Student Project: Neuron Membranes Segmentation in Electron Microscopy (EM)

COMP9517 Computer Vision

T2 2019

Dataset Description

Image Segmentation

- Learning-based Pipeline for Computer Vision Tasks
- Machine Learning based Approach
- Deep Learning based Approach

Evaluation

- V^{Rand} and V^{Info}
- Cross Validation

Dataset Description

- Dataset: ISBI 2012 EM Segmentation Challenge dataset
- 30 ssTEM images provided with corresponding segmentation ground truths.

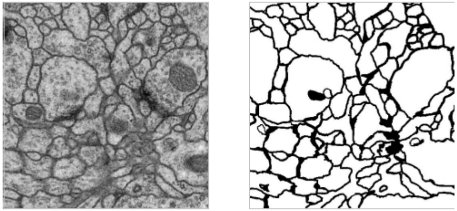


Figure: ssTEM image and its membranes segmentation GT.

Image Segmentation

- Learning-based Pipeline for Computer Vision Tasks
- Machine Learning based Approach
- Deep Learning based Approach

Image Segmentation - Learning-based CV Pipeline

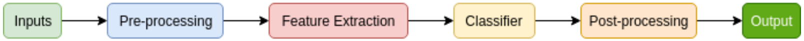


Image Segmentation - Learning-based CV Pipeline

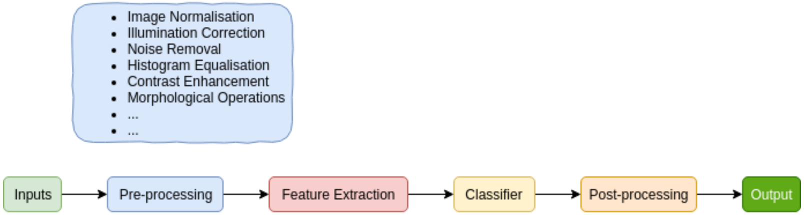


Image Segmentation - Learning-based CV Pipeline

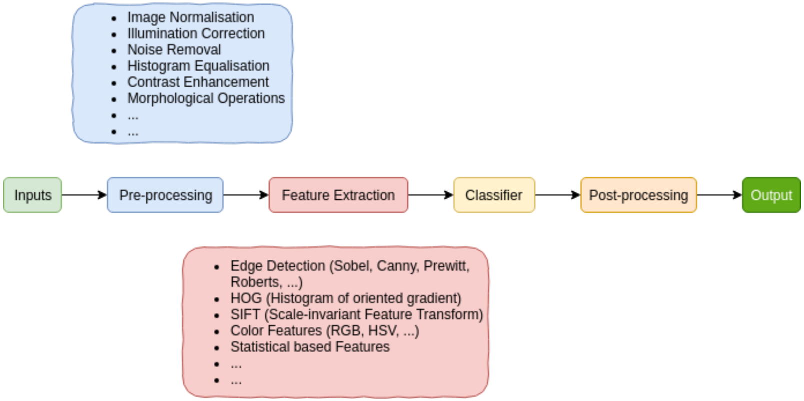


Image Segmentation - Learning-based CV Pipeline

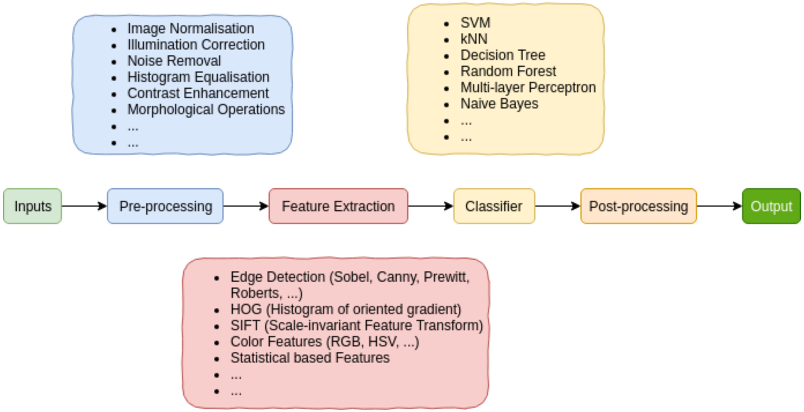
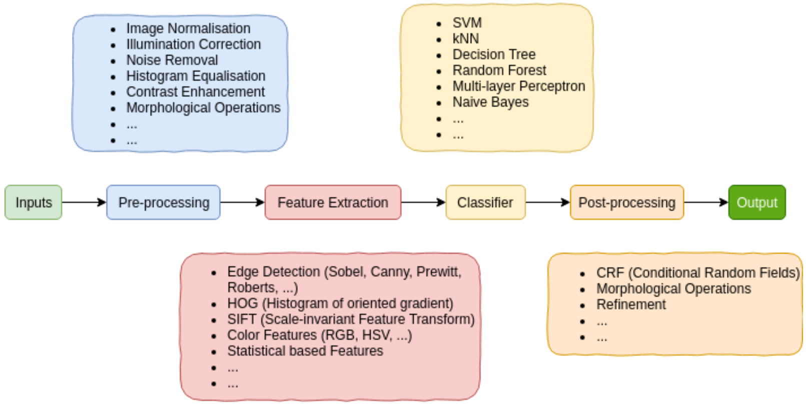


Image Segmentation - Learning-based CV Pipeline



Feature measures for the segmentation of neuronal membrane using a machine learning algorithm

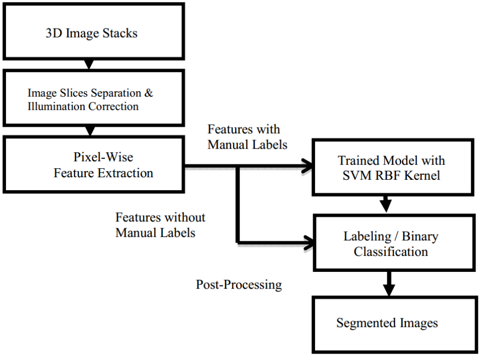

3D Image Stacks
Image Slices Separation & Illumination Correction
Pixel-Wise Feature Extraction
Features with Manual Labels
Trained Model with SVM RBF Kernel
Features without Manual Labels
Labeling / Binary Classification
Post-Processing
Segmented Images

Figure: Pipeline of framework proposed in (Iftikhar and Godil, 2013).

