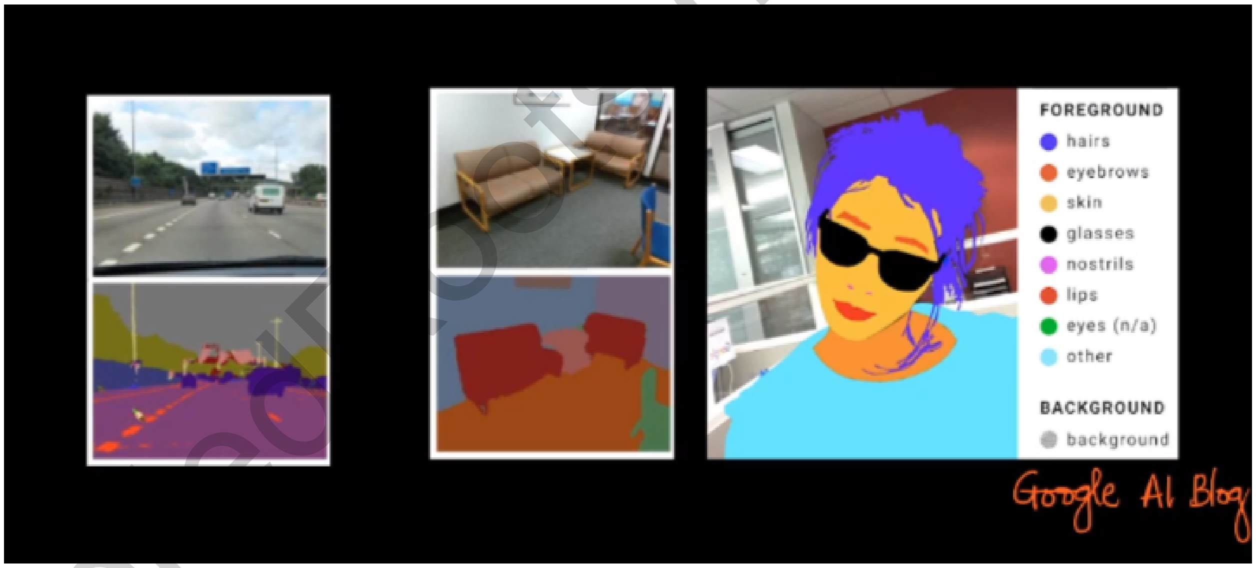
Image Segmentation using CNNs

In this session, we learn some more concepts which are very relevant in the context of Image segmentation in addition to what we have learned earlier. Semantic Segmentation is a modern term for Image segmentation.
Unet paper was published in late 2015, which is actually designed for biomedical image processing. In Unet there is an interesting concept called Upconvolutions.

