Weakly Supervised Microscopy Cell Segmentation via Convolutional LSTM Networks

Assaf Arbelle and Tammy Riklin Raviv

I. METHODS

We address individual cells’ segmentation from microscopy sequences. The main challenge in this type of problems is not only foreground-background classification but also the separation of adjacent cells. We apply two orthogonal approaches to overcome the multiple instance problem. From the segmentation perspective, we adopt the three class loss used by [1], [2]. The segmentation representation is designed to enhance individual cells’ delineation by a partitioning of image domain into three classes: foreground, background and cell contours. From the detection perspective, we get our inspiration from [3], and aim to detect rough cell markers. The markers, as opposed to the full segmentation, do not cover the entire cell, but are rather a small "blob" somewhere within the cell. The markers have two desirable properties. First, they are much smaller than the object and thus are easier to separate between instances. One marker will never overlap or touch boundaries with a neighboring marker. Second, the markers are easy to annotate, as the annotator does not need to be precise, making data acquisition a simpler task. Often, for microscopy image sequences, the only available annotation is in the form of markers or approximate cell centers. We train the proposed network to estimate both the segmentation and the markers and merge the two using the Fast Marching Distance (FMD) [4]. The entire framework is illustrated in Figure 1.

A. Input and Output

The input to the method is a sequence of live cell microscopy images of arbitrary length T. We define the d dimensional (2 or 3) image domain by \( \Omega \in \mathbb{R}^d \). We denote a frame in the input image sequence as \( I_t : \Omega \rightarrow \mathbb{R} \), where the subscript \( t \in [0, T - 1] \) denotes the frame index and \( I_t(v) \) is the intensity of a pixel (or voxel), \( v \in \Omega \). The output of the network consists of two components, the scalar marker map (Section I-D) \( M_{I_t} : \Omega \rightarrow [0, 1] \) which represents the probability of a pixel (voxel) to belong to a marker (cell segmentation core) and the soft segmentation map (Section I-C) denoted as \( S_t : \Omega \rightarrow [0, 1]^3 \) which represents the probabilities of each pixel (voxel) to belong to either the foreground, background or cell boundary. These two maps are then passed to an instance segmentation block (Section I-F), which outputs the final labeled segmentation map, \( \Gamma_{I_t} : \Omega \rightarrow \mathbb{N}^+ \). Figure 1 shows an overview of the proposed method with visualization of the intermediate steps.

A Arbelle and T Riklin Raviv are with The Department of Electrical and Computer Engineering, and The Zlotowski Center for Neuroscience, Ben-Gurion University of the Negev.

B. LSTM-UNet

The proposed network incorporates C-LSTM [5] blocks into the U-Net [6] architecture. This combination, first suggested in our preliminary work [1], is shown to be powerful. The U-Net architecture, built as an encoder-decoder with skip connections, enables to extract meaningful descriptors at multiple image scales. However, this alone does not account for the cell specific dynamics that can significantly support the segmentation. The introduction of C-LSTM blocks into the network allows considering past cell appearances at multiple scales by holding their compact representations in the C-LSTM memory units. We propose here the incorporation of C-LSTM layers in every scale of the encoder section of the U-Net. Applying the C-LSTM on multiple scales is essential for cell microscopy sequences since the frame to frame differences might be at different scales, depending on cells’ dynamics. The specific architecture was selected based on preliminary work which shows the empirical advantage over other alternatives [1]. The network is fully convolutional and, therefore, can be used with any image size\(^1\) during both training and testing. Figure 1 illustrates the network architecture detailed in Section II.

The network is composed of two sections of \( N \) blocks each, the encoder recurrent block \( E_{\theta_n}^{(n)}(\cdot) \) and the decoder block \( D_{\theta_n}^{(n)}(\cdot) \) where \( \theta_n \) are the network’s parameters. The input to the C-LSTM encoder layer \( n \in [0, \ldots, N - 1] \) at time \( t \in T \) includes the down-sampled output of the previous layer; the output of the current layer at the previous time-step; and the C-LSTM memory cell. We denote these three inputs as \( x_t^{(n)}, h_{t-1}^{(n)}, c_{t-1}^{(n)} \), respectively. Formally we define:

\[
(h_t^{(n)}, c_t^{(n)}) = E_{\theta_n}^{(n)}(x_t^{(n)}, h_{t-1}^{(n)}, c_{t-1}^{(n)})
\]

where,

\[
x_t^{(n)} = \begin{cases} I_t, & n = 0 \\ MaxPool(h_t^{(n-1)}), & 0 < n < N 
\end{cases}
\]

The inputs to the decoder layers \( n \in [N, \ldots, 2N - 1] \) are the up-sampled\(^2\) output of the previous layer and the output of the corresponding layer from the encoder denoted by \( y_t^{(n)} \) and \( h_t^{(2N-1-n)} \), respectively. We denote the decoder output as \( z_t^{(n)} \). Formally,

\[
y_t^{(n)} = \begin{cases} h_t^{(n-1)}, & n = N \\ UpSample(z_t^{(n-1)}), & N < n < 2N - 1 
\end{cases}
\]

\(^1\)In order to avoid artifacts it is preferable to use image sizes which are multiples of eight due to the three max-pooling layers.

