# U-Net for Yeast Cell Segmentation

Martin Hanzel
martin.hanzel@epfl.ch
313710

Rastislav Kováč
rastislav.kovac@epfl.ch
309532

Martin Kostelanský
martin.kostelansky@epfl.ch
312818

***Abstract*—Computerized cell segmentation is a common task in the life sciences. It is a classification problem where the input is a microscope image of a group of cells, and the output is a matrix of integers that indicates which pixels belong inside and outside of a cell. Pixels can also be labeled to indicate precisely which cell they represent. Because of the tremendous variance in cell morphology, segmentation is a difficult problem, even when one considers only a single type of cell. We present a fully convolutional neural network based on U-Net [9] that can correctly segment an image of a yeast cell colonies.**

**We demonstrate that U-Net can be used on yeast cell segmentation tasks with 97% to 99.5% accuracy of correctly-classified pixels. We also show the performance of several variants of U-Net by varying the architecture of the neural network, applying Sobel and high-pass filters on input data, and training the network on cell borders instead of cell bodies. These variants yielded similar accuracy, but did not appear to give a significant improvement over our baseline models.**

## I. Introduction

In the last decade, convolutional neural nets have been setting new benchmarks in many domains, such as image recognition [10] and sentence classification [5]. Many neural network architectures exist, such as. recurrent CNN [8], very deep CNN [10], NN with depth-wise separable convolutions and deep residual CNN [3]. One of them, called U-Net, has been proposed in 2015 by Ronneberger, Fischer, and Brox [9]. This new "U"-shaped, fully-convolutional neural net has achieved much success in segmentation of biological images. This architecture has been the keystone for many new approaches in image segmentation [2] [7]. The authors claim that U-Net is substantially faster and more accurate than competing methods — indeed, it outperformed the runner-up algorithm in the 2015 ISBI cell-tracking challenge by "a large margin".

In our work, we evaluated U-net in yeast cell segmentation. We collaborated with Professor Sahand Jamal Rahi from the Laboratory of the Physics of Biological Systems at Ecole Polytechnique Fédérale de Lausanne.

## II. Data & Feature Engineering

Our input images consisted of several different time-series of light microscopy images, as well as manually-annotated images indicating pixels that belong in a cell.

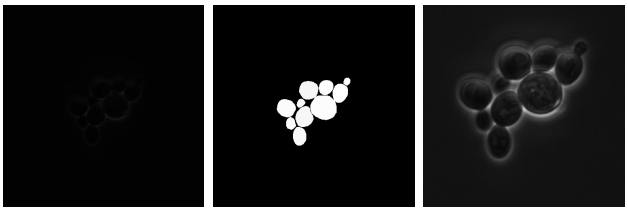

Figure 1. The sample of training data. Left: raw image (extremely low dynamic range). Middle: binarized segmentation mask, manually annotated. Right: Raw image, scaled and contrast-adjusted to show features.

We describe our data preparation process:

### A. Filtering

One of the first things we noticed is that the inner bodies of cells seldom have a constant colour. Instead, they contain gradients and blobs, and thus a cell body is not necessarily a constant feature. However, we observed that the edges of cells are conveniently highlighted due to the microscopy technique used, and we hypothesized that it may be easier to identify the borders of cells instead of the bodies.

To this end, we developed three strategies:

- **Input filtering.** We applied Sobel and high-pass filters to input images in order to attenuate regions of low change, for example, the inside of cell bodies. This also has the fortunate side-effect of highlighting areas of rapid change, like cell borders. The purpose of this strategy is to more easily map pixels to output classes — a pixel with a high derivative is unlikely to appear inside a cell body. We defined our high-pass filter to be $I - G(I, \sigma = 3)$, where $I$ is the original image and $G$ is a Gaussian blur function on $I$ with standard deviation $\sigma$. See Figure 2.
- **Labeled borders.** Our labeled images contain two classes, indicating whether a pixel belongs in a cell or not. We processed the labeled images to mark the borders of cells instead, by applying the following operation on each labeled image:

  $$C_{9\times9}(D_{9\times9}(I_y)) - I_y$$

  where $I_y$ is the original labeled image, $C_{9\times9}$ is a morphological closing operation, $D$ is a morphological dilation operation. Structuring elements were square with pixel dimensions given in the subscripts of each operation. These morphological operations were chosen to give the best visual representation of borders among our set of images while reducing any gaps where cells are too close together. The purpose of this strategy is to force