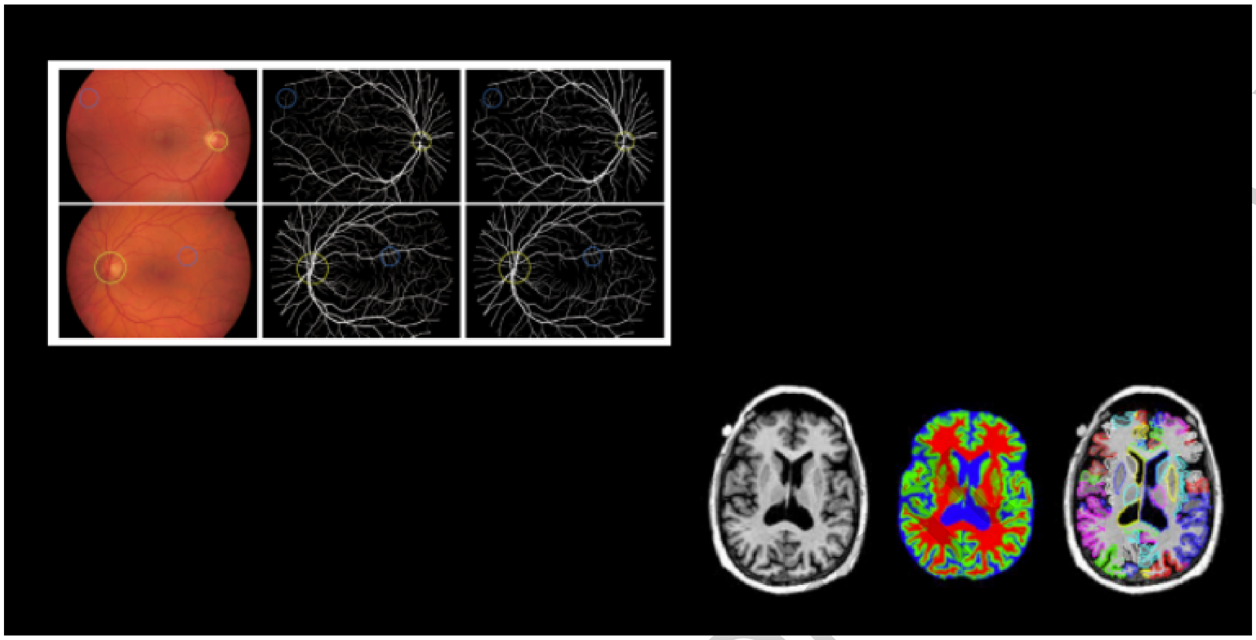
Applications:

Given an input image which could be a colored or greyscale image, we want to color each of the logical segments in the image with one unique color.



The important point here is Input and Output both are images.
Image segmentation is an important part of self-driving cars.

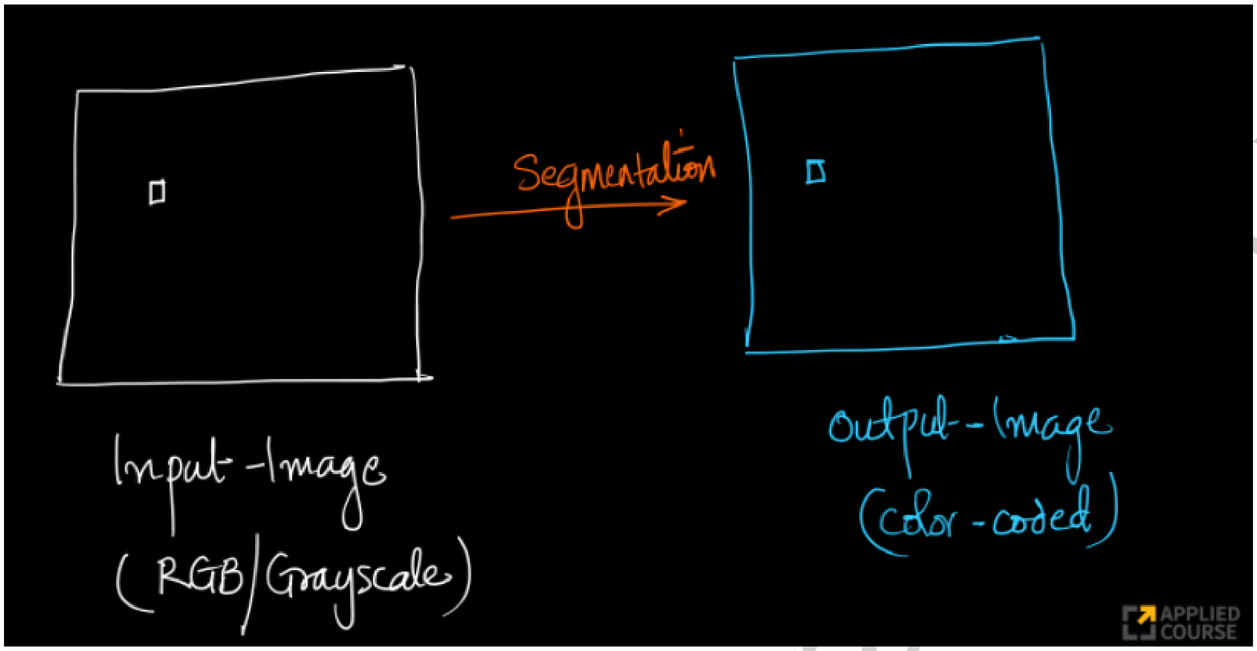
Another important application of Image segmentation is Medical diagnosis. If we have an input image of an eyeball, we need to find the number of the nerves inside it because based on the nerves or clusters, we can detect the strain or disease.



