Summary of *Fully Convolutional Networks for Semantic Segmentation*

The author has built a fully convolutional, which can take input of arbitrary size and produce a corresponding output with more efficiency, trying to make a prediction at every pixel. They trained the network end-to-end, pixels-to-pixels on semantic segmentation. It is the first work to train FCNs for pixelwise prediction and from supervised pre-training.

The way they conduct a Transformation from recent success to dense prediction is to reinterpreting classification nets as fully convolutional and fine-tuning from their learned representations.

At first, they review related works on deep classification nets, FCNs, and recent approaches to semantic segmentation using convnets.

Second, they compare the general deep net with FCN theoretically. And they discuss a way to simplify the loss function with receptive fields. They also proffer the way how they convert a classification net to a fully convolutional net. They introduce a trick for fast scanning with shift-and-stitch in 3.2, deconvolution layers for upsampling in 3.3, patch-wise training in 3.4.

Third, they conclude and compare their results in section 4. And they give an introduction to their experimental framework. The *Optimization, Fine-tuning,* and *More Training Data* contributes to a better result, while *Patch sampling, Class Balancing, Dense Prediction* and *Augmentation* don't make a difference.

My understanding about FCN is as follows. In our semantic segmentation work, we want to obtain an output mask, which has the same width and height as our input image, while the depth of the mask refers to the number of classes that we really interested in. We are trying to predict which class every individual pixel belongs to. The encoder from previous Networks focuses on the character of objects, like VGG 16 or ResNet. If we want to reconstruct the width and the height of the input image after the last pooling layer (pool 5), we have to upsample 32 times, that is why the mask looks very coarse, which means a lot of information have been lost when we pooling down. The way we reserve the information is that we add the information from pool 4 and 2* pool 5, which means we need only 16 times of upsampling. And the meaning of FCN-32s is that we have to use a stride of 32 by 32 to reconstruct that. And it is the same when we add 2* (pool 4 + 2* pool 5) and pool 3, which we finally need to upsample with 8 by 8 stride (FCN-8s). And the way we implement the upsampling is deconvolution.