U-Net

The typical use of convolutional networks is on classification tasks, where the output to an image is a single class label. However, in many visual tasks, especially in biomedical image processing, the desired output should include localization, i.e., a class label is supposed to be assigned to each pixel. Moreover, thousands of training images are usually beyond reach in biomedical tasks.

↓ biomedical task를 해결하기 위해서는 output이 이미지의 localization 하나의 class label으로 할당하는 것이 아니라 "위치정보"도 포함할 수 있어야 한다.
또한, 근본적으로 biomedical task를 해결하기 위해서는 수백, 수천개의 학습 데이터를 적용해야 하기 때문에 해당 논문의 저자는 data augmentation을 많이 진행했다고 논문 이후의 부분에 등장한다.

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 net-

① ② ↳ 기존에 사용하던 sliding window와 달리 end-to-end의 방식 사용

↓ 위에서 네트워크의 구조를 설명하는 과정에서 알 수 있겠지만 U-Net은 두개의 path로 구성된다.

① contracting path
– capture context

② expanding path
– enables precise localization

cf. FCN (Fully Connected Network)

Convolution만 이용한 기존의
이미지분류 모델
(AlexNet, VGG16, GoogleNet)
Image Classification Model

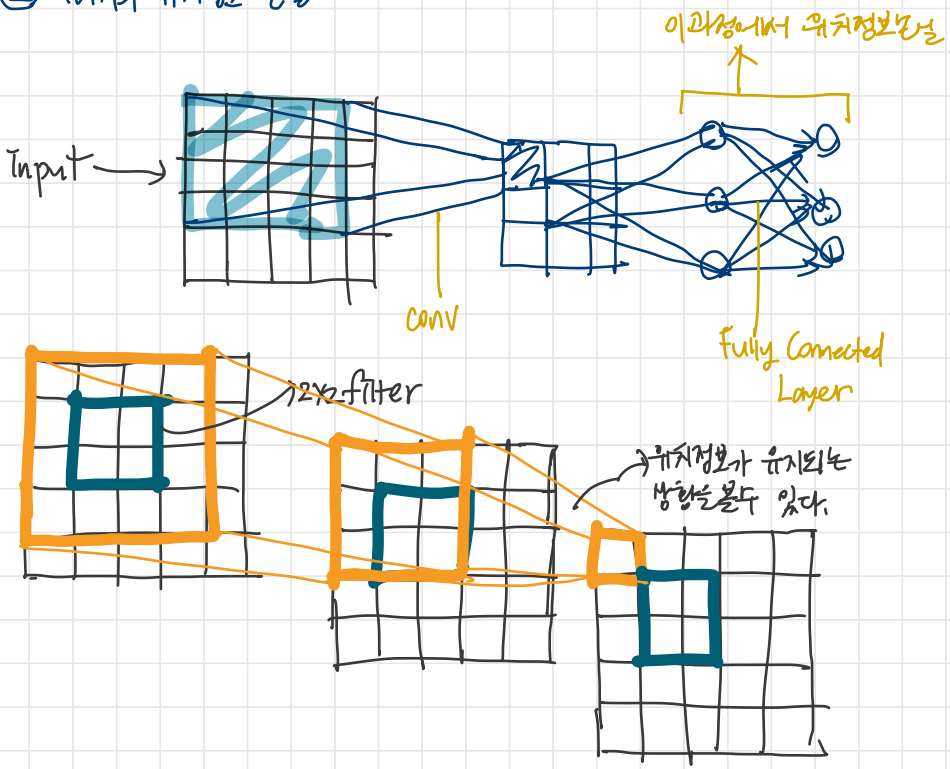
→ Semantic Segmentation Model
- convolutionalization
- Deconvolution (upsampling)
- Skip Architecture

1) convolutionalization

· 기존의 image classification 모델들은 학습과정에서 이전에 입력받은 특징벡터를 (conv에 의해 추출된) fully connected layer에 넣어서 classification 분류를 진행함.