The main reason image segmentation is powerful is because of its visualization. Imagine we need to detect the brain cancer of patients, we can train resnet based classifier to tell whether yes or no. In addition to that, the doctor must know the problematic part that is the reason for cancer. So, using unet it is possible to pinpoint and highlight that region that is causing the problem.

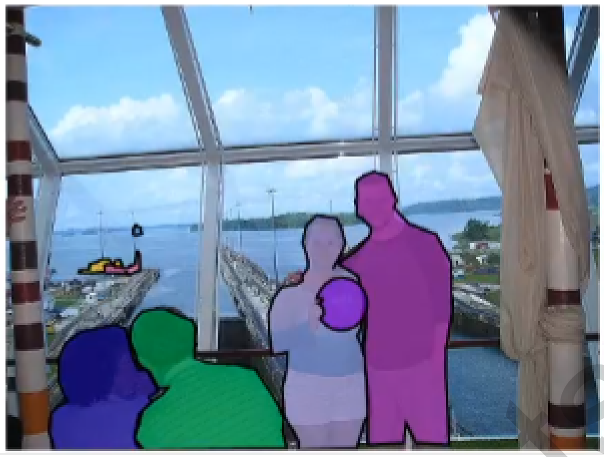
Problem definition:

We have an input image either grayscale or color image, we have to generate the image with color coding for each pixel in the input image.
Obviously, each color has one importance.
So, Basically logic is, Given an image, we want to generate an output image where each pixel belongs to multiple classes. And these multiple classes can be visually observed by using color-coding.



Datasets:

For this problem, there are numerous datasets.
One of the most popular datasets is the coco dataset. Just like the way Imagenet dataset is phenomenal for the image classification task, which is a great innovation from Stanford University.
http://cocodataset.org/#home
This coco dataset has 1.5million objects which are labeled. The data is collected by top research institutes and institutions. Given the whole image, it gives all the labels present in the image.



This coco dataset is a general-purpose dataset. There are a bunch of other datasets that are domain-specific.

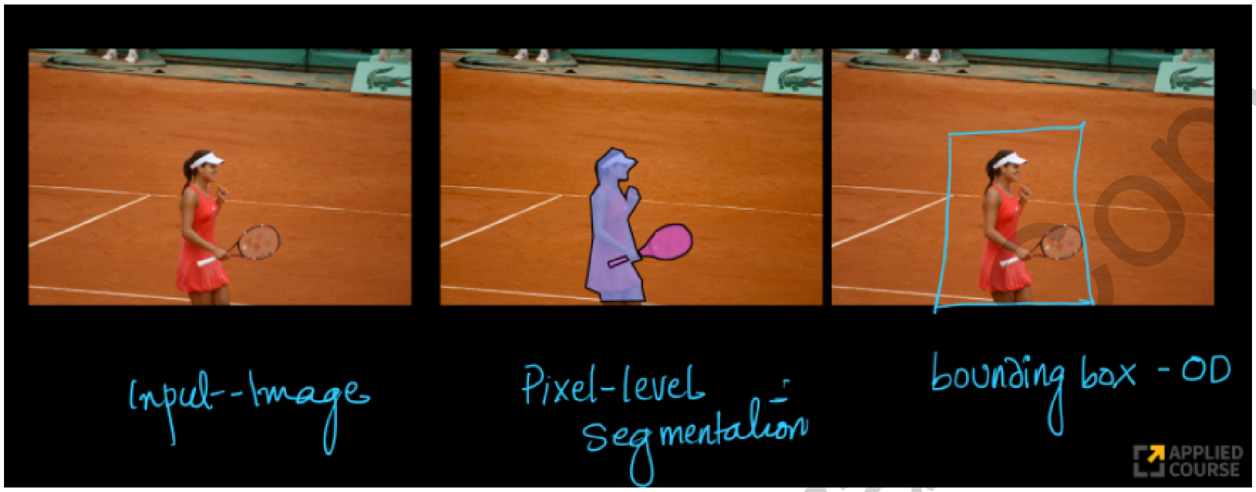
For example, this is the popular dataset called the cityscapes dataset.
https://www.cityscapes-dataset.com/examples/
This is specifically designed for designing self-driving cars.
So, Given image and class labels, proper segmentation can be obtained.

