Best Practices in Deep Learning-Based Segmentation of Microscopy Images

Tim Scherr[1], Andreas Bartschat[1], Markus Reischl[1],
Johannes Stegmaier[2], Ralf Mikut[1]

[1] Institute for Automation and Applied Informatics,
Karlsruhe Institute of Technology, Karlsruhe, Germany
[2] Institute of Imaging and Computer Vision,
RWTH Aachen University, Aachen, Germany
E-Mail: tim.scherr@kit.edu

1 Introduction

Deep neural networks are state-of-the-art methods in image classification [1, 2, 3, 4, 5], single-object localization [1, 2, 3], object detection [2, 4, 6], semantic segmentation [7, 8], combined object detection and instance segmentation [6], and segmentation of 2D/3D biological and medical microscopy images [9, 10, 11]. Common convolutional neural networks consist of fully-connected layers, convolutional layers with fewer, shared weights operating locally, and pooling layers for downsampling [12]. Challenges in image segmentation are, for instance, inherent variation present within and among different data sets, class imbalance [13], a lack of task-specific training data [14], imperfect segmentation labels [15], and clustered and overlapping objects.

Image segmentation challenges have shown that, besides an adapted network architecture, further improvements such as task-specific data augmentation, customized loss functions, and specialized post-processing is needed, e.g., [16]. For the design of a deep learning-based segmentation, the developer has to choose the network architecture and the training process settings [17], including regularization [18], activation and loss functions, gradient descend optimization algorithms [19, 20, 21], batch normalization [22], and the corresponding hyperparameters. In contrast to shallow networks, deeper networks are able to

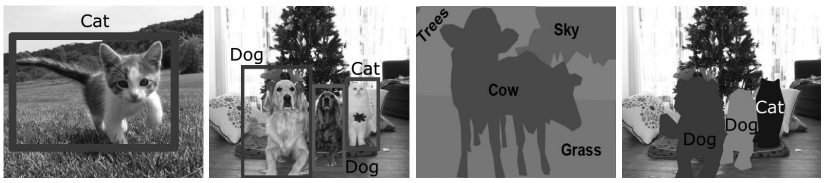
use far fewer parameters per layer to fit the training set and often generalize to the test set, but are also harder to optimize [17]. Task-specific ideal network architectures must be found experimentally, guided by the validation set error [17].

Despite the great success of deep learning in image segmentation tasks, there are only a few software tools for non-specialists, e.g., CellProfiler 3.0 [23]. This is due to the many possible combinations of network architectures, demands on the segmentation (binary, semantic, instance), fields of applications (e.g., biology, medicine), data formats (e.g., gray scale, RGB, 2D, 3D), and training process settings.

In this contribution we show how to start segmenting 2D microscopy images using a selected deep learning framework and architecture. We give recommendations to support developers in the selection of the architecture and training process settings, and show how the segmentation can be improved on the basis of the already finished Kaggle 2018 Data Science Bowl segmentation challenge [16]. This contribution is limited to 2D data, but the outcome of 2D data is useful for 3D or 3D+t data as well.

2 Network Architectures for 2D Image Segmentation

Neural networks for image segmentation are often inspired by image classification and object detection networks. Figure 1 shows the tasks of image classification (a), object localization (a), object detection (b), and of the image segmentation subclasses semantic (c) and instance (d) segmentation. In image classification, an image is assigned to one single class. The image should ideally contain only one object. Localization predicts a bounding box of the object. If there are multiple objects, the task is called object detection. Semantic segmentation partitions an image pixel-wise. Touching objects of the same class cannot be distinguished. In contrast, in instance segmentation different instances of the same class have separate labels and touching objects should be distinguished. Usually, two-stage networks with object proposals are used for instance segmentation. Common to all tasks is the requirement to find class-specific features.



(a) Classification w. Localization (b) Object Detection (c) Semantic Segmentation (d) Instance Segmentation

Figure 1: Tasks of classification with localization (a), object detection (b), and semantic (c) and instance (d) segmentation. Modified from [24].

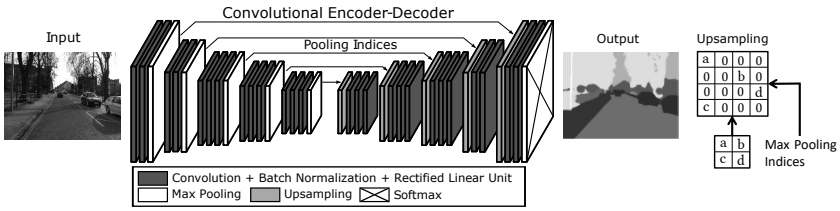


Figure 2: SegNet is a fully convolutional network for semantic segmentation [8]. The pooling indices (right) are used for the upsampling. Modified from [8].