\(^2\)We use bi-linear interpolation
where: 
\[ i_t^{(n)} = D_{\theta_n}(y_t^{(n)}, h_t^{2N-1-n}) \]  
(4)

In [?] the network had a single output for the segmentation map while here, the last layer of the network is split to produce outputs for the soft segmentation map and marker map, \( S_t \) and \( M_t \), to be discussed in the following sections. We define the full network as the composition of \( N \) encoder blocks followed by \( N \) decoder blocks. Note, that the encoder blocks, \( E_{\theta_n} \), encode high-level spatio-temporal features at multiple scales and the decoder blocks, \( D_{\theta_n}^{(n)} \), refines that information into full scale segmentation and marker maps.

**C. The Soft Segmentation map**

The output of the soft segmentation layer is defined as follows:

\[ S_t = D_{\theta_S}(y_t^{2N-1}, h_t^{(0)}) \]  
(5)

where \( \theta_S \) are the parameters of this layer.

\( S_t \) has the same spatial dimension as the input and has three channels corresponding to the un-normalized evidence for the three classes, background, foreground and cell boundary, encoded in the entries \( l \in \{0, 1, 2\} \). We define the soft segmentation, \( S_t \) by the following softmax equation:

\[ S_t(v) \triangleq p(l|S_t(v)) = \frac{\exp([S_t(v)]_l)}{\sum_{l' \in \{0,1,2\}} \exp([S_t(v)]_{l'})} \]  
(6)

**D. The Marker map**

Similarly to the soft segmentation layer, the marker layer is defined as:

\[ M_t = D_{\theta_M}(y_t^{2N-1}, h_t^{(0)}) \]  
(7)

where \( \theta_M \) are the weights of this layer. Note that the weight \( \theta_M \) are different from the soft segmentation weights \( \theta_S \).

\( M_t \) has the same spatial dimension as the input and has one channel corresponding to the un-normalized evidence for the marker. We define the marker map, \( M_t \) by the following sigmoid function:

\[ M_t(v) \triangleq \sigma(M_t) = \frac{1}{1 + \exp^{-M_t}}, \]  
(8)

An ablation study, designed to show the affect of the marker map estimation is presented in Section ??.

**E. Addressing 3D Data**

In this paper, we address both 2D and 3D image sequences. Due to the heavy computational load of the 3D data, we generate the volumetric \( M_t \) and \( S_t \) from 2D slices. Let \( \{i_t^{(k)}\}_{k=1}^K \) be the input \( I_t \) decomposed into \( K \) slices along the depth dimension. Similarly, we denote the corresponding soft segmentation and marker maps \( S_t = \{S_t^{(k)}\}_{k=1}^K \) and \( M_t = \{M_t^{(k)}\}_{k=1}^K \), respectively. In order to keep the spatial coherence we process \( i_t^{(k)} \) by feeding the network with three consecutive slices, \( \{i_t^{(k-1)}, i_t^{(k)}, i_t^{(k+1)}\} \). These three slices are concatenated along the channel axis. The outputs \( S_t \) and \( M_t \) are constructed by stacking the respective 2D slices. We perform an ablation study to assess this approach for handling 3D data, see Section ??.

**F. Instance Segmentation**

The instance segmentation block merges the two network outputs, \( S_t \) and \( M_t \), into a single, multi-label map. We define the semantic cell region by:

\[ R_t^{\text{cell}} = \{v|\arg\max_{l \in \{0,1,2\}}|S_t(v)|_l = 1\} \]  
(9)

The boundary class in \( S_t \) is used to partition the cell region, \( R_t^{\text{cell}} \), into connected components, where each component should represent a single cell instance. Yet, due to possible under-segmentation, the number of connected components may not necessarily correspond to the number of blobs in the marker map, which more faithfully represents the true number of cells. We, therefore, use \( M_t \) to facilitate the separation of merged cell instances. Let the marker regions be the thresholded marker map such that:

\[ R_t^{\text{blob}} = \{v|M_t(v) \geq 0.5\} \]  
(10)

\( R_t^{\text{blob}} \) is partitioned into \( N_t \) connected components, \( \{m_i\}_{i=1}^{N_t} \). We construct the final label map \( \Gamma_t \) for each \( v \in R_t^{\text{cell}} \), by searching for the nearest \( m_i \) with respect to a geodesic distance defined by the soft segmentation foreground map \( [S_t]_l \):

\[ \Gamma_t(v) = \begin{cases} \arg\min_{i=1...N_t} d_f(m_i, [S_t]_l) & v \in R_t^{\text{cell}} \\ 0 & \text{otherwise} \end{cases} \]  
(11)

The function \( d_f \) denotes the FMD, which computes the shortest distance between a source point, \( v \), and a target point \( m_i \), given a speed map \( [S_t]_l \) in either 2D or 3D, with respect to the input domain. Areas in a speed map with high values will result in shorter paths, whereas areas with lower values will constrain the solution to longer distances. Specifically, pixels (voxels) with high boundary probability imply low foreground probability and thus resulting in very low speed.

