U-Net : Convolutional Networks for Biomedical Image Segmentation

Abstract

- Deep networks : requires many annotated training samples
- Use Data augmentation to use available annotated samples more efficiently
- Architecture
  - Contracting path : capture context
  - Expanding path : enable precise localization
- Network : trained end-to-end
- Outperform sliding-window convolutional network

Introduction

- Deep CNN : limited
  - Size of available training set
  - Size of considered networks
- Typical use of CNN : classification (output to image : single class label)
- In biomedical image : desire output should include localization
  - Class label : assigned to each pixel

- Sliding-window setup : predict class label of each pixel by provide a local region(patch) as input
  1. Can localize
  2. Training data in terms of patch : larger than #training images

⇒ Drawbacks

1. Network run separately for each patch → quite slow & overlapping patches → a lot of redundancy
2. Trade-off between localization accuracy and use of context
   - Larger path : use of context ↑ /need more max-pooling layer → localization accuracy ↓
   - Smaller path : see only little context

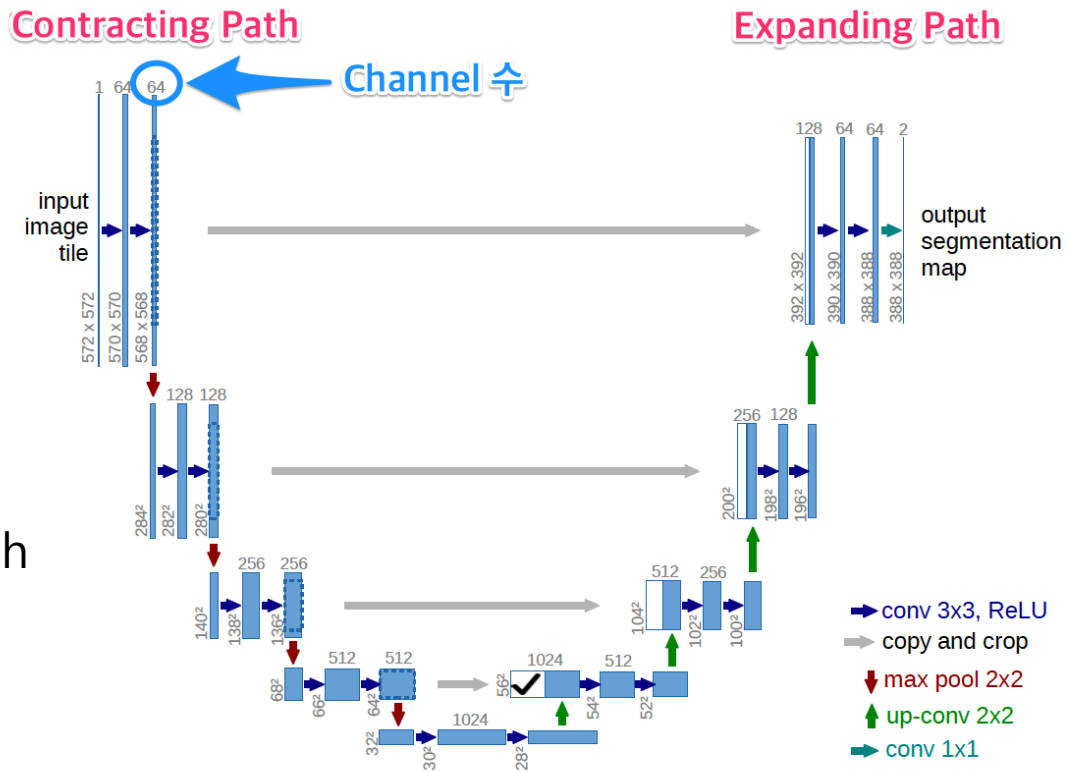
Introduction

	Localization accuracy	Use of context
Small patch	높음 -> patch 사이즈가 작기 때문에 max-pooling을 보다 적게 해도 됨	낮음 -> patch 사이즈가 작으면 전체 context의 더 적은 부분만을 보게됨
Larger patch	낮음 -> patch 사이즈가 크기 때문에 원하는 사이즈로 줄이기 위해 max-pooling을 더 많이 해야함 Max-pooling을 많이 할 경우, 본래의 사이즈보다 작아지기 때문에 localization accuracy가 줄어든다.	높음 -> patch 사이즈가 커서 더 많은 context를 한 번에 볼 수 있다. => global한 내용을 더 많이 고려할 수 있다.

