KIT-Sch-GE (2)

Authors: Katharina Löffler, Tim Scherr, Ralf Mikut

Email: ralf.mikut@kit.edu

Platform: Linux (tested on Ubuntu 16.04 and 18.04)

Prerequisites: ≥ 16 GiB RAM, ≥ 12 GiB VRAM (CUDA = 11.0), Gurobi installed (for tracking)

KIT-Sch-GE (2): SUMMARY

Our method follows the tracking by detection paradigm. For the segmentation step, a deep learning-based prediction of cell distance and neighbor distance maps is used followed by a watershed post-processing. The tracking step is based on a displacement estimation in combination with a matching formulated as maximum flow minimum cost problem. A description of the segmentation method provides [1] and some minor improvements are summarized in [2]. The tracking method is described in [3]. The code repository is available at https://git.scc.kit.edu/KIT-Sch-GE.

KIT-Sch-GE (2): PREPROCESSING

This step involves a minimum-maximum normalization of the single frames into the [−1, 1] range (whole volume for 3D data). The contrast limited adaptive histogram equalization (CLAHE) can be applied with the parameter *apply_clahe*. The inputs for the segmentation CNN are zero-padded if necessary. Downsampling can be applied with a scaling parameter $s \leq 1$.

KIT-Sch-GE (2): SEGMENTATION

Architecture. Since most of the provided reference segmentation annotations are 2D, our convolutional neural network architecture is based on the 2D U-Net architecture [4]. However, instead of using a single decoder path, two parallel decoder paths are used. This allows each decoder path to focus on features related to the desired output. The maximum pooling layers are replaced with 2D convolutional layers with stride 2 and kernel size 3 and batch normalization layers are added. The number of feature maps is doubled from 64 feature maps to a maximum of 1024 and halved in each decoder path correspondingly. The ReLU or the Mish activation function [5] is used within the network and a linear activation for the output layer.

Cell Distances and Neighbor Distances. The modified 2D U-Net is trained to predict cell distance and neighbor distance maps. In the cell distance map, each pixel belonging to a cell represents the normalized distance to the nearest pixel not belonging to this cell, whereas in the neighbor distance map each pixel belonging to a cell represents the inverse normalized distance to the nearest pixel of the closest neighboring cell. To be able to handle elongated cells and to simplify the post-processing, the normalization and scaling step used in [1] has been adapted, resulting in fewer manually tunable parameters. With the normalization of [1], large areas with high neighbor distances could occur for elongated cells making the post-processing difficult (e.g., for **Fluo-C2DL-MSC**). In addition, the grayscale closing has been replaced with a bottom-hat-transform-based closing which fits better into the neighbor distance maps. In [2], the influence of these minor modifications is evaluated.

Inference. The preprocessed frames are fed into the CNN (slice-wise for 3D data). Our executable enables multi-GPU support and the batch size can be adjusted to the available VRAM.

Post-Processing. First, the predicted cell distance and neighbor distance maps are smoothed slightly using a Gaussian kernel with fixed size. The smoothed cell distance map is thresholded with the threshold T_{mask} to extract the region to flood with a watershed algorithm. Markers are extracted by calculating the difference between the smoothed cell distance map and the smoothed and scaled neighbor distance map. The obtained result is thresholded with the threshold T_{marker}. Markers with a size smaller than 10% of the mean marker size in a frame are removed. If no marker is found in a frame, the mask threshold is lowered till at least one object is found or $T_{\text{marker}} < T_{\text{mask}}$. With the parameter *apply_merging*, touching cells with only little neighbor distance information in the touching region can be merged (for cells split due to their shape, such as dumbbell shaped cells). This can avoid oversegmentation (e.g., for **BF-C2DL-MuSC**). Markers can be fused in axial direction (since 3D data are only processed slice-wise) with a kernel of shape (3, 1, 1) using the *fuse_z_seeds* parameter. For 3D data, local maxima are used as markers if more than *N_splitting* cells are detected in a frame with the method described above. Furthermore, in that case merged cell nuclei are detected if their volume is bigger than $\frac{7}{5}$ times the mean cell volume at that single frame. Detected merged cells are tried to split using iteratively a higher marker threshold $T_{\text{marker}} \in \{0.5, 0.6, 0.75\}$. A movement-based region of interest extraction (artifact correction) can be applied with the parameter *artifact_correction* which only works well for dense cell packaging, (e.g., for **BF-C2DL-HSC**). Finally, for datasets with a field of interest (FOI) defined (e.g., **Fluo-N2DL-HeLa**), a FOI correction is applied.