In the following, three popular state-of-the-art architectures for image segmentation are shortly described:

SegNet is an encoder-decoder architecture commonly used for semantic segmentation [8]. Figure 2 shows that the encoder is topologically equivalent to the convolutional layers in the VGG16 network for image classification [1]. A novelty was the decoder which upsamples the low resolution input feature maps to full input resolution feature maps using pooling indices computed in the pooling step of the corresponding encoder [8]. Thus, there is no need for learning to upsample. SegNet uses the pre-trained weights of the VGG16 part.

U-Net is an encoder-decoder network that was initially developed for biological and medical image data [9]. In contrast to SegNet, corresponding encoder and decoder feature maps are concatenated. This allows successive convolutional layers to assemble a more precise output than without [9]. Figure 3 shows a slightly modified U-Net with batch normalization to fix the means and the variances of the layer inputs [22], and learnable transposed convolutions for the upsampling. If zero padding is used in the convolutional layers, there is

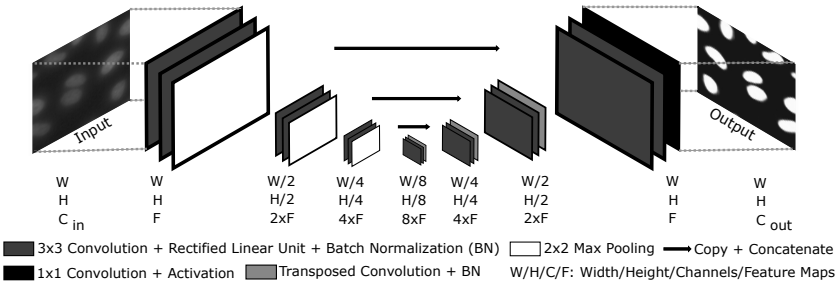


Figure 3: U-Net architecture for image segmentation. The shown network is named 3-block U-Net, as three pooling layers are used in total. Modified from [25].

no need for cropping the feature maps before concatenating as in the original architecture.

Mask R-CNN combines object detection and segmentation enabling instance segmentation [6]. It is a two-stage architecture. In contrast to ResNet und U-Net, the architecture is more complex due to its different branches: a branch for predicting segmentation masks on each region of interest in parallel with a branch for classification and bounding box regression. Recently, Mask R-CNN has also been used for the instance segmentation of nuclei in biological microscopy images [26].

3 Getting Started with Microscopy Image Segmentation

This section covers basic information for the segmentation of roundish objects in microscopy images using Python and a deep learning framework. Deep learning frameworks provide efficient implementations and GPU support for highly parallelized computations. Popular frameworks are PyTorch and TensorFlow. The high-level API Keras is capable to run on top of TensorFlow and is designed for easy and fast prototyping. This allows easy realization of concepts and enables a fast implementation, e.g., for challenges [27]. Thus, we recommend to start with a high-level API.

Table 1: Freely available training data sets for microscopy image segmentation.

Data Set	**Description**
Broad Bioimage Benchmark Collection (BBBC)[1]	Biological image data sets with various labels (counts, outlines, masks).
ISBI Cell Tracking Challenge[2]	2D and 3D data sets covering a wide range of biological cell types and image quality.
Masaryk University Cell Image Collection[3]	3D synthetic benchmark data sets generated using a virtual microscope.
Benchmarks for Embryomics [28]	Semi-synthetic benchmark generator.

[1] `https://data.broadinstitute.org/bbbc/image_sets.html`
[2] `http://www.celltrackingchallenge.net/datasets.html`.
[3] `https://cbia.fi.muni.cz/datasets/`

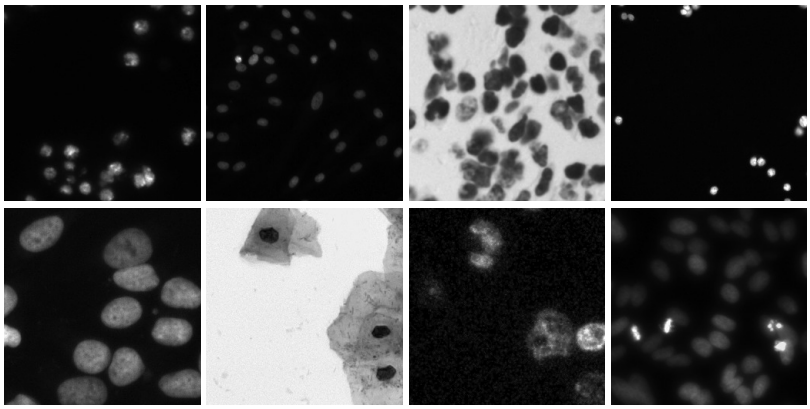


Figure 4: BBBC038v1 data set [29]. The shown images are cropped to 256×256 px.

3.1 Data Sets for Microscopy Image Segmentation

In general, larger training data sets tend to prevent the network from overfitting, but labeling images is time-consuming. Common approaches to enlarge the training set are the use of additional free data sets, e.g., those listed in Table 1, and of data augmentation (data generation using artificially transformed versions of the original data) [17]. Using 3D data slice by slice can also be useful to train a network for 2D segmentation.

