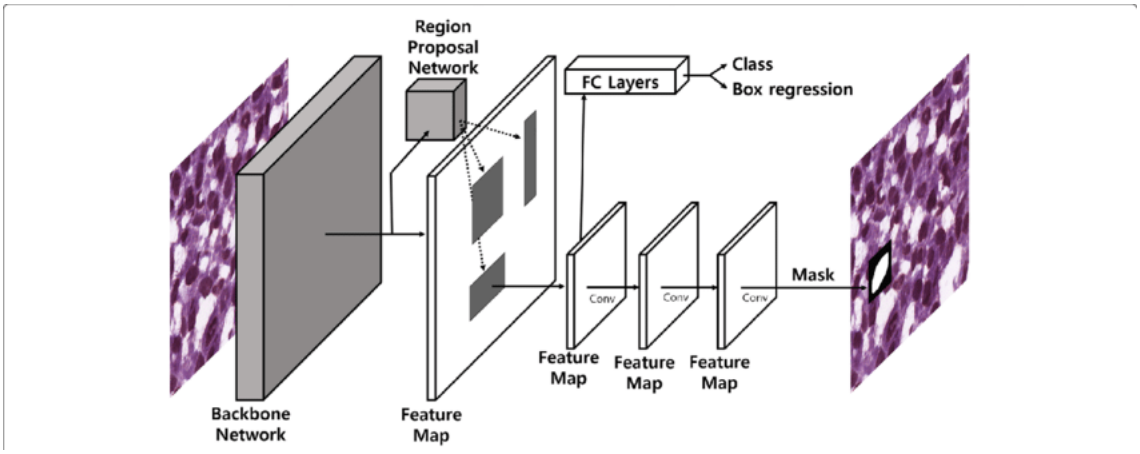


Figure 4: Mask R-CNN Architecture

The Mask R-CNN architecture is an extension of Faster R-CNN. Faster R-CNN involves predicting the object class and the bounding boxes for the objects. In addition to it, Mask R-CNN involves predicting segmentation masks on each Region of Interest (RoI). The architecture of Mask R-CNN model is as follows:

1. Backbone Model:
   - The backbone model is a convolutional network which is responsible for generating feature maps from the input image.
   - ResNet 101 is commonly used for backbone model. Even any other architecture like VGGNet or any other ResNet model can be used for the backbone model.
2. Region Proposal Network (RPN):
   - The RPN accepts the feature maps from the backbone model as input and predicts whether an object is present in a specific region or not.
   - From this network, we obtain the regions where objects are present in the image.
3. Region of Interest (RoI):
   - The regions predicted by RPN are of arbitrary shape. Thus a pooling layer is applied which converts all the different output shapes to same shape.
   - The RoI is then passed to a set of FC layers which predicts the class of the object present along with the bounding box for the object. The bounding box is predicted by using IoU (Intersection over Union) with the ground truth boxes. (IoU discussed in detail in a later section.)

- The Faster R-CNN has all the steps discussed till this point. The upcoming ones are not present in Faster R-CNN but present in Mask R-CNN.

4. Segmentation Mask Model:
   - The segmentation mask model contains a set of convolutional layers (3 or 4 layers) along with batch normalisation and Relu activation function.
   - The model accepts the features maps for the region of interests and pixel-wise segments the object from the background.

5.1 Difference Between Mask R-CNN and UNET

Both the networks perform well in segmenting tasks on live feed. But there are some advantages and disadvantages between them.

1. The UNet architecture just generates a segmented mask, but Mask R-CNN gives a segmented mask along with the bounding boxes and probability of detection of the object. There are less post-processing steps involved which applying Mask R-CNN, but that's not the case for UNET.

2. Training the UNET architecture takes lesser time than training the Mask R-CNN architecture. Also Mask R-CNN involves many hyper-parameters to tune than that in UNET.

3. To conclude, UNET is useful when only few classes are present and it is not required to find the bounding boxes. Otherwise Mask R-CNN or any other architecture can be used for semantic segmentation purpose.