The post-processing starts with a slight 2D/3D Gaussian smoothing of the cell and neighbor distance predictions with a fixed smoothing kernel. Then, the region to flood with a 2D/3D watershed is extracted out of the smoothed cell distance prediction using the threshold T_{mask}. To get markers, the smoothed neighbor distance prediction is squared (and scaled slightly applying the tangent function) and subtracted from the smoothed cell distance prediction. The obtained result is thresholded with the threshold T_{marker}. Markers with an area smaller than 10% of the mean marker area in a frame are filtered out. If no marker is found in a frame, the mask threshold is lowered till at least one object is found or $T_{\text{marker}} < T_{\text{mask}}$. Markers can be closed in axial direction (since 3D data are only processed slice-wise) with a kernel of shape (3, 1, 1) using the *fuse_z_seeds* parameter. A movement-based region of interest extraction (artifact correction) can be applied with the parameter *artifact_correction* which only works well for dense cell packaging (e.g., for **BF-C2DL-HSC**). For 3D data, local maxima are used as markers if more than *N_splitting* cells are detected with the method described above. Furthermore, in that case merged cell nuclei are detected if their volume is bigger than $\frac{7}{5}$ times the mean cell volume at that single frame. Detected merged cells are tried to split using iteratively a higher marker threshold $T_{\text{marker}} \in \{0.5, 0.6, 0.75\}$. In the end, the for some data sets needed field of interest (FOI) correction is applied (which may induce errors for cells directly at the FOI borders).

Training Datasets. For the training, crops of the size 320 × 320 pixels are generated automatically and reproducibly from the available segmentation GTs and STs. Based on the cell size statistic of the GTs and STs, the scale parameter s and a search radius (to reduce computation time) are set automatically. For fully annotated 3D data, slices of interest are selected using the image mean and the slice mean. In addition, slices from 3D annotations are processed with 2D morphological opening and closing operations with cell size specific kernels to remove 3D artifacts within a slice of interest and to close emerging gaps. A training/validation split of 80%/20% is used. The tracking GTs are used to check roughly if almost all cells in a segmentation GT crop are annotated. For datasets with fewer than 30 generated, fully annotated crops, also crops which have at least 80% of the cells annotated are included. Only STs from frames without any annotated GT cell are used. The fraction of ST crops relative to the number of GT crops is set to 33% for the training set and to 25% for the validation set. However, if less than 75/15 crops (training/validation) per cell type are available, more ST crops are included until 75/15 crops are available in total (or all possible ST crops have been included).

Training Process. To learn the cell and the neighbor distances, PyTorch's *SmoothL1Loss* is used and both losses are added. During training flipping, scaling, rotation, contrast changing, blurring and noise augmentations are applied. Models are trained for a maximum number of epochs which depends on the training dataset size:

$$N_{\text{epochs}}^{\max} = \begin{cases} 200 & \text{if } N_{\text{crops}}^{\text{train}} + N_{\text{crops}}^{\text{val}} \geq 1000, \\ 240 & \text{if } 1000 > N_{\text{crops}}^{\text{train}} + N_{\text{crops}}^{\text{val}} \geq 500, \\ 320 & \text{if } 500 > N_{\text{crops}}^{\text{train}} + N_{\text{crops}}^{\text{val}} \geq 200, \\ 400 & \text{if } 200 > N_{\text{crops}}^{\text{train}} + N_{\text{crops}}^{\text{val}} \geq 100, \\ 480 & \text{if } 100 > N_{\text{crops}}^{\text{train}} + N_{\text{crops}}^{\text{val}} \geq 50, \\ 560 & \text{if } N_{\text{crops}}^{\text{train}} + N_{\text{crops}}^{\text{val}} < 50. \end{cases}$$

