**MU-Akb-CZ**

Authors: Cem Emre Akbas, Michal Kozubek

Email: ceakbas@gmail.com

Platform: Linux

Prerequisites: Python 3.6

*MU-Akb-CZ: SUMMARY*

Our cell segmentation algorithm is based on a U-Net-like deep convolutional neural network. After pre-processing and data augmentation, the network predicts the segmentation masks. The predicted images are post-processed to obtain final segmentation masks. Like the U-Net [1], this network consists of a contracting path for capturing context and a symmetric expanding path for getting precise locations of detected image features. We employ 4x4 max-pooling to ensure high-amplitude activations in the network. This network needs only a few completely annotated images for proper training thanks to extensive data augmentation.

*MU-Akb-CZ: PREPROCESSING*

In the pre-processing pipeline, we aim to normalize the input images in order to provide the network with images having very similar intensity levels and minimal noise so that the CNN can specialize better on its segmentation task. To this end, we use intensity normalization. We compute the average intensity value of the training sequences, $m_i$, first. Then, the intensity values of all training, validation, and test images are normalized to the average intensity of the training set. After the normalization of intensity values, the top hat filter of size *s* is applied in order to filter the noise.

*MU-Akb-CZ: SEGMENTATION*

After the pre-processing step, data augmentation is applied to the input data. In this part, we use rotation (random in range [0, 359] degrees), shifting (random in 20% range in x and y axes), elastic transformations, and combinations of these three methods. The purpose is to increase the variety in the training dataset so that the trained network can perform better on unseen examples. Finally, intensity values are scaled down to [-0.5, 0.5] by first dividing all intensity values by 255 and then subtracting 0.5 from all pixels. We observe that pixel values within this range provide faster activation in the convolutional layers since the mean intensity values are kept.

Our network is strongly inspired by the U-Net. It consists of a contracting path and an expansion path. Contracting path consists of convolution, ReLU and max-pooling layers, whereas expansion path consists of up-convolution, ReLU and convolutional layers. We use the same design philosophy, but introduce several changes in the U-Net structure. First, we use triple convolutional and up-convolutional layers. Second, we use 5x5 convolution kernels to extract features from bigger windows. Third, we use 4x4 max-pooling (with stride 4x4 for down-sampling) instead of 2x2. The first two changes are necessary to utilize larger (4x4) max-pooling. 4x4 max-pooling ensures high-amplitude activations that propagate better to later layers than the ones generated by 2x2 max-pooling. We use 15 convolutional layers instead of 18 (used in the original U-Net). The expansive path consists of up-convolutional, convolutional layers, ReLU activations and concatenation operations. Up-convolutional layers use 5x5 convolution kernels with 4x4 strides for up-sampling. The resulting up-sampled feature maps are of the same size as the corresponding ones in the contracting path, thanks to the padded convolutions. These corresponding feature maps are simply concatenated without cropping. This operation is followed by three convolutional layer&ReLU pairs for localization of extracted features. Symmetrical to the contracting path, only two up-sampling steps are used. To produce the final output, a 1x1 convolutional layer is used with a sigmoid activation function. This output layer maps multi-channel feature maps to a single-channel binary segmentation mask, which is the network output. The network output is of exactly the same size as the network input. During the implementation, there is a size constraint due to down-/up-sampling. We down-sample the images by 16 in both dimensions. In order to produce feature maps of consistent sizes in the expansion path, the dimensions of input images should be integer multiples of 16. If this is not the case, the images are zero-padded to the nearest integer multiple of 16. The zero-padded regions are cropped right after the prediction. Our network can finish the training in 1000 training iterations, using around 400 images (without down-scaling) with the use of data augmentation that needs only a few annotated images for proper training. The proposed network is trained with the following combined loss function.

The loss function is a crucial part of the learning process in cell segmentation networks due to the under-representation of boundary pixels of cells. The ideal loss function should be able to handle both imbalanced inputs (i.e., over-representation of the background pixels) and imbalanced outputs (penalization of false positive (FP) and false negative (FN) pixels in the output) while correctly segmenting individual cell boundaries. The weighted cross-entropy loss function used in the U-Net is able to penalize FPs and FNs, but it is a sub-optimal solution to the imbalanced inputs problem. The Dice loss function [2] successfully balances the input classes; however, it cannot control the penalization of either