Feature measures for the segmentation of neuronal membrane using a machine learning algorithm

- **Image Pre-processing**: illumination correction.
- **Patch Subsetting**
  - 3 by 3 neighbourhood patches extracted from the raw images.
- **Feature Selection (for pixels)**
  - A set of 24 distinct features is computed from each neighbourhood patches, followed by the feature normalization step.
- **Pixel Classifier**
- **SegMask Post-processing**

Feature measures for the segmentation of neuronal membrane using a machine learning algorithm

- **Image Pre-processing**
- **Patch Subsetting**
- **Feature Selection**
- **Pixel Classifier**
  - An SVM-RBF based classifier.
- **SegMask Post-processing**
  - Morphological operations
  - Shape-related thresholding

Details of Feature Selection

- CandidatePixel **R** → NeighbourPatch **S** → FeatureVector **T**
- To perform a pixel-level classification for a **candidate pixel R**, its 3 by 3 **neighbourhood patch S** is extracted firstly, then a **feature vector T** of length 24 is computed, including:
  - the intensities (3x3) of patch S.
  - the median value (1) of the intensities of patch S.
  - the range (1) of patch S, which equals to $Max(S) - Min(S)$.
  - the energy (1) of patch S, which equals to $\sum_{i \in S} i^2$.
  - the 2rd, 3nd and 4th spatial moments (3) of patch S, which equals to $\frac{1}{9} \sum_{i \in S} (i - \mu)^r$ for $r = 2, 3$ and 4 respectively, where μ is the mean value of patch S.
  - the gradients (3x3) of patch S, which can be used to increase the visibility of edges and other details.
- **Feature normalization** is essential to make sure that each feature is normalized to a range between [-1, +1].
- MATLAB.