In the following, the BBBC038v1 image set is used, which was part of the Kaggle 2018 Data Science Bowl segmentation challenge [29, 16]. The diversity of cell types and density in combination with a low abundance of some cell types, heavy varying image resolutions, and the rather small size of the data set, are challenges of this data set. Figure 4 shows some exemplary images.

3.2 Network Architecture

We start with a U-Net as shown in Figure 3. It is easy to implement, trainable from scratch, and expandable, e.g., using stacking and residual connections [30] or specialized encoders [31]. In [32], a U-Net model trained on only two images outperformed an advanced CellProfiler pipeline.

The strides for the convolutional, transposed convolutional and max pooling layers are set to 1, 2 and 2 respectively. Additionally, zero padding is used in these layers. The input image size of the network is fixed to 256×256 px since this is the smallest image size in the used data set. Another possibility is the use of zero padding for non-supported image sizes (due to the pooling/upsampling) and a non-fixed input image size. Using a 4-block U-Net allows the training on an Nvidia Quadro P4000 GPU using F=64 feature maps in the first layer as mentioned in [9].

3.3 Loss Functions

Choosing the loss function is important for an accurate segmentation, especially for unbalanced data sets adapted weights or loss functions are needed. Using an inadequate loss function for those data sets can result in a high false positive or high false negative rate, e.g., in background prediction for all pixels if the training data contains almost only background. Table 2 gives an overview of various loss functions for microscopy image segmentation.

3.4 Training Process Settings

A common approach is to divide the data set into a training, a validation, and a test set [17]. The training set is used to learn the parameters. The validation

Table 2: Loss functions for microscopy image segmentation.

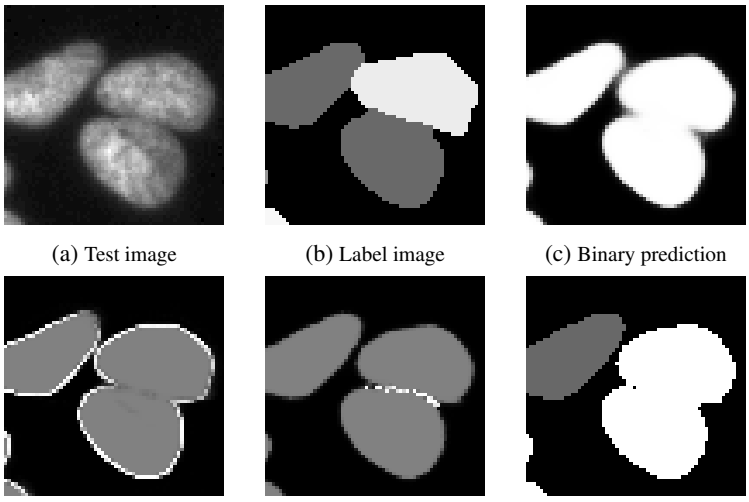
Loss function	**Description**
Binary/categorical cross entropy (`Bce`/`Cce`)	Measure for dissimilarity between prediction and label. Predictions may be a bit fuzzy.
Weighted `Bce`/`Cce`	Using weights to deal with class imbalance.
`Dice`/F1 loss [33]	Harmonic mean of precision and recall. No fuzzy predictions but problem of probabilities close to 0 or 1 even for wrong pixels [31].
Generalized `Dice` loss [13]	Generalized `Dice` loss for unbalanced data using weights based on the label area to reduce the correlation between region size and Dice score.
`BceDice`/`CceDice` loss	Idea: overcome the limitations of pure `Bce`/`Cce` or `Dice` loss [31]. Also a weighted sum can be used.

set is used to estimate the generalization error during the training and to guide the selection of the hyperparameters. The test set held back during training can then be used to estimate the generalization error after training. Since there are no labels for the provided BBBC038v1 test set, the last 120 of the 670 training images are used as test set. Before a training process starts, 20% of the remaining 550 training images are selected randomly as validation set. Small unnatural holes in the label images of the training, the validation and the test set are filled using a morphological closing.

We use the adaptive Adam optimizer [21] in the AMSGrad [34] variant (parameters: learning rate $lr = 1 \times 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.99$, $decay = 0$) and a batch size of 8. Without AMSGrad, a higher learning rate is needed. The use of callbacks enables to stop the training process after a fixed number of epochs without validation loss decrease and to save model checkpoints and intermediate results, e.g., loss and validation loss.

