**Fully Convolutional Network**

This helps preserve special information throughout the entire network. This can work on images of any size.

*A Fully Convolutional Neural Network preserves spatial information.*

**Techniques in FCN**

1. Replaces Fully Connected layers with 1 by 1 convolution layers.
2. Upsampling through the use of transposed convolutional layers.
3. Skip connections🡪 this allows the network to use information from multiple resolution scale.

**The structural part of FCN**

**Encoder**: A series of convolutional layer such as (vgg and resnis), it extracts features from the image.

**Decoder**: This upscales the output of the image such that it has the size as the original image. (segmentation and prediction of each image pixel).

**Transposed Convolution/deconvolution**

For this, the forward and the backward patches are swapped.

**Skip Connections**

In convolution, we usually narrow down the pixel by looking closely at some features and somewhat lose the bigger picture. However, during decoding, some of the information might have been lost already, therefore, skip connections is a way of retaining the information easily. Skip connection works by connecting the output of one layer to a non-adjacent layer.

**Semantic Segmentation**

This involves assigning meaning to a part of an object. This can be done at the pixel level by assigning every pixel to a target class. This helps us to write valuable information about every pixel in the image. In this, we often want to input an image into a neural network and we as well want to output a category for every pixel in the image.

*This is also called scene understanding*

One approach of scene understanding is training multiple decoders, and each decoder is trained on a separate task, one segmentation decoder and depth measurement decoder. This network will not predict the pixel call but also how far the object is.

**Prediction in the FCNN**

This output Tensor of predictions contains values that classify every pixel in the image, so if we look at a single pixel in this output, we would see a vector that looks like a classification vector -- with values in it that show the probability of this single pixel being a cat or grass or sky, and so on. We could do this pixel-level classification all at once, and then train this network by assigning a loss function to each pixel in the image and doing backpropagation as usual. So, if the network makes an error and classifies a single pixel incorrectly, it will go back and adjust the weights in the convolutional layers until that error is reduced.

**Limitations of the Approach**

* It's very expensive to label this data (you have to label every pixel), and
* It's computationally expensive to maintain spatial information in each convolutional layer