- **Details of Pixel Classifier**
  - An SVM with RBF kernel:
    - SVM: support vector machine.
    - RBF (radial basis function): $K(x, x') = exp(-\gamma||x - x'||_2^2)$, where $\gamma = \frac{1}{2\sigma^2}$.
  - The optimal setting of the related parameters, including gamma (width of the kernel), cost and weight, was obtained by setting different ranges of them from 10-fold cross validation.
  - LIBSVM package.

Image Segmentation - Learning-based CV Pipeline

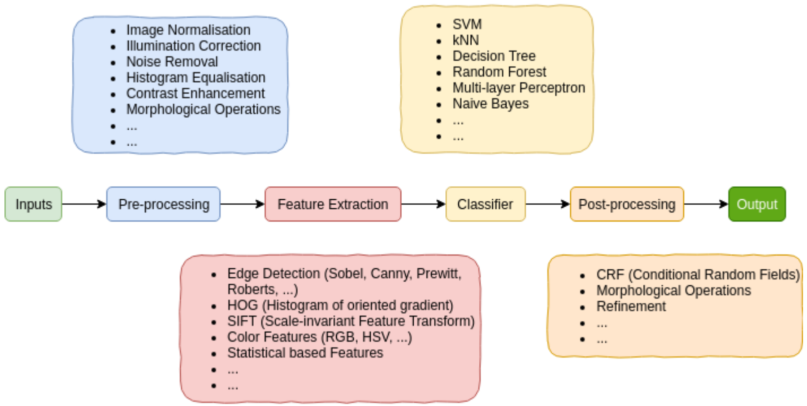
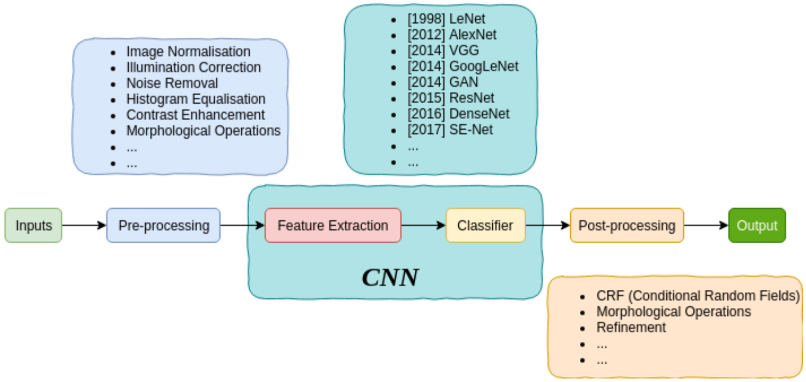


Image Segmentation - (Deep) Learning-based Pipeline



U-Net: Convolutional Networks for Biomedical Image Segmentation

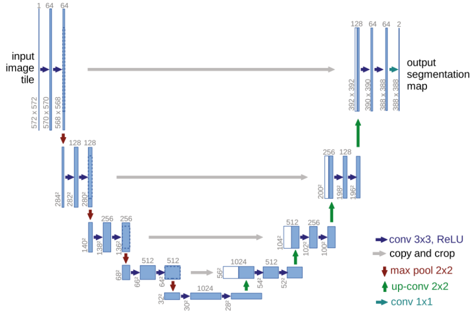


Figure: U-Net Architecture (Ronneberger et al., 2015)

Highlights of U-Net

- **Skip connections**. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization.
- An efficient solution with **limited dataset provided**.
- To output a 2D segmentation mask (rather than a global image label), the network
  does not have any fully connected layers, and only uses the valid part of each convolution, which is quite similar to FCN (**Fully Convolutional Network**) proposed as (Long et al., 2015).

Details of Objective/Loss Function

- **Softmax function** applied pixel-wisely over the final featuremap of the network, to convert the outputs of last activation function to probabilities.
- **Weighted cross entropy function** adopted as the loss function for the back-probagation step.

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

Figure: Energy function for the training of U-Net

- **Weight map** $w(x)$ [next slide] is pre-computed for each ground truth segmentation.

Details of Weight Map $w(x)$

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

- Motivation: to force the network to learn the small separation borders between touching cells.