3.5 Data Pre-Processing

Images larger than the network input size of 256×256 px are cropped into subimages. After the training an overlap between subimages enables the combination of the predictions without boundary effects. Subimages with less than two objects are excluded from the training process to avoid a bias towards false



(a) Test image (b) Label image (c) Binary prediction

(d) Boundary method (e) Border method (f) Post-processed

Figure 5: First results using the 4-block U-Net. The predictions (c)-(e) of the test image show merged objects. The misclassification of two pixels in the border (white) in (e) causes the merging of the objects (gray). Shown are 64×64 px crops. The label and the post-processed image are color-coded.

negatives. Every single 8-bit color image $\mathbf{A}$ is converted to single precision and normalized according to:

$$\mathbf{A}' = \frac{\mathbf{A}}{127.5} - 1 \; . \tag{1}$$

3.6 First Results

Figure 5c shows first results using the 4-block U-Net and one output channel. If the data analysis demands no instance segmentation, such an approach may be fine. The sum of the `Bce` and the `Dice` loss was used (`BceDice`) and the sigmoid activation function. In tests, no clear trend towards better predictions than with pure `Bce` or `Dice` loss is visible. However, sometimes small improvements on single objects are possible. To get the final binary segmentation a simple thresholding post-processing can be applied. Selecting a fixed threshold on an exemplary prediction is fine most of the time.

If the data analysis demands instance segmentation, weight maps can be used to force the network to learn small separation borders introduced between touching objects as in [9]. Problems occur if a predicted separation border is not perfect.

Another idea is to generate boundaries from the training labels (cf. [32]), and train a network with three one-hot-encoded output classes: background, interior, boundaries. Challenges arise from missing and non-closed boundaries between the touching objects as in Figure 5d. This results in merged objects after post-processing. The reason is that a nearly closed border is quite a good result for this class. As loss function, the sum of the `Cce` loss for every class and of the channel-wise `Dice` losses for the object class and the boundary class was used (`CceDice`). The activation function was the softmax function.

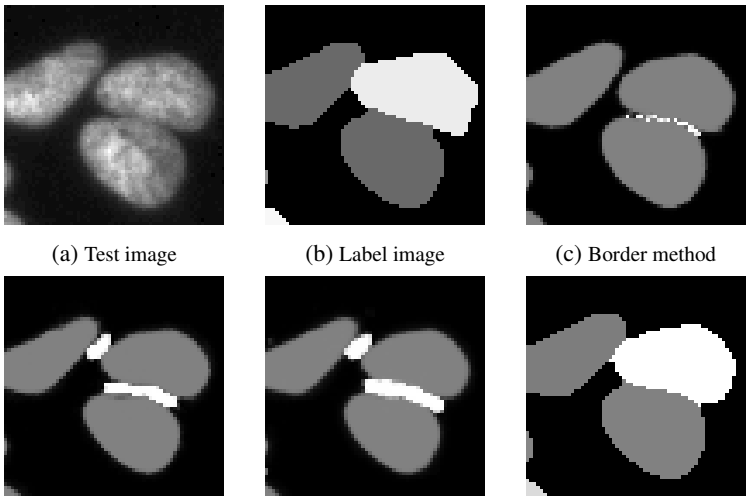
In [31], it is suggested to use the output classes: background, object, border between touching objects. Thus, the network is enforced to learn the border, where it is useful. Figure 5e shows an exemplary erroneous result using the `CceDice` loss and the softmax activation. Now there is a border in between the two objects, but it is not closed. This again can result in merged objects after post-processing. However, the idea of the border method seems to offer an elegant way to train the network and to tweak its output to a desired direction. The training borders can be generated from the provided label images.

For the instance segmentation in Figure 5h, a marker-controlled watershed post-processing was used on the border method result (Figure 5e). The thresholded ($th = 0.3$), inverted background channel was used as input image and a thresholded ($th = 0.6$) with the border channel **B** processed object channel **O** as markers:

$$o'_{ij} = o_{ij} * (1 - b_{ij}) \ . \tag{2}$$

4 Improving the Segmentation of Microscopy Images

In the previous section, it was shown how to start segmenting microscopy images using standard architectures, methods and loss functions. If the demands on the segmentation accuracy are not fulfilled yet, there is need for modificati-



(a) Test image (b) Label image (c) Border method

(d) Two-network approach (e) Two-network approach with TTA (f) Prediction (e) post-processed

Figure 6: Improving the results of the 4-block U-Net using a two-network approach. The approach improves the borders (white) and the prediction. Shown are 64×64 px crops. The images (b) and (f) are color-coded. TTA: test-time augmentation (see section 4.2).

ons of the architecture, the training process, and the post-processing. A good source of information besides publications are forums of segmentation competitions, e.g., [31] a post of the Kaggle Data Science Bowl 2018 winner.

4.1 Two-Network Approach

The border method in Figure 5e shows a reasonably good prediction but the border is not closed. A simple but powerful trick can overcome this limitation in many cases: train the network on morphologically dilated borders and eroded objects. Figure 6d shows that this simple trick provides much better seeds for the watershed post-processing. Now, borders are closed, thick and wide enough. Additionally, even on regions where borders are not necessarily needed, the network predicts borders. Advantages of this approach, in contrast to a centroid prediction as marker, are less wrongly predicted seeds and split objects since the markers are nearly as big as the real object.

Since morphological erosion is used for the training of the network, the use of the inverted background channel for the watershed post-processing would result in too small predicted objects. Thus, a second network with a one channel output trained on the original labels is needed to obtain a post-processed result with realistic object size. The thresholded ($th = 0.3$) output of this network can then be used for the post-processing. Figure 6f shows the final result combining the network used for the prediction in Figure 5c with that for the predictions in Figure 6d and Figure 6e.