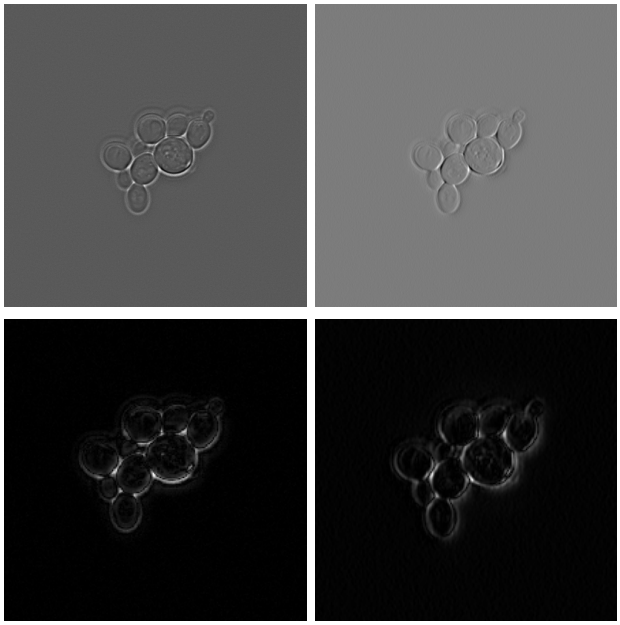

Figure 2. Example of filtered data. Left column: High-pass filter. Right column: Sobel filter. Top row: Normalization to $[0, 1]$. Bottom row: Absolute value of filters, then normalization to $[0, 1]$.

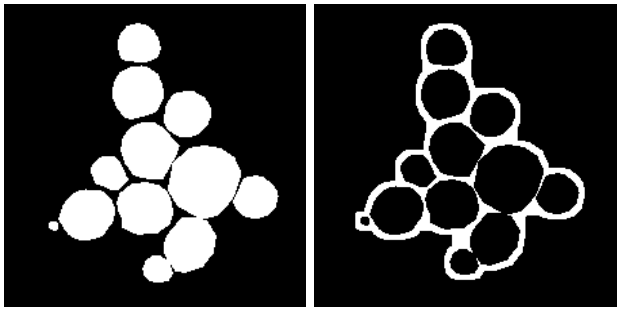

Figure 3. Sample of a constructed border-detection mask (right) constructed from provided segmentation mask (left).

the neural network to learn cell borders instead of cell bodies, in the hope that borders are easier to identify. Once the borders are known, cell bodies can be found via more traditional image-processing techniques, like blob detection and flood-filling. See Figure 3.

- **Hybrid.** We applied the Sobel and high-pass filters and trained on cell borders.

## III. Network architecture

U-net is a fully-convolutional neural network in which an input image passes through two phases: a "contracting path" of several convolution and downsampling steps, and an "expanding path" of convolution and upsampling steps. Tensors in the contracting path are concatenated with ones in the expanding path, enhancing the context around each pixel and allowing convolution to take advantage of a richer set of features. The general architecture as presented in [9] is shown in figure 4. We implemented the model as described in [9] as