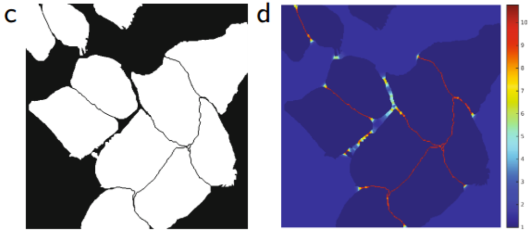


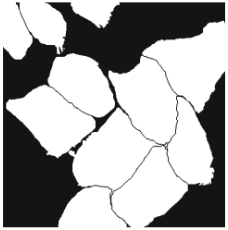
Figure: Left: binary GT mask; Right: weight map $w(x)$ pre-computed.

Details of Weight Map $w(x)$

- How: to assign **the separating background labels between touching cells** with **a large weight** in the loss function.

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

c



d

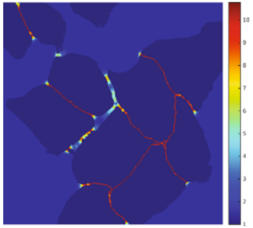


Figure: Border pixels, which are visualized in red, are assigned with large weights.

Details of Hyper-parameters and Experimental Settings

(used in the original implementation of U-Net)

- Input image size: 512×512 (raw)
- Input image size: 572×572 (pre-processed via overlay-tile strategy [next slide])
- Output image size: 388×388 (unpadded convolutions)
- Weight map pre-computation: $w_0 = 10$ and $\sigma = 5$.
- Batch size: 1
- Optimizer: SGD (stochastic gradient descent) with momentum set as 0.99
- Weight initialization: gaussian distribution.
- You are welcome to change them in experiments and see whether a better result can be achieved.

Details of Overlap-tile Strategy

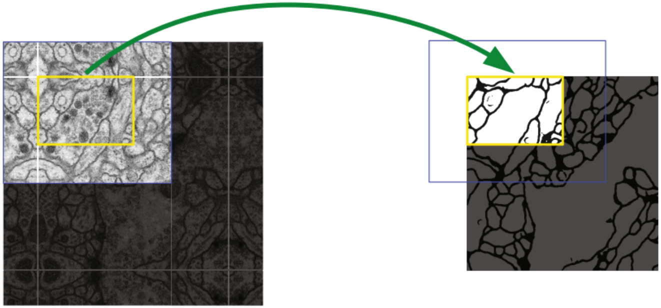
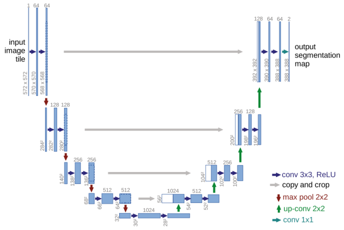


Figure: Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring. That is, mirroring and padding image from 512×512 to 572×572.

Details of U-Net Architecture



- Unpadded Convolutions.
- Copy and CenterCrop (skip connections).
- Pre-processing:
  - For input image, overlap-tile strategy is applied ($512 \times 512 \rightarrow 572 \times 572$).
  - For segmentation GT, CenterCrop is applied ($512 \times 512 \rightarrow 388 \times 388$).

Details of Data Augmentation Data augmentation is essential for the network training, especially when only few training samples are available.

- Random rotation ($[90^\circ, 180^\circ, 270^\circ]$ or other random degrees)
- Random shifting (x pixels horizontally or vertically)
- Random flipping (horizontally and vertically)
- Random elastic deformations (using random displacement vectors sampled from a Gaussian distribution with 10 pixels standard deviation, followed by bicubic interpolation)
- . . .

For image segmentation tasks, both inputs and GTs should be augmented accordingly (joint transform).

Training of CNN

- Record the training loss and testing loss simultaneously. Draw their curves during the training process to check whether/when the overfitting issue appears.
- Fine-tune learning rate. (Suggested by Andrew Ng's Machine Learning Course, an appropriate LR can be found via *dividing it by 3*. Put differently, try 1, 0.3, 0.1, 0.03, 0.01, 0.003, . . .).
- Weighted CE > CE, for problem with imbalanced class distribution.
- Data augmentation (on-the-fly).
- Optimizer (Try Adam at beginning).
- Dropout layers (dropout rate).
- LR warm up, LR scheduler, Weight initialization, . . .