그래서 이는

① 이미지의 위치정보 손실



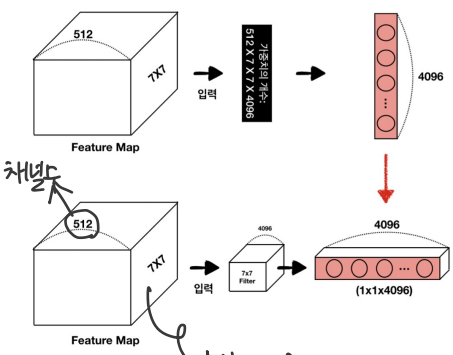
② 입력 이미지의 크기의 고정

fully-connected layer(완전연결층)로 사용되는 Dense layer을 사용하기 때문에

Dense layer에는 가중치개수가 고정되어있기 때문에 이전 layer의 feature map의 크기 또한 고정되며, 따라서 입력이미지의 크기도 고정되어진다.

그러나 segmentation의 경우에 목적이 한장의 사진을 각각 pixel단위로 구분하고 배경을 분할하는 것이기때문에 위치정보의 정보가 매우 중요하다.

✳ 이러한 fc layer의 한계를 극복하기 위해서 모든 fc layer을 convolution layer로 대체하였다. 즉, feature map이 너무 coarse하다는 한계를 극복했다.

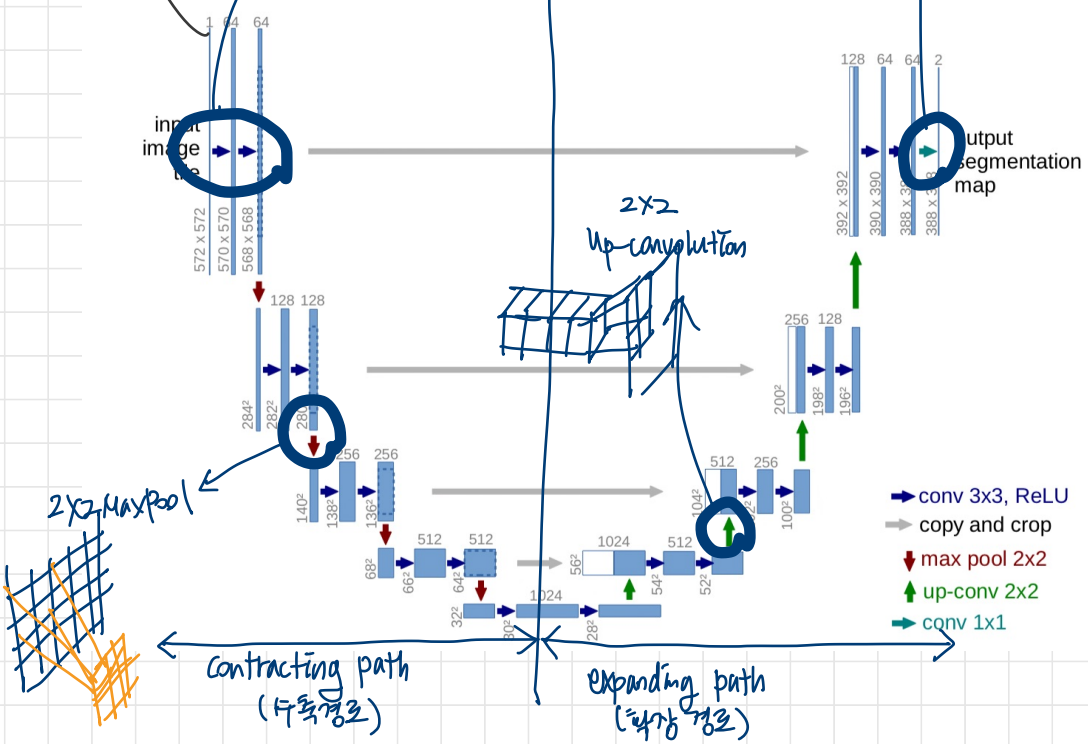


입력 feature map의 width, height와 같은 크기의 filter을 사용해서 4096개의 가중치개수를 유지할수있게 해줌

각각의 파란색 네모는
multi-channel
feature map
을 의미한다.