4.2 Ensembling

Following the idea in the last section of using more than one network to improve the post-processing, the averaged output of multiple networks can be used to improve the final prediction. This results in an increase of training and prediction time. A simpler approach is the so-called test-time augmentation (TTA). In TTA, one single network is trained, but multiple outputs are estimated, e.g., by flipping the test image, making a prediction, and flip the prediction back. The back-flipped prediction should be nearly the same as the original prediction, but may provide additional information about borders or prediction errors. The resulting object channel **O** and the boundary channel **B** are then the pixelwise mean or maximum respectively of the corresponding ensemble channels.

For the prediction in Figure 6e flipping (up-down, left-right) and a 90°-rotation were used. The TTA prediction shows more clear borders. In cases, where the border is not closed, TTA offers a simple but powerful tool for corrections and better markers (cf. Figure 7). In some special cases, it may worsen the prediction, e.g., if two shifted borders are predicted instead of one big border, resulting in two markers and wrongly split objects. However, this should not occur with the two-network approach.

4.3 Evaluation Metrics

When the obvious segmentation errors are identified and minimized, it is beneficial to have some measure for the segmentation accuracy. This measure is

(a) Test image (b) Label image (c) Without TTA (d) With TTA

Figure 7: Improving predictions using test-time augmentation (TTA). With TTA the border (white) is complete and the objects (gray) can be separated.

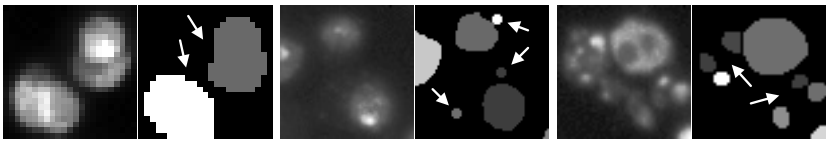
called a metric. However, a look onto the raw predictions is still useful since error sources such as too short boundaries cannot be identified in the metric score. A metric is applied to a final, post-processed result. Thus, it cannot represent the potential of a model.

Common metrics are recall, precision and F-Score. In the Kaggle segmentation challenge [16], the mean average precision at different intersection over union (IoU) thresholds is used. The thresholds t range from 0.5 to 0.95 with a step size of 0.05. A true positive is counted when a single predicted object matches a ground truth object with an IoU above the threshold. The mean average precision P_{IoU} is then calculated as [16]:

$$P_{\text{IoU}} = \frac{1}{N_t} \sum_t \frac{TP(t)}{TP(t) + FP(t) + FN(t)} , \tag{3}$$

with the true positives TP, the false positives FP, the false negatives FN, and the number of thresholds N_t. The final score $\overline{P}_{\text{IoU}}$ is the mean taken over the individual average precisions of each test image.

In [15], a critical analysis about challenges as standard validation for biomedical image analysis methods is provided. It is shown that in challenges the rank of an algorithm is generally not robust to the test data, the ranking scheme, and the observers making the reference labels. For segmentation, rankings can change a lot by using another metric. Additionally, different annotators may produce different winners. Figure 8 shows some examples for label errors in the test set.



(a) Possibly merged (b) Possibly artifacts (c) Possibly split

Figure 8: Possible label errors (arrows) in the used test set. The shown label images are color-coded which enables to distinguish touching objects.

4.4 Improving the Training Process

Data augmentation is a common method to increase the training set and to improve the robustness of a model [35]. Obvious augmentations are rotation and flipping of images since characteristic image properties are preserved. Other augmentations are, for instance, blurring, contrast changes, noise adding and affine transformations. In [31], it is suggested to copy and paste nuclei within an image to introduce more borders to the training process. Anyway, augmentations can also result in unnatural images and it is not clear whether the robustness will improve or not.

Tuning the hyperparameters of the training process can improve the segmentation accuracy as well. To tune the learning rate, schedulers can be used, which set the learning rate after a specified number of epochs to a decreased value. The reduction of the learning rate on a validation loss plateau may improve the segmentation accuracy as well. Keras provides various callbacks to modify the learning rate during training. Instead of Adam, e.g., stochastic gradient descent [20] can be used. Additionally, some regularization can be added, e.g., spatial dropout [36].

4.5 Architecture Adaption

If the segmentation accuracy is still not high enough, architecture changes may be needed. A reason for choosing the U-Net architecture was the number of existing modifications. Another promising approach is the exchange of the encoder with a pre-trained specialized image classification network and using end-to-end training as suggested in [31]. Anyway, for that also the hardware

Table 3: Overview of the trained networks used in Figure 9. The `BceDice` loss and the sigmoid activation were used to train the networks B, and the `CceDice` loss and the softmax activation for the networks M. In a two-network approach, the networks B provide the needed thresholded binary (B) images and the networks M the markers (M). The training process is stopped after 10 epochs without validation loss improvement.