Introduction

- Build on Fully convolutional network
  - → work with few training images
  - → yield more precise segmentation

- Fully convolutional network
  1. Contracting network
  2. Use upsampling operator instead of pooling operator
     - → increase output resolution
  - → to localize, high resolution feature from contracting path combine with upsampled output
  - ⇒ assemble more precise output



Introduction

- Import modification : upsampling part – a large number of feature channels
  - At downsampling : 64 → 1024
  - At upsampling : 1024 → 64
- Expansive path : symmetric to contracting path → u-shaped architecture
- No fully connected layer & only use valid part of each convolution

→ segmentation map only contain pixels ⇒ full context is available in input image

- Overlap-tile strategy
  - Allow seamless segmentation of arbitrarily large img
  - To predict border pixel

  → extrapolate by mirroring input img

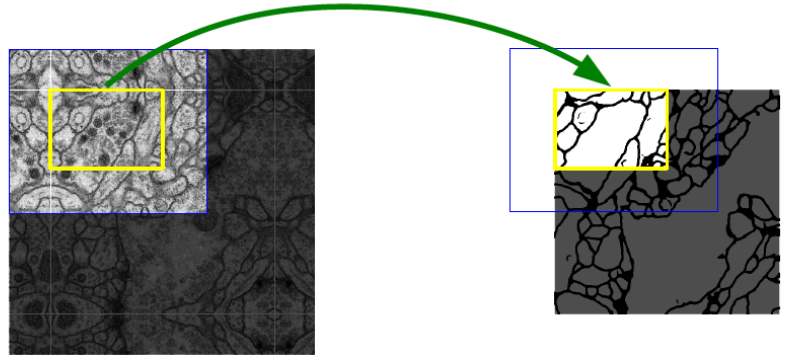
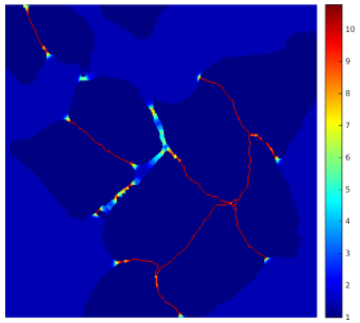


Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring

Introduction

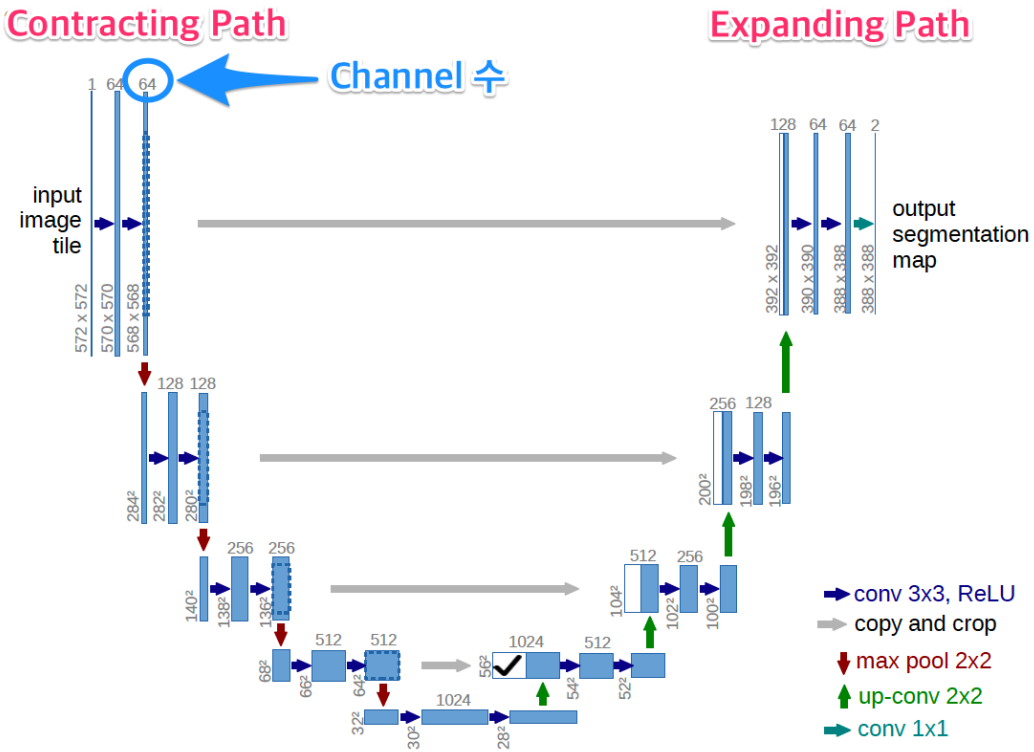
- Excessive data augmentation : apply elastic deformation
  - Network learn invariance to deformation (without annotation about these transformation)
  → important in biomedical segmentation (deformation : common variation in tissue)