3x3 conv를 반복 진행

1x1 conv



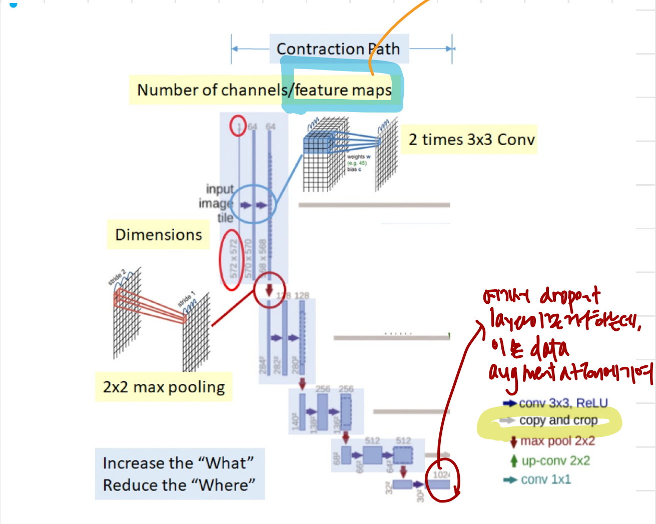
① 이미지의 feature map에서 context 추출

② Downsampling 과정을 거치면서 feature map 생성

① 정확한 localization을 수행할 수 있게

② 이전 layer의 정보를 concatenate하여 사용
∵ upsampling을 진행할 때의 feature map이 정보가 많이 손실되는 단점을 극복할 수 있게 된다

여기서 concat을 주목

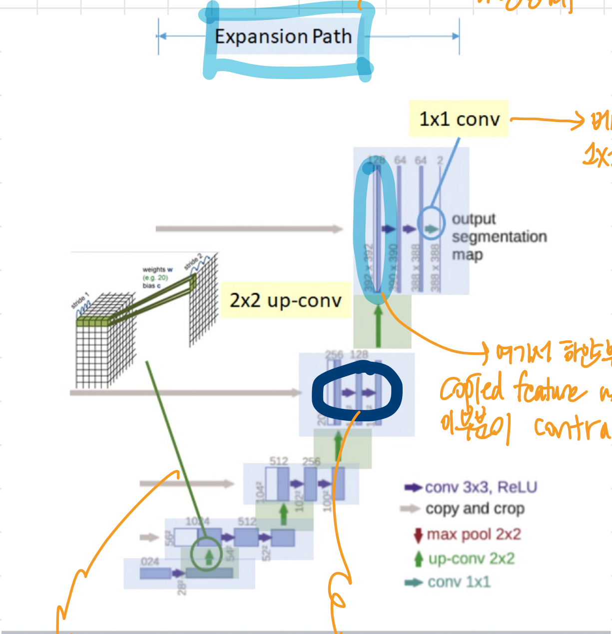


여기서 dropout layer이 추가되는데, 이는 data augmentation이기에

① 3x3 convolution layer에 의해서 그림의 특징벡터의 크기가 감소한다. 이는 padding=1이기 때문이다.

② 2x2 max pooling layer에 의해서 downsampling을 진행한다. padding=2이고 여기서 벡터의 크기는 절반으로 줄어들게 된다.

정확한 localization을 위한 단계



1x1 conv → 비선형 예측을 위해서 2x2 conv를 진행함

여기서 하얀부분이 바로 copied feature map을 의미하는데, 이부분이 contracting path의 crop된 특징맵이다.

① 2x2 conv에 의해서 feature map의 크기가 두배로 늘어나게 된다.

② 각 expanding step마다 up-conv된 특징맵은 contracting path의 cropped된 특징맵과 concat

채널의 수는 upsampling 할때마다 절반으로 감소

각 expanding step마다 3x3 conv를 두번반복하며, ReLU 활성화 함수를 이용한다.