Network	Blocks, F	Classes	lr	Augment.	Erosion/Dilation
B1	4, 64	1	1e-4	-	-
B2	4, 64	1	1e-4	$\times^{a}$	-
B3	4, 64	1	1e-4	$\times^{b}$	-
B4	4, 64	1	2e-4*	$\times^{b}$	-
B5	5, 32	1	2e-4*	$\times^{b}$	-
M1	4, 64	3 (boundary)	1e-4	-	-
M2	4, 64	3 (border)	1e-4	-	-
M3	4, 64	3 (border)	1e-4	-	$\times$
M4	4, 64	3 (border)	1e-4	$\times^{a}$	$\times$
M5	4, 64	3 (border)	1e-4	$\times^{b}$	$\times$
M6	4, 64	3 (border)	2e-4*	$\times^{b}$	$\times$
M7	5, 32	3 (border)	2e-4*	$\times^{b}$	$\times$

[a] flipping (left-right), flipping (up-down), 90°-rotation, noise, scale, blur.
[b] flipping (left-right), flipping (up-down), 90°-rotation.
* learning rate is quartered on a validation loss plateau (5 epochs without improvement).

has to be available. Finding the optimal cut-off layer in transfer learning may also be useful [37].

4.6 Evaluation of the Segmentation Accuracy

Figure 9 shows the averaged test set precision score $\overline{P}_{\mathrm{IoU}}$ of the networks in Table 3 which were initialized and trained only once. As expected, the two-network approach M3B1 using the networks M3 and B1 outperforms the simple border method M2 and the boundary method M1. TTA improves the precision for every network. A comprehensive study of training augmentation types with multiple initializations is planned for future work. To validate the improvement of a higher learning rate that is reduced on a plateau, more initializations are needed too. The Kaggle Data Science Bowl winner [31] reached a score of 0.631 on the official test set which includes new cell types. However, since no labels are provided for that data, a comparison is not possible. Training a similar model as used in [31] is also planned but demands a GPU with more memory than the used Nvidia Quadro P4000.

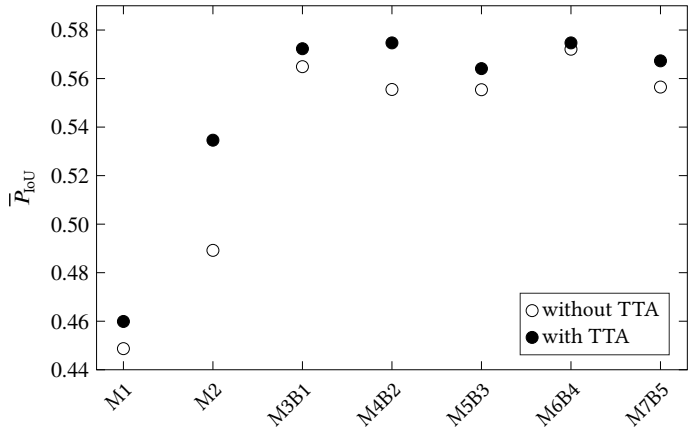


Figure 9: Mean average precision on the test set of the trained networks in Table 3. The combination of two networks B and M means that the two-network approach was used.

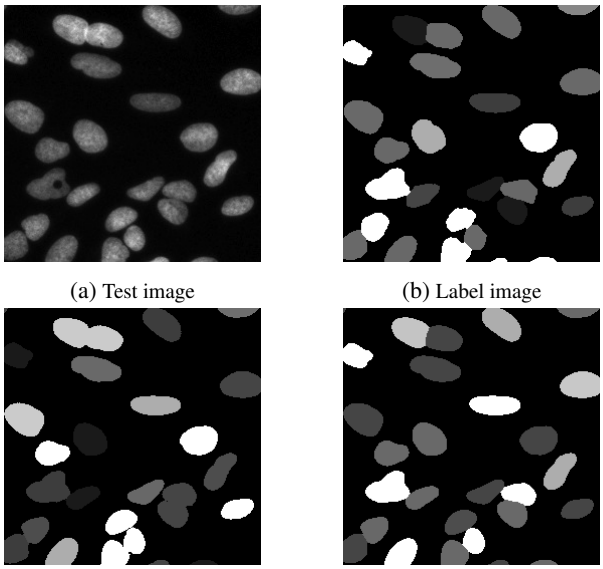
The training time on a single cropped training image is between about 0.1 s (4-block U-Net) and 0.05 s (5-block U-Net) on the used Nvidia GPU. The number of needed epochs ranges from 40 to 100 for every network. The use of a 5-block U-Net with reduced feature maps (M7B5) reduces the mean training and prediction times without losing accuracy.

Figure 10 and Figure 11 show some exemplary segmentations of test images. The network M6B4 with the highest mean averaged precision shows a better generalization to the test set as the first result M2. However, errors still occur. Further improvements may be possible with more training data similar to Figure 11a.