Similarly, there is another powerful dataset that is based on the medical domain.https://www.kaggle.com/mateuszbuda/lgg-mri-segmentation/version/2

Segmentation versus Object Detection

Let us take the image of a tennis player taken from the coco dataset.

Pixel level segmentation is representing the labels for person and tennis racket. While object detection detects where the object is present. This can also be done by bounding boxes.

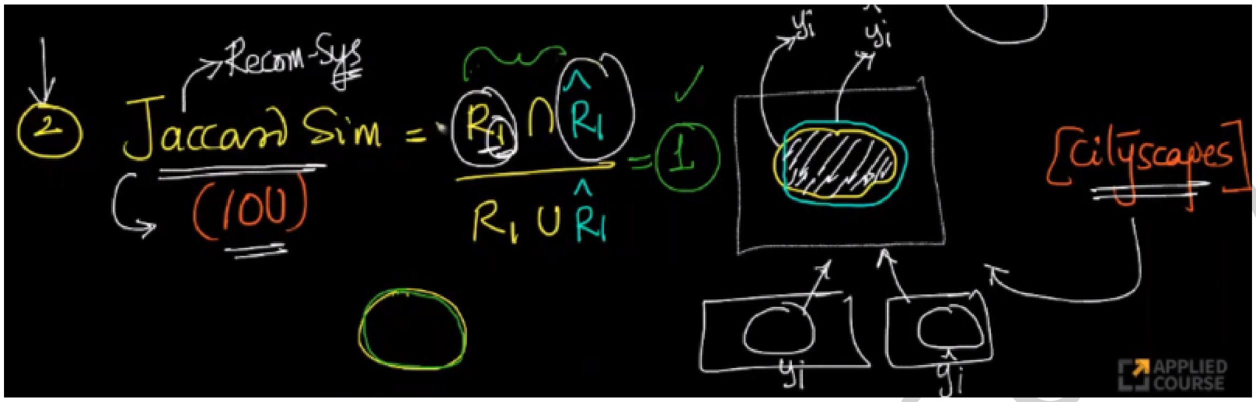


Instead of creating every pixel for the class label, It outputs a box saying within the box there is a person.
Sometimes, Segmentation and Object Detection terms are used interchangeably. Sometimes object detection is called object localization.
Bounding box-based Object Detection is faster as we don't need to worry about the pixel details. So this is mostly used in self-driving cars.