∴ 총 23개의 층으로 이루어진 FCN구조이다.

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 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 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 2x2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 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 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total the network has 23 convolutional layers.

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 2x2 max-pooling operations are applied to a layer with an even x- and y-size.

FeatureMap
+
Cropped Feature map

3X3 conv 2번
↓
ReLU
↓
2X2 MaxPool 1번 (stride=2) — downsample (절반) / Feature Channel수 (X2)

2X2 conv

3X3 conv 2번
↓
ReLU
↓
1X1 conv
↓
64개의 feature vector를 원하는 class의 개수로 바꾸어줌

이전 path, 즉 크기줄이는 역할을 하던 path에서의 특징벡터의 테두리를 자른 특징맵이다

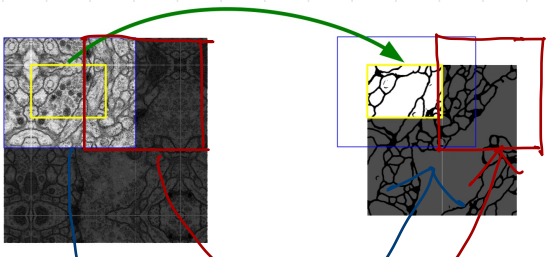


Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring

파란영역을 입력하면 노란영역의 segmentation 결과를 얻을 수 있다.
타일에 대한 segmentation 방법으로 연구하여서
이미지의 일부를 효율적으로 처리

Overlap-tile Strategy

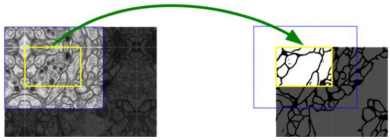
This strategy allows the seamless segmentation of arbitrarily large images by an overlap-tile strategy (see Figure 2). To predict the pixels in the border region of the image, the missing context is extrapolated by mirroring the input image. This tiling strategy is important to apply the network to large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

① fully-connected layer인 경우에는 커널의 특성상 입력이미지의 크기에 제약이 없다.
대신 보통의 처리는 크기가 큰 이미지의 경우에는
이미지경계를 사용하는 대신에 overlap-tile 방법을 사용했다.

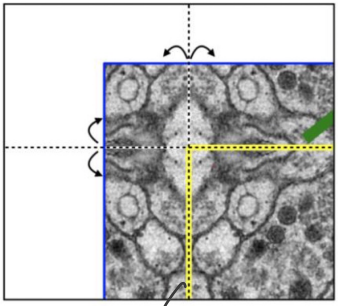
② 이미지를 patch로 나누어서 입력으로 사용한다.

③ 이미지의 경계부분의 pixel에 대한 segmentation을 위해서 0이나 일정의
padding은 사용하는 대신 이미지경계부분의 mirroring을 사용한
extrapolation 기법을 사용한다.

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 2x2 max-pooling operations are applied to a layer with an even x- and y-size.



Mirroring extrapolatation



Mirroring Extrapolation.

① 이미지의 경계부분 pixel에 대한 segmentation을 위해 0 이나 원본padding값을 사용하는 대신에 이미지 경계부분의 mirroring을 이용한 extrapolation을 사용한다.

↳ 여기서 오른쪽의 원본이미지인데, 이 원본이미지에 대해서 거울에 반사된것처럼 대칭이동해서 확장한다.

Another challenge in many cell segmentation tasks is the separation of touching objects of the same class; see Figure 3. To this end, we propose the use of a weighted loss, where the separating background labels between touching cells obtain a large weight in the loss function.

① 많은 cell segmentation 문제들에서 객체들사이의 (같은 class의) 닿아있는, 혹은 겹쳐있는 부분의 분리가 필요한 상황이다.