6 Losses and Accuracies for Semantic Segmentation

Even though cross-entropy loss function can be used for semantic segmentation, the training happens to be very slow and also we can't give weightage to classes that we want to predict (might be having lesser occurrences). Some of the losses are as follows:

6.1 Dice Loss

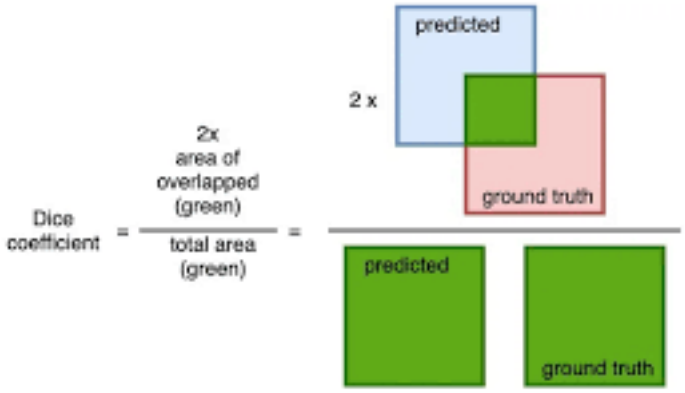


Figure 5: Dice Loss

The above mentioned formula is the one used to calculate dice coefficient (DC). 1-DC gives the dice loss. The dice coefficient can even be calculated class-wise as well, and also can be used as a accuracy score.

6.2 Intersection over Union (IoU)

$$IoU = \frac{TP}{(TP + FP + FN)}$$

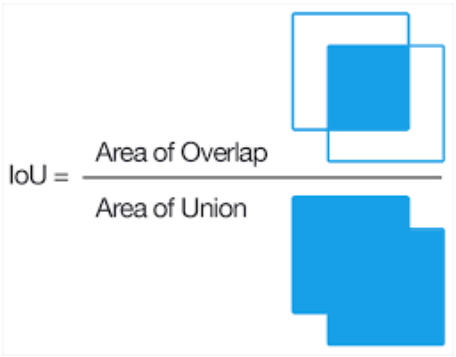


Figure 6: IOU Score

The IoU score is much similar to the dice score. The IoU score is the area of overlap divided by area of union. The IoU score is generally slightly lesser than the dice score. Again 1-IoU score gives the IoU loss which can be used for the segmentation purposes.

6.3 Focal Loss

$$\mathrm{FL}(p_\mathrm{t}) = -\alpha_\mathrm{t}(1 - p_\mathrm{t})^\gamma \log(p_\mathrm{t}).$$

Figure 7: Focal Loss

The Focal loss is a modified version of the cross-entropy loss. It involves weighing the categorical cross-entropy such that the harder to detect classes are given more preference. The classes which are easier to predict have higher probability of detection. Since 1-p is multiplied to the loss function, the losses for easier to detect classes are reduced to a large extent, thus helping the model focus on detecting the harder classes.

7 A Demonstration on using UNet for semantic segmentation

I have previously done a project on **Steel Surface Defect Detection** where there are four different classes of steel surface defects and the defects were located and segmented using the UNet architecture. A combination of cross-entropy and dice loss was used as the loss function, and IoU and Dice score was used to calculate the accuracy scores.

Github Link to the project: https://github.com/Manoj-152/Steel-Surface-Defect-Detection

The results obtained from the model are as follows:

Results

Model size: 14.3 M parameters

Image size	Val Accuracy	Val IOU	Val Dice	Forward inference speeds
64x400	0.99	0.96	0.98	0.026
128x800	0.99	0.97	0.98	0.09

Output samples

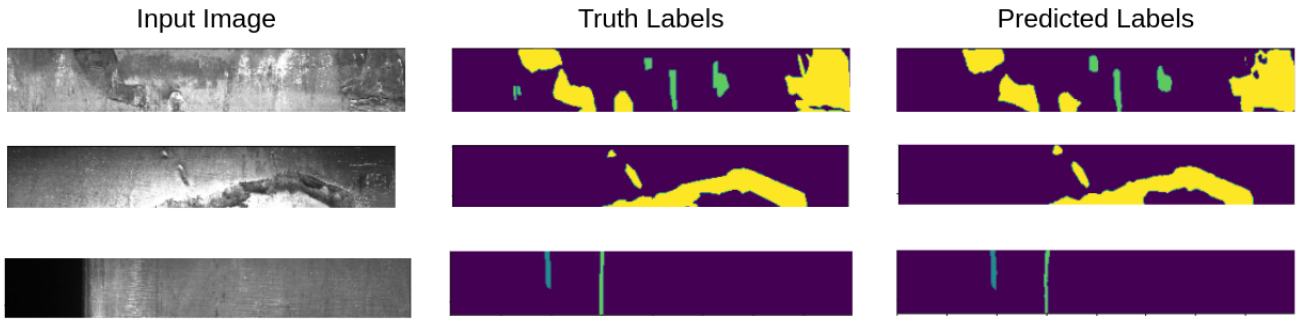


Figure 8: Results of Steel Surface Defect Detection Project

8 Conclusion

This covers some of the basics required for performing semantic segmentation for autonomous vehicles. Concepts could be built on this to make a more accurate and less time-consuming models which can provide better frame rates on processing live feeds.

-x-x-x-x-x- Thank You -x-x-x-x-x-