Performance metric

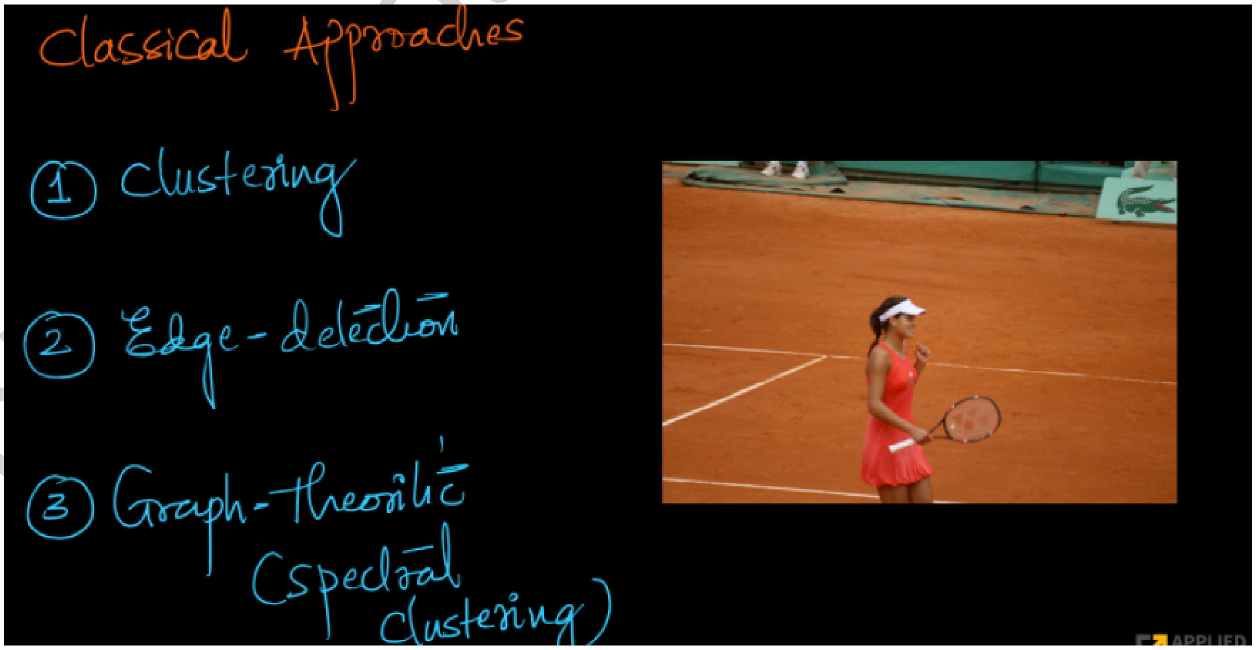
How to measure if the model is working properly?
Suppose xi is the input image and yi is the labeled output image and yi_hat is the predicted image. How to compare if the yi and yi_hat are equal?
There are two performance metrics that are widely used for this purpose.

1. Average precision and Recall for each class or color.
   This is used in the coco dataset
2. Jaccard Similarity
   This is also referred to as IOU Score [which means intersection over Union]. A lot of modern research papers and practical applications use this score.



In this, we look at the overlapping parts in both the labeled and predicted images for the Intersection part.
When the predicted image yi_hat is closer to the yi, the score comes closer to 1.
So the ideal score where yi_hat and yi will be exactly similar will be 1 and the worst score will be 0.
So, compute this term for multiple regions that we have in the image and calculate the average of this.