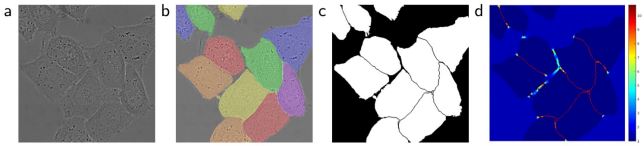


Fig. 3. HeLa cells on glass recorded with DIC (differential interference contrast) microscopy. (**a**) raw image. (**b**) overlay with ground truth segmentation. Different colors indicate different instances of the HeLa cells. (**c**) generated segmentation mask (white: foreground, black: background). (**d**) map with a pixel-wise loss weight to force the network to learn the border pixels.

따라서 객체간의 분리를 위해 가중치가 부여된 모델을 적용한다.

즉, 바깥의 배경 label에 대해서 (맞닿아있는 cell 사이의 경계 등에 해당하는 배경부분)을 가중치를 주어서 더욱 cell을 인지할 수 있도록 한다.

✱ momentum = 0.99 (관성)

↳ 높은값을 사용해서 이전단계에서 학습에 사용한 sample에 현재 optimization step도 영향력을 있도록한다.

✱ SGD를 사용 (확률적 경사 하강법) 해서 모델을 계산하고 이를 학습에 반영한다.

✱ Convolution Layer의 padding이 없다보니 output이 input보다 작아져, 따라서

✱ Gaussian Distribution을 이용해서 학습과정에 초기화 (=정규분포)

The energy function is computed by a pixel-wise soft-max over the final feature map combined with the cross entropy loss function. The soft-max is defined as $p_k(\mathbf{x}) = \exp(a_k(\mathbf{x}))/\left(\sum_{k'=1}^{K}\exp(a_{k'}(\mathbf{x}))\right)$ where $a_k(\mathbf{x})$ denotes the activation in feature channel k at the pixel position $\mathbf{x} \in \Omega$ with $\Omega \subset \mathbb{Z}^2$. K is the number of classes and $p_k(\mathbf{x})$ is the approximated maximum-function. I.e. $p_k(\mathbf{x}) \approx 1$ for the k that has the maximum activation $a_k(\mathbf{x})$ and $p_k(\mathbf{x}) \approx 0$ for all other k. The cross entropy then penalizes at each position the deviation of $p_{\ell(\mathbf{x})}(\mathbf{x})$ from 1 using

$$E = \sum_{\mathbf{x}\in\Omega} w(\mathbf{x})\log(p_{\ell(\mathbf{x})}(\mathbf{x})) \tag{1}$$

네트워크의 출력값은 픽셀별로 (단위시) softmax를 사용한다.

where $\ell : \Omega \to \{1,\dots,K\}$ is the true label of each pixel and $w : \Omega \to \mathbb{R}$ is a weight map that we introduced to give some pixels more importance in the training.

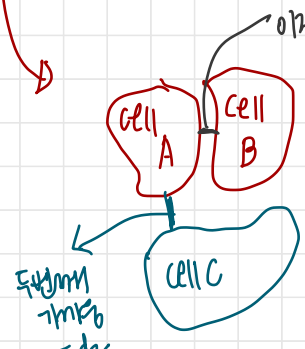
We pre-compute the weight map for each ground truth segmentation to compensate the different frequency of pixels from a certain class in the training data set, and to force the network to learn the small separation borders that we introduce between touching cells (See Figure 3c and d).

The separation border is computed using morphological operations. The weight map is then computed as

$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right) \tag{2}$$

where $w_c : \Omega \to \mathbb{R}$ is the weight map to balance the class frequencies, $d_1 : \Omega \to \mathbb{R}$ denotes the distance to the border of the nearest cell and $d_2 : \Omega \to \mathbb{R}$ the distance to the border of the second nearest cell. In our experiments we set $w_0 = 10$ and $\sigma \approx 5$ pixels.

① 각 픽셀마다 class분포과 빈도분포를 이용해서 ground-truth에 대한 weight map을 학습에 이용한다.



② 각각의 코너의 빈도를 계산한 weight map 또한 만들어서 이용한다.