The use of a-priori information about the object sizes, may improve the post-processing and segmentation accuracy. However, in this data set the false negative rate may increase due to artifacts like in Figure 8.

5 Towards Segmentation of 3D Microscopy Images

One approach for the segmentation of 3D or 3D+t microscopy data is to apply 2D segmentation slice by slice and to fuse the segments afterwards. Using



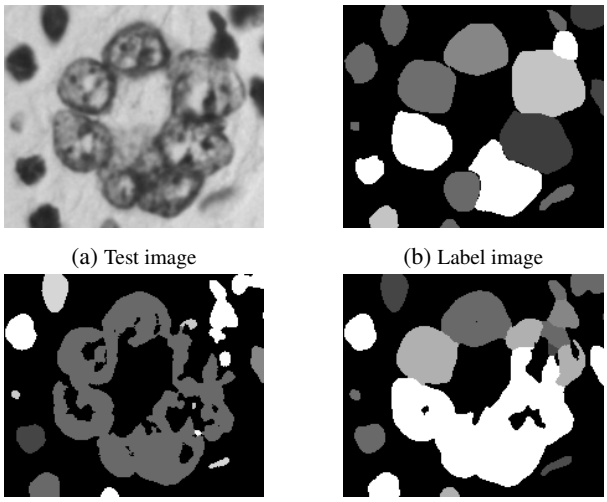
(a) Test image (b) Label image

(c) First result: M2 (d) Improved result: M6B4

Figure 10: Exemplary segmentation of a test image. This image type is frequent in the training set. Shown are 256×256 px crops and the labeled images are color-coded which enables to distinguish touching objects. In contrast to the first result (c), the improved result (d) shows no merged objects.

3D convolutional layers such as in [10] or [33], segment fusion can be omitted. Instead of border lines in 2D segmentation, border areas may provide a powerful tool for instance segmentation. Furthermore, a combination of three 2D-U-Nets for the *xy*-, *xz*-, and *yz*-slices of the 3D volume can boost memory efficiency [38]. Weight sharing of these 2D-U-Nets is possible and may reduce the training time.

A comparison of the possible ways towards 3D segmentation with a classical segmentation algorithm, e.g., [39], is planned. If for 3D or 3D+t data the use of additional information, e.g., the structure tensor which can also be used for segmentation [40], can improve the training process is an open question we also want to investigate.



(a) Test image (b) Label image

(c) First result: M2 (d) Improved result: M6B4

Figure 11: Exemplary segmentation of a test image. Objects similar to the big cells are not present in the training data. Shown are 180×210 px crops and the labeled images are color-coded which enables to distinguish touching objects. The improved result (d) shows a better generalization than (c).

6 Conclusion

In this contribution, we showed a possible workflow for starting and improving the segmentation of roundish objects in microscopy images. The hereby gained expertise should also help readers in 3D segmentation tasks and in segmentation of arbitrarily shaped objects. Summarized, our suggestions are:

1. Start with a one channel output U-Net to get a feeling how challenging the data are and if there are difficulties with specific objects.
2. Be aware of class imbalance.
3. Use borders instead of boundaries for instance segmentation.
4. Try erosion and dilation to get better markers for the watershed post-processing. Consider the two-network approach in this case.
5. Do not use training data augmentation naively. Some augmentations may worsen the accuracy.
6. Try to boost performance with test-time augmentation.

7. Use metrics but do not rely on them solely. Look at your data to find bottlenecks and difficulties.
8. Look for additional training data sets. This may improve the generalization of the network to the test set.
9. Adjust your training process hyperparameters.
10. Use a-priori information, e.g., about the object sizes, to improve the post-processing.
11. Modify your architecture and try other encoders if the accuracy is not high enough and if the required hardware is available.

A future goal is to improve the segmentation quality in such a way that in sophisticated medical and biological analyses, e.g., of biological zebrafish data using EmbryoMiner [41], no more manual corrections are needed.

References

[1] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proc. International Conference on Learning Representations*, 2014. arXiv: 1409.1556v6.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, ..., L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[5] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *AAAI Conference on Artificial Intelligence*, pages 4278–4284, 2017.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[7] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[8] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[9] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer International Publishing, 2015.

[10] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, pages 424–432. Springer International Publishing, 2016.

[11] V. Ulman, M. Maška, K. EG Magnusson, O. Ronneberger, C. Haubold, N. Harder, ..., C. Ortiz-de-Solorzano. An Objective Comparison of Cell-Tracking Algorithms. *Nature Methods*, 14(12):1141–1152, 2017.

[12] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521:436–444, 2015.

[13] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso. Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248. Springer, 2017.

[14] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. W. M. van der Laak, B. van Ginneken, and C. I. Sánchez. A Survey on Deep Learning in Medical Image Analysis. *Medical Image Analysis*, 42:60–88, 2017.

[15] L. Maier-Hein, M. Eisenmann, A. Reinke, S. Onogur, M. Stankovic, P. Scholz, ..., A. Kopp-Schneider. Is the Winner Really the Best? A Critical Analysis of Common Research Practice in Biomedical Image Analysis Competitions, 2018. arXiv: 1806.02051v1.