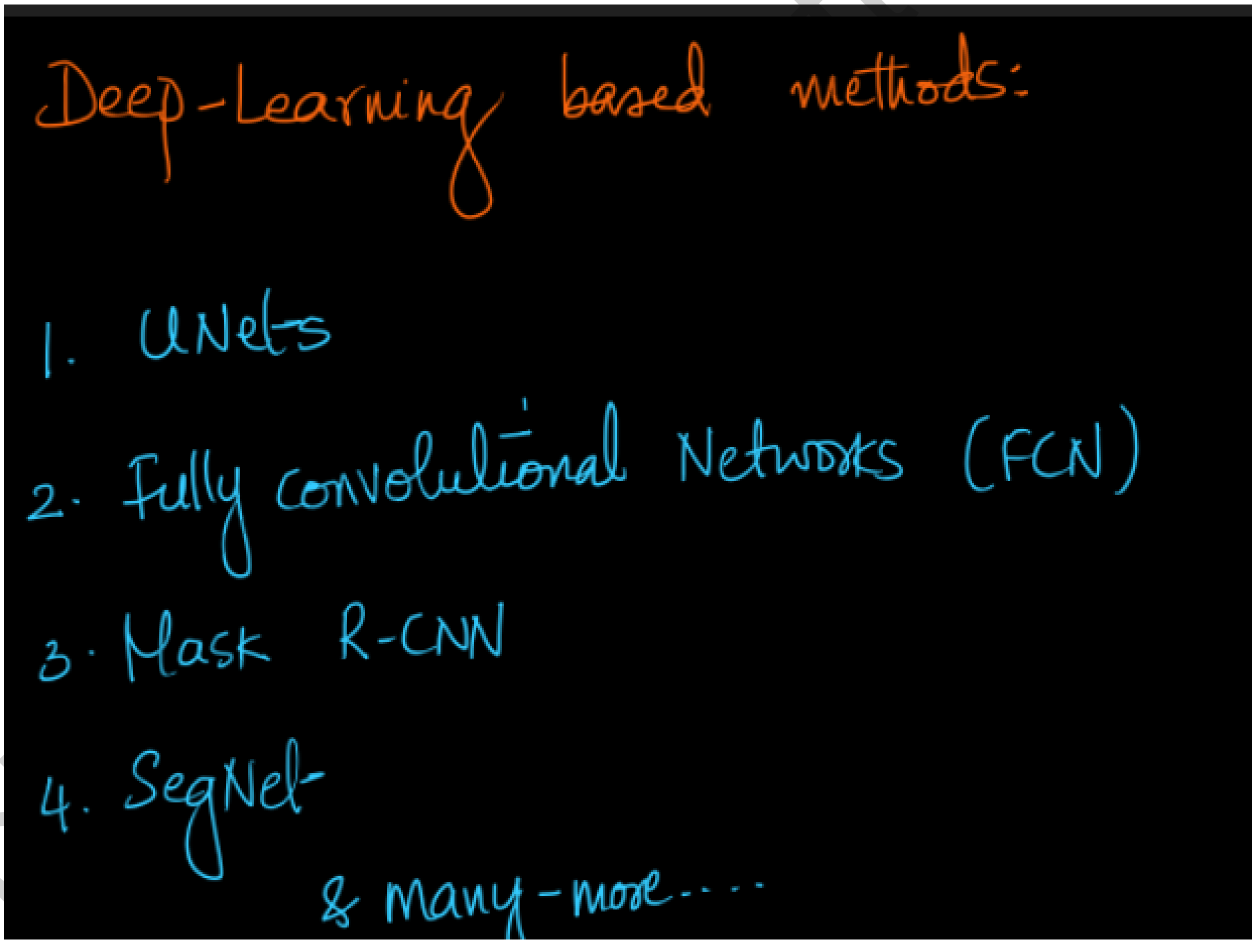
Classical Approaches:



Now we have to find the model to solve the problem.

1. Simple clustering algorithms.
   This can easily mess up the colors in the image.
   So, it is not recommended.
2. Edge detections
   It can also not deliver correct results when there are similar colors for different objects. For example, the hand and ground are so similar in color.
3. Graph-theoretic or spectral clustering

All these are the models which were being used before deep learning.



Deep-Learning based methods:

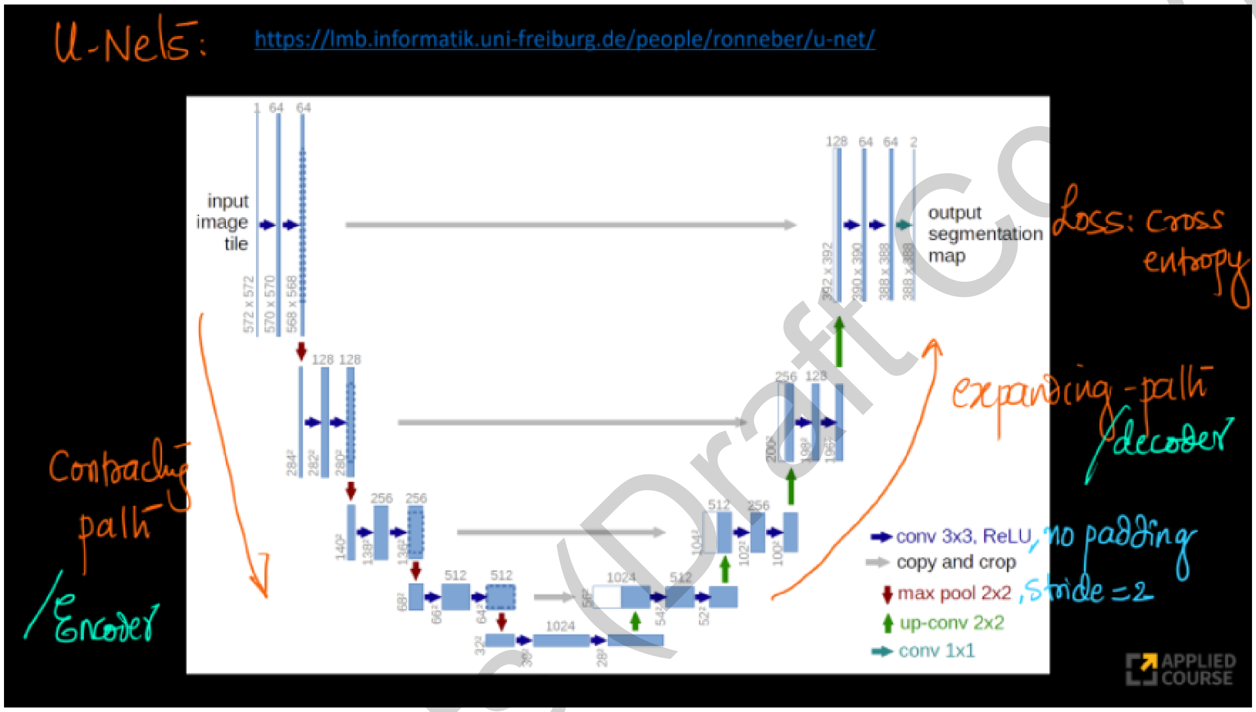
1. UNets
2. Fully convolutional Networks (FCN)
3. Mask R-CNN
4. SegNet

& many-more....

There are tons of models in deep learning. You can visit scholar.google.co.in and search for semantic segmentation in deep learning. You can get numerous research papers that were published for Image segmentation.