- Another Challenge : separation of touching objects of the same class
  → propose use of weighted loss
  ⇒ separating background label between touching cells obtain large weight in loss function



Network Architecture

- Contracting path
  - Typical architecture of convolutional network
  - Two[conv(3X3, valid) → ReLU]⇒max pooling(2X2, s=2)
- Expansive path
  - Upconv(2X2, s=2) ⇒ concatenate from contracting path
  ⇒ two[conv(3X3, valid) → ReLU]
- Final layer : 1X1 conv
  - map each 64 feature vector to desired number of class
- Total network : 23 convolutional layer
- For 2X2 max pooling operations
  → select input tile size with even x- and y-size



Training

- Optimizer : stochastic gradient descent
- To maximize use of GPU memory : large input tile size (rather than large batch size)
  - Small batch size→ momentum : 0.99 (reflect previous value more)
- Loss function : softmax cross entropy
  - Softmax : $p_k(x) = \frac{\exp(a_k(x))}{\sum_{k'=1}^{K} \exp(a_{k'}(x))}$ $a_k(x)$: channel k activation
    K : # classes
    $p_k(x)$: approximated maximum-function
  - $E = \sum_{x \in \Omega} w(x) \log(p_{l(x)}(x))$
    : 정답의 추정값을 log에 사용 -> 해당하는 정답의 확률을 가져옴
    $l(x)$: 정답 class (softmax 수식에서 정답의 label에 해당하는 k값을 반환하는 함수)
  - $w(x) = w_c(x) + w_0 \cdot \exp(-\frac{(d_1(x)+d_2(x))^2}{2\sigma^2})$
    x위치에 해당하는 class의 빈도수에 따라 값 결정 → training data에서 x 픽셀이 background일 경우가 많은가 foreground일 경우가 많은 가의 빈도수에 따라 결정
    x위치의 픽셀에 가중치를 부여하는 함수
    d_1 : x에서 가장 가까운 세포까지의 거리
    d_2 : x에서 두번째로 가까운 세포까지의 거리
  - In this experiments : $w_0 : 10$, $\sigma \approx 5$ pixels

Training

- x : 세포 사이에 존재하는 pixel
  - 두 세포 사이의 간격이 좁을 수록 weight 큼
  - 두 세포 사이가 넓을수록 weight 작음

- Network parameter initialization : He 초기화 사용
  - 위 architecture가 conv와 ReLU를 반복하고 있기 때문

Training – Data augmentation

- Data augmentation : network에게 desired invariance와 robustness properties를 가르쳐야 하는데, available한 training sample이 많지 않을 때 효율적
  - Microscopy 이미지의 경우 : shift와 rotation하고, gray value도 조절해준다
  - Random elastic deformation of training sample하는 것이 아주 적은 labeling된 image만 갖고 segmentation network를 학습시킬 때 매우 중요하다
  - Deformation : 3X3 grid에서 random displacement vector로 elastic 변환 행렬을 통해 수행
  - Displacement : 가우시안 분포(10pixel SD)로 sample
  - Pixel 당 displacement : bicubic interpolation
  - Contracting path의 맨 끝의 drop out layer : implicit data augmentation

- 세포를 segmentation 하는 것 → elastic deformation 적용 → 성능 향상에 매우 큰 역할

Experiments

1. Neuronal structure in EM recording
   Training data : 30 image (512 X 512 pixel) : annotated ground truth 같이 있음
   Evaluation 지표 : warping error / rand error / pixel error 사용

Conclusion

- U-net : 매우 다른 biomedical segmentation application에서 좋은 성능을 보임
  - Elastic deformation을 적용한 data augmentation 덕분!
  - Annotated image가 별로 없는 상황에서 매우 합리적!

- Image Segmentation task 에서 가장 많이 쓰임
  - U 자형 architecture와 fully convolution & deconvolution구조를 가지고 있음
  - 정확한 localization을 위해 contracting path의 feature를 copy and crop해서 expanding path와 concatenate 하여 upsampling 함