FPs or FNs. Therefore, a combined loss function is recently proposed to handle both input and output imbalances [3]. However, Taghanaki's loss function is designed for medical images and underperforms at cell separation regions; it struggles to segment neighboring cells separately. Our loss function aims to adapt Taghanaki's loss function to the bioimage segmentation by employing the weighted cross-entropy loss function in [1] to enhance the learning at the cell boundary regions:

$$L = \alpha\left(-\frac{1}{N}\sum_{n=1}^{N} w(n)(t_n \cdot \ln(p_n)) + ((1-t_n)\cdot \ln(1-p_n))\right) + (1-\alpha)\left(1 - \frac{2\sum_{n=1}^{N} t_n \cdot p_n + \varepsilon}{\sum_{n=1}^{N} t_n + \sum_{n=1}^{N} p_n + \varepsilon}\right)$$

where $\alpha \in [0, 1]$ (default value is 0.5) is the relative cross-entropy weight, $t_n$ is the ground truth label for each pixel, $p_n$ is the predicted probability map for the foreground label over N pixels, $w(n)$ is pixel weights defined in [1] and $\varepsilon \sim 0$ and used to avoid division by 0.

Our loss function looks similar to the combined loss function in [3], yet it has significant changes. In the cross-entropy part, it uses pixel weights, instead of constant weights. Pixel weights enhance the representation of the pixels between neighboring cells in the loss function. Thus, the network can learn better at cell boundaries and separate neighboring cells more successfully. The second component of our combined loss function is the Dice loss function [2]. It is used to handle imbalanced inputs, i.e., when there is an imbalance between the presence of background and foreground pixels. In our loss function, the Dice loss part is modified so that it computes the loss value without a shift because the training process is sensitive to the shifts in the input values. It is experimentally shown that the proposed combined loss function not only performs well in the case of imbalanced inputs and outputs but also improves the separation of touching cells. Therefore, the proposed loss function is a tailor-made loss function for the cell segmentation task in biomedical images.

*MU-Akb-CZ: POST-PROCESSING*

We apply four post-processing steps to the predicted images in order to further reduce both false positives (FP) and false negatives (FN) at the pixel level in the predicted images. Step 3 and Step 4 use temporal information.

1. Small particle removing: In the predicted images, there may appear FPs that form small objects. Therefore, we remove markers that are smaller than a specified size threshold $o_{size}$.

2. Hole filling: In some frames, predicted markers may have holes (FNs) usually due to bright cell parts. We fill those holes.

3. Merging: If the area covered by the separately segmented markers contains only one marker in the previous and the next frame, we merge wrongly separated markers by replacing them with the union of the corresponding markers in these three frames (correcting FNs in between them).

4. Division of mistakenly merged markers: In some frames, the CNN can segment two neighboring cells as a single marker due to FPs. This step deals with those merged markers. Here, we use temporal information and hence call this method as the temporal inconsistency check algorithm. We use 5-neighborhood.

---

*Temporal Inconsistency Check Algorithm:*

---

*For each binary marker in frames 1, …, N (N: Total number of frames)*

*i) For marker $m_t$, count the number of cells in the area covered by $m_t$ in frames t-2, t-1, t+1 and t+2; and store as $count_{t-2}$, $count_{t-1}$, $count_{t+1}$, $count_{t+2}$ accordingly.*

*ii) Only in the area covered by $m_t$, sum the values in frames t-2, t-1, t+1 and t+2, and store the sum as $sum_{m_t}$.*

*iii) If ($count_{t-2}$ > 1 or $count_{t-1}$ > 1) and ($count_{t+1}$ > 1 or $count_{t+2}$ > 1), start eroding $sum_{m_t}$. Else, continue with the next marker (step i).*

*a. Erode $sum_{m_t}$ until more than one cell appears in the area covered by $m_t$*

*b. Dilate the eroded markers iteratively until they merge. Then, take the divided markers from the preceding iteration.*

c. *Remove the initial marker from the predicted image and add the divided markers to the predicted image.*

---

As the final touch, the small particle removing step is applied again in order to remove unwanted small markers created by the temporal inconsistency check algorithm. The output of this process is used as final segmentation masks.

**REFERENCES**


1. Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of Medical Image Computing and Computer-Assisted Intervention*, 234-241 (2015).
2. Milletari F, Navab N, Ahmadi S. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *Proceedings of the 4th International Conference on 3D Vision*, 565–571 (2016).

3. Taghanaki SA, Zheng Y, Zhou SK, Georgescu B, Sharma P, Xu D, Comaniciu D, Hamarneh, G. Combo loss: Handling input and output imbalance in multi-organ segmentation. *Computerized Medical Imaging and Graphics* **75,** 24–33 (2019).