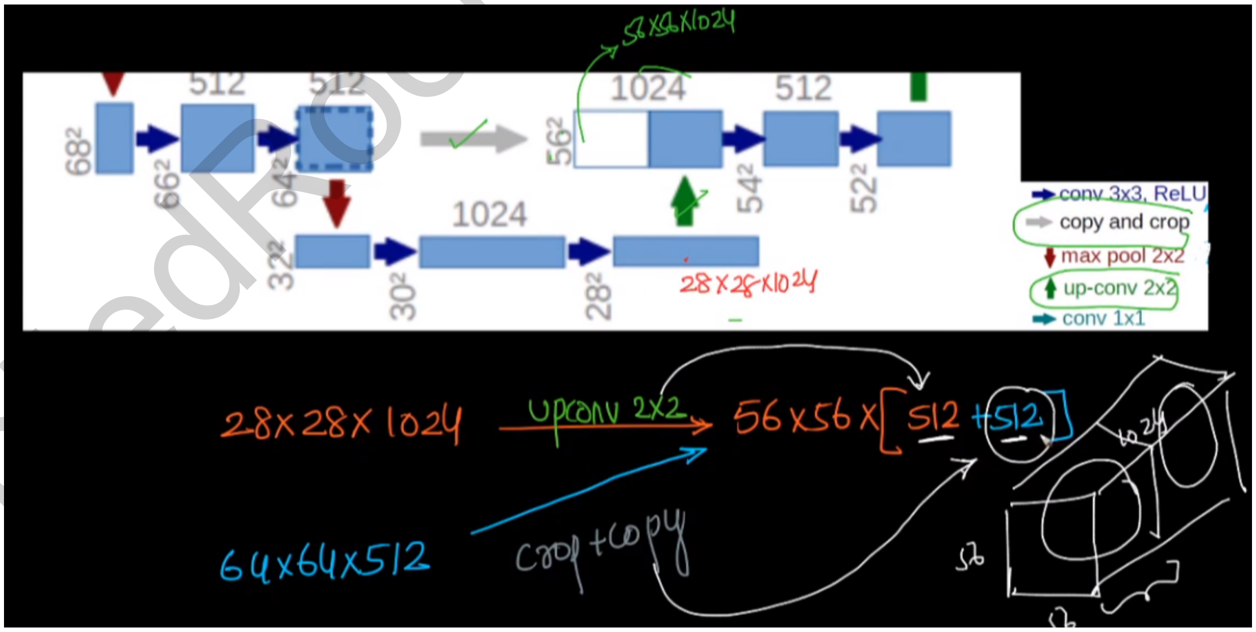
Segments are innovated for specific domain purposes.

We cover more deep concepts in U-Nets.
This is the structure of U-Nets.



This is called UNet because of the U-shape.
Copy and crop and Up-conv are the new terminologies here to learn.
Architecture:
On the dataset provided by Freiburg.
The size of the input image is 572*572 with depth is1 as the image is a grey-scale image.
The blue arrow represents 3*3 convolution, RELU activation, and no padding.
Such 64 3*3 convolutional kernels are created by which we get 570*570*64 [572-3+1=570] tensor. From this again 64 convolutions of size 3*3 are used. This again reduces the shape to 568*568*64 after 2 convolutions.
The orange arrow represents max pool 2*2 i.e max pool with stride 2.
After this, the 568*568*64 tensor becomes exactly half, which means 284*284*64.

Again we perform 2 convolutions with 128 kernels, which give the tensor of size→ 280*280*128.
Again we perform the max pool which results in tensor of size→ 140*140*256.
Again 2 convolutions→ 136*136*256
Here, when we came to size down, the image size is getting half but the depth is doubling.
We carry on the same for another 2 times and the final output is 28*28*1024.
After this in traditional paper, we have a bunch of FC layers and then a softmax layer.
Till now, this path is the contracting path. And we again start expanding the image and the path is called the expanding path of the Unet.
The contracting path is also called the encoder and the expanding path is also called the decoder.
At the beginning of the decoder, we encounter the green arrow which is up convolution, and the grey arrow called copy and crop.



The simplest method for up Conv is duplicating every pixel. This makes the 28*28 tensor to 56*56 tensor.

The crop operation only crops the 56 pixels from the 64*64 tensor. So we just cropped the central part and copied using the crop and copy operation.
As we are going up the expanding path, we are also taking some information from previous layers. These are sort of like skip connections like in the Resnets, as, here also there is the flow of information from the previous to the next layer.

So, In Unet, when a 572*572 image is given as input, 388*388 image is generated as output.

We can take the loss metric as Categorical cross-entropy as it is multiclass classification.

You can refer to the code for the Unet architecture implementation.
https://github.com/zhixuhao/unet/blob/master/model.py#L33

Data Augmentation:

In Transfer learning videos, we have used data augmentation techniques.
The core idea is, in addition to the raw data, we must use augmentation techniques to make the model perform better.

Please go through these blogs about Unet to have a better understanding.

https://github.com/zhixuhao/unet/blob/master/model.py#L33

https://github.com/divamgupta/image-segmentation-keras

https://ai.googleblog.com/2018/03/mobile-real-time-video-segmentation.html