

U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies,
University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: <http://lmb.informatik.uni-freiburg.de/>

Abstract. There is large consent that successful training of deep networks requires many thousand annotated training samples. In this paper, we present a network and training strategy that relies on the strong use of data augmentation to use the available annotated samples more efficiently. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. We show that such a network can be trained end-to-end from very few images and outperforms the prior best method (a sliding-window convolutional network) on the ISBI challenge for segmentation of neuronal structures in electron microscopic stacks. Using the same network trained on transmitted light microscopy images (phase contrast and DIC) we won the ISBI cell tracking challenge 2015 in these categories by a large margin. Moreover, the network is fast. Segmentation of a 512x512 image takes less than a second on a recent GPU. The full implementation (based on Caffe) and the trained networks are available at <http://lmb.informatik.uni-freiburg.de/people/ronneber/u-net>.

useful for
Small
datasets

Reduce
the size

Medium
size

In this paper, we build upon a more elegant architecture, the so-called “fully convolutional network” [9]. We modify and extend this architecture such that it works with very few training images and yields more precise segmentations; see Figure 1. The main idea in [9] is to supplement a usual contracting network by successive layers, where pooling operators are replaced by upsampling operators. Hence, these layers increase the resolution of the output. In order to localize, high resolution features from the contracting path are combined with the upsampled

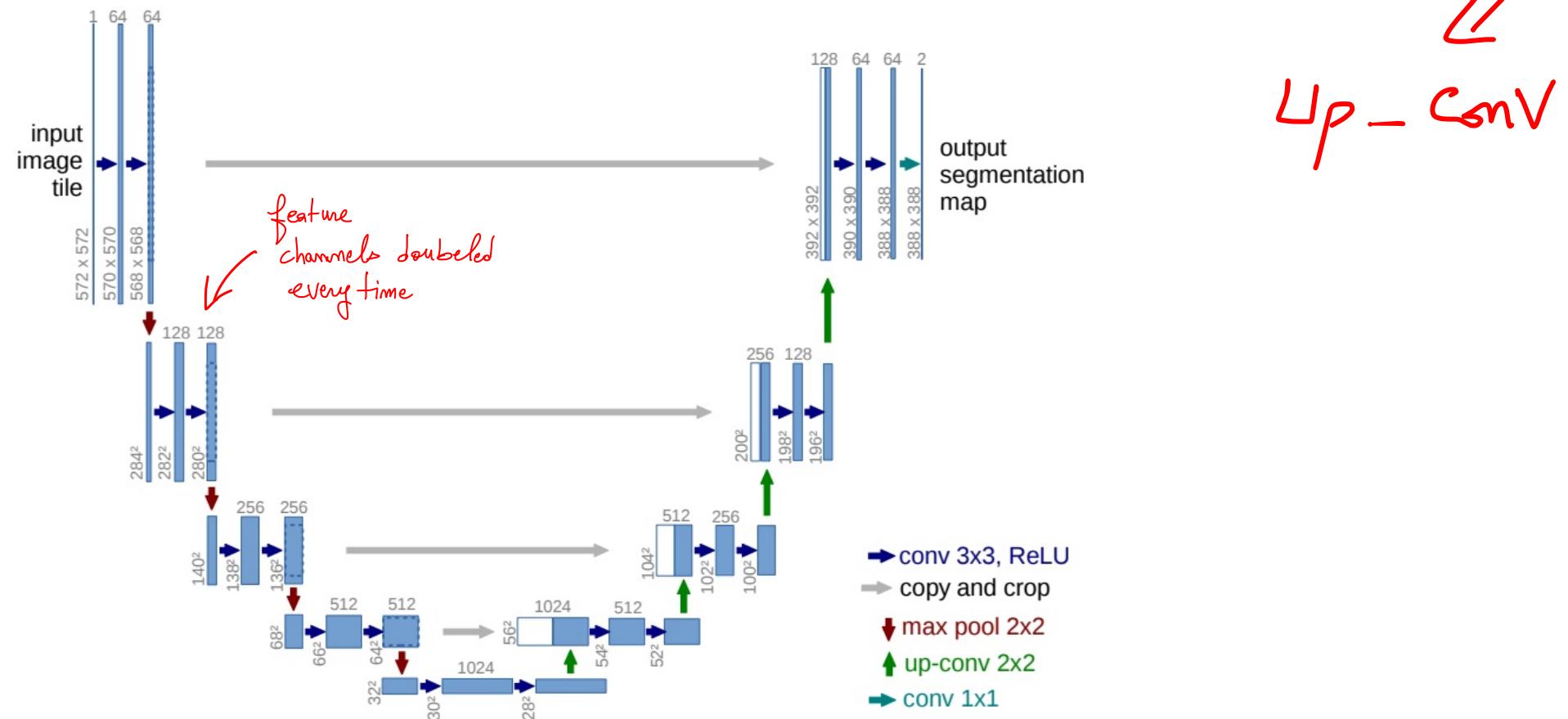


Fig. 1. U-net architecture (example for 32×32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

output. A successive convolution layer can then learn to assemble a more precise output based on this information.

One important modification in our architecture is that in the upsampling part we have also a large number of feature channels, which allow the network to propagate context information to higher resolution layers. As a consequence, the expansive path is more or less symmetric to the contracting path, and yields a u-shaped architecture. The network does not have any fully connected layers and only uses the valid part of each convolution, i.e., the segmentation map only contains the pixels, for which the full context is available in the input image.

→ A fully connected (dense layer) takes a 1D vector and outputs another 1D vector.

But in image segmentation, you want:

- pixel-wise output
- same spatial structure as the input
- locally preserved (each pixel classified independently)

If you add a dense layer :

- You must flatten the feature maps
→ You destroy the spatial information.
- The output does not match the 2D/3D structure anymore.

One important modification in our architecture is that in the upsampling part we have also a large number of feature channels, which allow the network to propagate context information to higher resolution layers. As a consequence, the expansive path is more or less symmetric to the contracting path, and yields a u-shaped architecture. The network does not have any fully connected layers and only uses the valid part of each convolution, i.e., the segmentation map only contains the pixels, for which the full context is available in the input image.

↙ valid convolution: no padding is used.

border pixels are removed \Rightarrow only pixels with full context remain.

This strategy allows the seamless segmentation of arbitrarily large images by an overlap-tile strategy (see Figure 2). To predict the pixels in the border region of the image, the missing context is extrapolated by mirroring the input image. This tiling strategy is important to apply the network to large images, since otherwise the resolution would be limited by the GPU memory.

→ We cut the large image into smaller patches.
These patches overlap, not split cleanly.

Why overlapping?

because UNet uses Valid Conv.
→ output shrinks

If you cut tiles without overlap, the predicted segmentation maps would have gaps between tiles.

When predicting pixels near the border of a tile UNet again needs full context (full receptive field)

⇒ we use mirror padding

⚠ mirror padding
in preprocessing

∅ - padding in conv.

2 Network Architecture

The network architecture is illustrated in Figure 1. It consists of a contracting path (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3×3 convolutions (unpadded convolutions) each followed by a rectified linear unit (ReLU) and a 2×2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels. Every step in the expansive path consists of an upsampling of the feature map followed by a 2×2 convolution (“up-convolution”) that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3×3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a 1×1 convolution is used to map each 64-component feature vector to the desired number of classes. In total the network has 23 convolutional layers.

It is applied element-wise to each pixel or feature in the output map.

To allow a seamless tiling of the output segmentation map (see Figure 2), it is important to select the input tile size such that all 2×2 max-pooling operations are applied to a layer with an even x- and y-size.