Training of CNN

- **Bag of Tricks for Image Classification with Convolutional Neural Networks** (He et al., 2019)
- https://towardsdatascience.com/a-bunch-of-tips-and-tricks-for-training-deep-neural-networks (NO.21 suggestion is very good).

Evaluation - V^{Rand} and V^{Info}

- Two key metrics:
  - Foreground-restricted Rand Scoring after border thinning: V^{Rand}.
  - Foreground-restricted Information Theoretic Scoring after border thinning: V^{Info}.
- To obtain these two key metrics for your segmentation results, you need:
  - **Fiji**: an advanced image processing software.
  - **Evaluation script**: to help you calculate these values (using Fiji).
- Both of them and related details can be found and downloaded from `http://brainiac2.mit.edu/isbi_challenge/evaluation`.

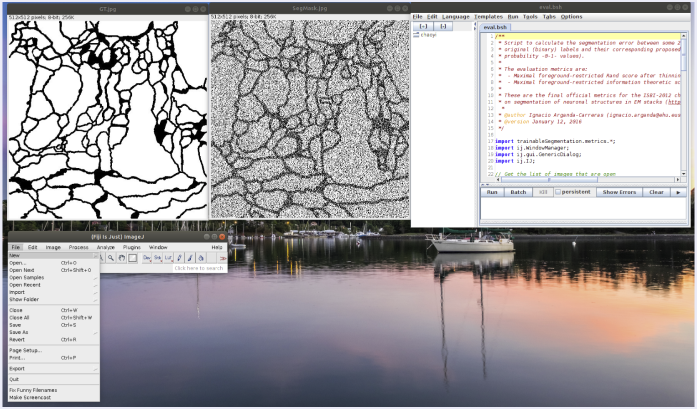
Evaluation - V^{Rand} and V^{Info}

Steps

1. Download the script and save it as ".bsh" file, such as "eval.bsh".
2. Open the paired images for evaluation (i.e., GroundTruth and SegmentationMasks) in Fiji (Fiji: File -> Open. . .).
3. Open the "eval.bsh" in Fiji (Fiji: File -> Open. . .), and click on "Run" button.
4. Choose "Original labels" and "Proposal" correctly. Tick "Binary proposal" section if a binary segmentation mask is to be evaluated, then click on "OK" button.
5. Done. Your V^{Rand} and V^{Info} will be calculated automatically.

Evaluation - V^{Rand} and V^{Info}

Step 1 to 3



```
/**
 * Script to calculate the segmentation error between some 2
 * original (binary) labels and their corresponding proposed
 * probability -0-1- values).
 *
 * The evaluation metrics are:
 *  - Maximal foreground-restricted Rand score after thinnin
 *  - Maximal foreground-restricted information theoretic sc
 *
 * These are the final official metrics for the ISBI-2012 ch
 * on segmentation of neuronal structures in EM stacks (http
  *
 * @author Ignacio Arganda-Carreras (ignacio.arganda@ehu.eus
 * @version January 12, 2016
 */

import trainableSegmentation.metrics.*;
import ij.WindowManager;
import ij.gui.GenericDialog;
import ij.IJ;

// Get the list of images that are open
```

Evaluation - V^{Rand} and V^{Info}

Step 4

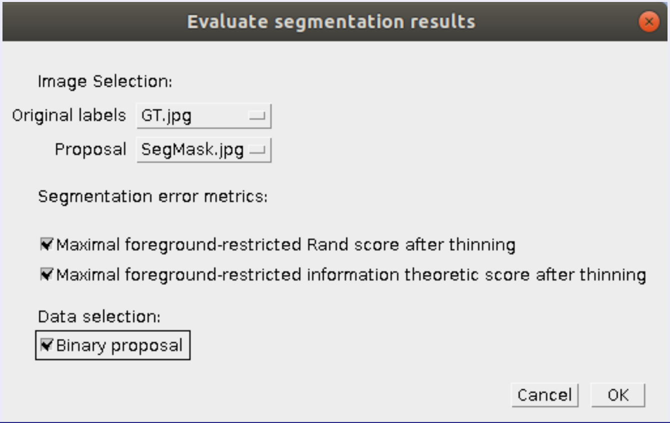

Evaluate segmentation results

Image Selection:

Original labels: GT.jpg

Proposal: SegMask.jpg

Segmentation error metrics:

☑ Maximal foreground-restricted Rand score after thinning

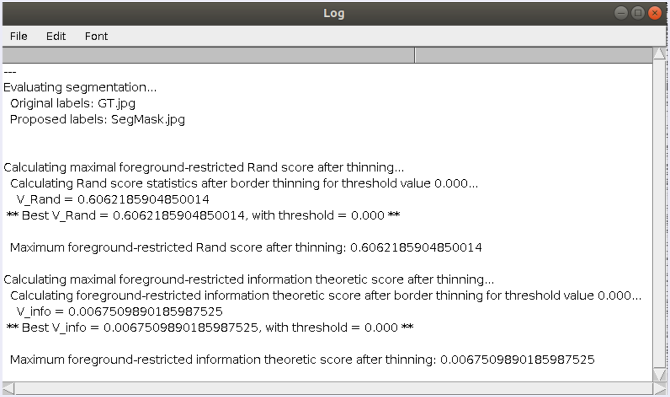
☑ Maximal foreground-restricted information theoretic score after thinning

Data selection:

☑ Binary proposal

Evaluation - V^{Rand} and V^{Info}

Step 5



```
---
Evaluating segmentation...
 Original labels: GT.jpg
 Proposed labels: SegMask.jpg


Calculating maximal foreground-restricted Rand score after thinning...
 Calculating Rand score statistics after border thinning for threshold value 0.000...
  V_Rand = 0.6062185904850014
** Best V_Rand = 0.6062185904850014, with threshold = 0.000 **

 Maximum foreground-restricted Rand score after thinning: 0.6062185904850014

Calculating maximal foreground-restricted information theoretic score after thinning...
 Calculating foreground-restricted information theoretic score after border thinning for threshold value 0.000...
  V_info = 0.0067509890185987525
** Best V_info = 0.0067509890185987525, with threshold = 0.000 **

 Maximum foreground-restricted information theoretic score after thinning: 0.0067509890185987525
```

Evaluation - V^{Rand} and V^{Info}

- Two key metrics:
  - Foreground-restricted Rand Scoring after border thinning: V^{Rand}.
  - Foreground-restricted Information Theoretic Scoring after border thinning: V^{Info}.
- **Please present V^{Rand} and V^{Info} in your report.**
- **Furthermore, you are very welcome to use other quantitative metrics [such as, IOU (intersection over union)] for the evaluation of your segmentation masks generated.**

Evaluation - Cross Validation

- Cross-validation is required for the evaluation of your learning-based approaches.

Steps towards K-fold Cross Validation

1. Split dataset (30 ssTEM with GTs) into K folds and assign $ID_{fold} \in [1, 2, \ldots, k]$ to each fold.
2. Repeat experiments for K times: for the mth time of the experiments, train and evaluate your methods proposed by choosing the mth fold ($ID_{fold} == m$) as the testing dataset and the rest $K - 1$ folds as the training dataset.
3. Calculate the average of the evaluation metrics for each experiment performed, such as V^{Rand} and V^{Info}.

Evaluation - Cross Validation

- Cross-validation is required for the evaluation of your learning-based approaches.
- In other words, each single image from the dataset given should be treated as testing data (and once only) during the K-fold cross validation.

What we are looking for …

- Your abilities in designing/implementing several appropriate method flows/pipelines for the image segmentation task (www.draw.io),
- Showing your understandings on the underlying algorithms (discussion section),
- And your efforts put in improving the accuracy (ablation studies),
- Rather than a very high accuracy itself (considering the potentially long training time required).

References

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., and Li, M. (2019). Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 558–567.

Iftikhar, S. and Godil, A. (2013). Feature measures for the segmentation of neuronal membrane using a machine learning algorithm. In *Sixth International Conference on Machine Vision (ICMV 2013)*, volume 9067, page 90670V. International Society for Optics and Photonics.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.