A batch size of 8 is used (effective batch size of 4 since two GPUs are used). For each training dataset, models are trained with the Ranger optimizer (combines RAdam [6], LookAhead [7], and gradient centralization [8]) and Mish activation, and models with the Adam optimizer [9] and ReLU activation. In both cases, a learning rate scheduler and an early stopping criterion are used (see also Table 1 in [2]). Furthermore, Ranger models are trained a second time for 10% $\cdot N_{\text{epochs}}^{\max}$ epochs with cosine annealing. Models can also be pre-trained or retrained (i.e., our **Fluo-C2DL-Huh7** model is a model trained on GTs and STs of 13 other datasets and has been retrained on **Fluo-C2DL-Huh7** GTs.

Model Selection. All trained models are evaluated on the provided training datasets and the best model in terms of OP_{CSB} is selected. Thereby also multiple thresholds T_{mask}, and T_{marker} can be evaluated.

KIT-Sch-GE (2): TRACKING

The tracking task is split in three steps: (1) coarse tracking of segmented objects to find potential matching candidates, (2) creating a tracking graph by solving a coupled minimum cost flow problem, and (3) modifying the tracking graph to correct segmentation errors automatically. A detailed description of our proposed tracking algorithm is given in [3].

Coarse Tracking of Segmented Objects. Segmented objects are coarsely tracked over a time span Δt by defining a region of interest (ROI) to find potential matching candidates in successive time points.

Create Tracking Graph. A graph is constructed which models cell behavior such as movement, appearance, disappearance, mitosis, as well as the segmentation errors: false negatives, under- and over-segmentation of two or more objects. Edges are constructed between nodes corresponding to a segmented object and all nodes corresponding to its potential matching candidates. To find optimal paths through the constructed graph, and assign segmented objects to tracks, a coupled minimum cost flow problem is formulated and solved using integer linear programming. The formulation of the coupled minimum cost flow problem is inspired by [10], however with the extension to model false negatives as well as over- and under-segmentation of two or more objects. The costs assigned to edges are only based on position-based features and without previous training or parameter tuning to the selected dataset.

Correcting Segmentation Errors. The tracking graph from the previous step is modified by applying a set of graph operations: removing edges, merging tracks, and splitting tracks, to correct segmentation errors automatically.

Parameter Selection. For all datasets, we set the two parameters *delta_t* and *default_roi_size*. The former sets the maximum time span tracks with missing segmentation masks can be re-linked. The latter provides a scaling size of the ROI, resulting in a roi size of *default_roi_size* times the average size of the segmented objects within a sequence.

REFERENCES

1. Scherr T, Löffler K, Böhland M, Mikut R. Cell segmentation and tracking using CNN-based distance predictions and a graph-based matching strategy. *PLoS One* **15**, e0243219 (2020).
2. Scherr T, Löffler K, Neumann T, Mikut R. On improving an already competitive segmentation algorithm for the Cell Tracking Challenge - Lessons learned. bioRxiv 2021.06.26.450019 (2021).
3. Löffler K, Scherr T, and Mikut R. A graph-based cell tracking algorithm with few manually tunable parameters and automated segmentation error correction. *PLoS One* **16**, e0249257 (2021).
4. 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).
5. Misra D. Mish: A self regularized non-monotonic activation function. arXiv: 1908.08681 (2020).
6. Liu L, Jiang H, He P, Chen W, Liu X, Gao J, Han J. On the variance of the adaptive learning rate and beyond. arXiv: 1908.03265 (2019).
7. Zhang MR, Lucas J, Hinton G, Ba J. Lookahead optimizer: k steps forward, 1 step back. arXiv: 1907.08610 (2019).
8. Yong H, Huang J, Hua X, Zhang L. Gradient centralization: A new optimization technique for deep neural networks. arXiv: 2004.01461 (2020).
9. Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv:1412.6980 (2014).
10. Padfield D, Rittscher J, Roysam B. Coupled minimum-cost flow cell tracking for high-throughput quantitative analysis. *Medical Image Analysis* **15**, 650-668 (2011).