**G. Data and Annotations**

Manual segmentation of microscopy sequences is a tedious and time consuming task, in particular where 3D+t data is considered. The annotation task can be simplified if the annotator, instead, marks the approximate cell centers. While ideally we wish to have one-hot encoded (background, foreground and boundary) labels, \( S_t : \Omega \rightarrow \{0,1\}^3 \), for the entire training sequence, in practice, for most of the sequence cells are only weakly annotated by markers, \( M_t : \Omega \rightarrow \{0,1\} \). Specifically, we assume that at least one frame (or 2D slice in 3D) is fully annotated.

**H. Training and Loss**

The network is trained using Truncated Back Propagation Through Time (TBPTT) [7]. At each back propagation step the network is unrolled to \( \tau \) time-steps. The total loss for training is a sum of two losses, one for the marker map, \( L_M \), and one for the soft segmentation map, \( L_S \). For the marker map we use single class cross entropy loss:
**Figure 1:** Method outline: A. The input to the method is a time-lapse sequence of microscopy images. B. The network’s down-sampling path consists of a C-LSTM layer followed by a convolutional layer with ReLU activation, the output is then down-sampled using max pooling and passed to the next layer. The up-sampling path consists of a concatenation of the input from the lower layer with the parallel layer from the down-sampling path followed by two convolutional layers with ReLU activations. The last layer is split to produce the soft segmentation and marker maps, $S_t$ and $M_t$, respectively. C. The two network outputs are merged in to perform instance segmentation utilizing the FMD. The figures in the ”Instance Segmentation” block show a zoom-in view of the gray dashed line. The distances to the two relevant centers allows for optimal separation. D. The final label map, $\Gamma_t$, is the extracted by finding the minimal distance to the centers.

$$L_M = - \sum_{t'=t}^{t+\tau} \sum_{v \in \Omega} [m_t(v) \cdot \log(M_t(v))]$$

$$+(1 - M_t) \cdot \log(1 - M_t(v))]$$

The soft segmentation map is penalized by the weighted, multi-class cross-entropy loss, giving higher weights, $w_t$ to the cell contour pixels, due to the class imbalance:

$$L_S = - \sum_{t'=t}^{t+\tau} \sum_{v \in \Omega} \sum_{l \in \{0, 1, 2\}} w_l \cdot [S_t(v)]_l \cdot \log([S_t(v)]_l)$$

Frames, or pixels within frames, which did not have GT annotations were not taken into account in the loss calculation. The final loss is set to be the sum of the two:

$$L = L_M + L_S$$

**II. IMPLEMENTATION DETAILS**

**A. Libraries**

The method is implemented in python utilizing Tensorflow for the neural networks section and the scikit-fmm python library for the FMD.

**B. Architecture**

The network include $N = 4$ encoder and decoder blocks. Each block in the encoder section is composed of C-LSTM layer, leaky ReLU, convolutional layer, batch normalization [8], leaky ReLU and finally down-sampled using maxpool operation. The decoder blocks consist of a bi-linear interpolation, a concatenation with the parallel encoder block followed by two sets of convolutional layer, batch normalization [8], and leaky ReLU. The same network is used both for 2D and 3D datasets. All C-LSTM kernels are of size $5 \times 5$ and all convolutional layers use kernel size $3 \times 3$. The feature depths in the encoder and decoder paths is set to be (128, 256, 512, 1024) and (1024, 512, 256, 128), respectively. All maxpool layers use kernel size $2 \times 2$ without overlap. The last convolutional layer uses kernel size $1 \times 1$ with depth 3 for the soft segmentation layer $S_t$ and depth 1 for the marker layer, $M_t$, followed by a softmax or a sigmoid layer, respectively, to produce the final probabilities (see Figure 1).

**C. Training Regime**

We trained the networks on each dataset separately for $200K$ iterations ($300K$ for the 3D datasets) using the ADAM optimizer [9] with learning rate of 0.0001. The unroll length parameter was set to $\tau = 4$ and the batch size was set to five sequences. The weights $w$ were set to be 0.15, 0.25 and 0.6 for background, foreground and cell boundary, respectively (Section [1-H]).

**REFERENCES**