[16] Kaggle. 2018 Data Science Bowl, 2018. `https://www.kaggle.com/c/data-science-bowl-2018`.

[17] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[19] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. *Efficient BackProp*, pages 9–50. Springer Berlin Heidelberg, 1998.

[20] L. Bottou. *Stochastic Gradient Descent Tricks*, pages 421–436. Springer Berlin Heidelberg, 2012.

[21] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, 2014. arXiv: 1412.6980v9.

[22] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 448–456. PMLR, 2015.

[23] C. McQuin, A. Goodman, V. Chernyshev, L. Kamentsky, B. A. Cimini, K. W. Karhohs, ..., A. E. Carpenter. Cellprofiler 3.0: Next-Generation Image Processing for Biology. *PLOS Biology*, 16(7):1–17, 2018.

[24] F.-F. Li, J. Johnson, and S. Yeung. CS231n: Convolutional Neural Networks for Visual Recognition. Lecture 11: Detection and Segmentation, 2018. `http://cs231n.stanford.edu/syllabus.html`.

[25] X.-Y. Zhou, C. Riga, S.-L. Lee, and G.-Z. Yang. Towards Automatic 3D Shape Instantiation for Deployed Stent Grafts: 2D Multiple-Class and Class-Imbalance Marker Segmentation with Equally-Weighted Focal U-Net, 2018. arXiv: 1711.01506v4.

[26] J. W. Johnson. Adapting Mask-RCNN for Automatic Nucleus Segmentation, 2018. arXiv: 1805.00500v1.

[27] F. Chollet et al. Why Has Keras Been So Successful Lately at Kaggle Competitions?, 2016-08. `https://www.quora.com/Why-has-Keras-been-so-successful-lately-at-Kaggle-competitions`.

[28] J. Stegmaier, J. Arz, B. Schott, J. C. Otte, A. Kobitski, G. U. Nienhaus, ..., R. Mikut. Generating Semi-Synthetic Validation Benchmarks for Embryomics. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 684–688, 2016.

[29] V. Ljosa, K. L. Sokolnicki, and A. E. Carpenter. Annotated High-Throughput Microscopy Image Sets for Validation. *Nature Methods*, 9(7):637, 2012.

[30] A. Sevastopolsky, S. Drapak, K. Kiselev, B. M. Snyder, and A. Georgievskaya. Stack-U-Net: Refinement Network for Image Segmentation on the Example of Optic Disc and Cup, 2018. arXiv: 1804.11294v1.

[31] S. Seferbekov. *[ods.ai] Topcoders, 1st Place Solution*, 2018-04. Winner of Kaggle 2018 Data Science Bowl. `https://www.kaggle.com/c/data-science-bowl-2018/discussion/54741`.

[32] J. C. Caicedo, J. Roth, A. Goodman, T. Becker, K. W Karhohs, C. McQuin, ..., A. E. Carpenter. Evaluation of Deep Learning Strategies for Nucleus Segmentation in Fluorescence Images. *bioRxiv*, 2018.

[33] F. Milletari, N. Navab, and S. A. Ahmadi. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571, 2016.

[34] S. J. Reddi, S. Kale, and S. Kumar. On The Convergence of Adam and Beyond. In *International Conference on Learning Representations*, 2018.

[35] A. Bartschat, T. Unger, T. Scherr, J. Stegmaier, R. Mikut, and M. Reischl. Robustness of Deep Learning Architectures with Respect to Training Data Variation. In *Proc., 28. Workshop Computational Intelligence, Dortmund*, 2018.

[36] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient Object Localization Using Convolutional Networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–656, 2015.

[37] N. Prodanova, J. Stegmaier, S. Allgeier, S. Bohn, O. Stachs, B. Köhler, ..., A. Bartschat. Transfer Learning with Human Corneal Tissues: An Analysis of Optimal Cut-Off Layer, 2018. arXiv: arXiv:1806.07073v2.

[38] J. Wasserthal, P. Neher, and K. H. Maier-Hein. TractSeg - Fast and Accurate White Matter Tract Segmentation. *NeuroImage*, 183:239–253, 2018.

[39] J. Stegmaier, J. C. Otte, A. Kobitski, A. Bartschat, A. Garcia, G. U. Nienhaus, ... R. Mikut. Fast Segmentation of Stained Nuclei in Terabyte-Scale, Time Resolved 3D Microscopy Image Stacks. *PLOS ONE*, 9(2):1–11, 2014.

[40] B. Jähne. *Digital Image Processing*. Springer, 6th ed., 2005.

[41] B. Schott, M. Traub, C. Schlagenhauf, M. Takamiya, T. Antritter, A. Bartschat, ..., J. Stegmaier. EmbryoMiner: A New Framework for Interactive Knowledge Discovery in Large-Scale Cell Tracking Data of Developing Embryos. *PLOS Computational Biology*, 14(4):1–18, 2018.