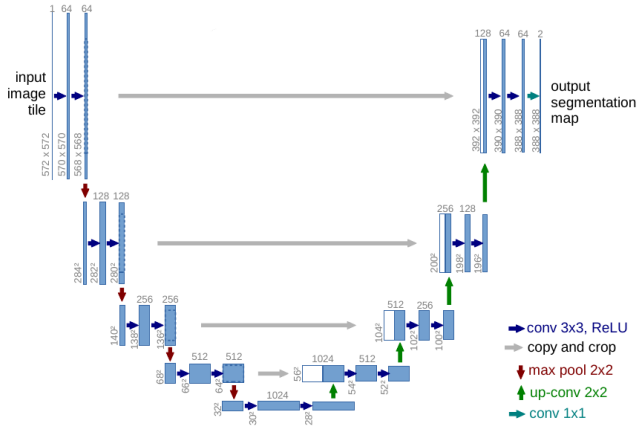


Figure 4. An overview of the architecture of U-Net. The network resembles a U shape, with a descending "contracting" path and an ascending "expanding" path.

| U-net type | Number of parameters |
|---|---|
| Small | 7,696,193 |
| Medium | 31,030,593 |
| Big | 31,093,921 |

Figure 5. The number of the trainable parameters in each of the proposed U-net architectures.

our base model (from now on called **Medium U-net**), then experimented with this architecture further and created two additional models:

1) **Big U-Net.** In this approach, we tried to use as an advantage the fact that the input images have resolution $2044 \times 2048$ pixels. We added one more layer of convolutional with 32 filters at the top of both paths of U-net. Since these blocks use only 32 filters, the dimensionality of input images changed to $1148 \times 1148$ from the original $572 \times 572$. The overall number of parameters increased only by $0.2\%$, but processing the features of this magnitude with convolutional filters with kernel size $3 \times 3$ and stride 1 increases both the training and the prediction time.
2) **Small U-Net.** Because of the limited size of training set, we proposed a version of U-net with the depth of the "U" reduced to 3, and the the number of filters in the block, starting at 32, was increased by factor of 2 after every downsapmling step, and decreased by factor of 2 in every upsampling step, so the number of filter is equal in the first and the last convolutional block. This approach radically decreased the number of trainable parameters to 7.7 million.

The complexity of proposed models is documented in figure 5. There is same final layer in all of these architectures — convoulutional layer with receptor field of size $1 \times 1$ pixel and sigmoid activation function, what describes the probability of pixel beiing cell or not.

## IV. Training

For detecting the cell bodies, we used all of the three models described in section III without preprocessing or preprocessed with high-pass filter. These models are numbered 1 to 6. In parallel, we tested the performance of Big U-Net with different types of preprocessig on border detection. These models are numbered 7 to 9. Summary of all used models:

1) Medium U-net, with standard deviation of inputs normalized to 1 and with labeled cell bodies. This is the baseline model that served as the benchmark for all other models.
2) Small U-net, with standard deviation of inputs normalized to 1 and with labeled cell bodies.
3) Big U-net, with standard deviation of inputs normalized to 1 and with labeled cell bodies.
4) Small U-net, with data prepossessed by high-pass filter and normalized standard deviation to 1 and with labeled cell bodies.
5) Medium U-net, with data prepossessed by high-pass filter and normalized standard deviation to 1 and with labeled cell bodies.
6) Big U-net, with data prepossessed by high-pass filter and normalized standard deviation to 1 and with labeled cell bodies.
7) Big U-net with visible-spectrum inputs, as in model 1, with labeled cell borders.
8) Big U-net with Sobel filter applied to the inputs, with labeled cell borders.
9) Big U-net with high-pass filter applied to the inputs, with labeled cell borders.

### A. Normalization

Because we developed these models in parallel, different group members used different normalization strategies on input images.

For the models 1, 2 and 3, the inputs were preprocessed only by normalizing the standard deviation of pixel luminance values of each image to 1. This transformation does not affect the mean of the distribution.

The models 4, 5 and 6 were trained on the data firstly preprocessed by high-pass filter and then was applied the normalization of its standart deviation.

In model 7, inputs were linearly scaled to a dynamic range of $[0, 1]$. This mainly made it easier for a human to match input images to predicted segmentations, as the input images were extremely dim — it does not affect training accuracy.

In models 8 and 9, inputs were linearly scaled to a dynamic range of $[0, 1]$, then the filter (Sobel or high-pass) was applied. Since the filter functions may return negative values, we took the absolute value of the filtered result, then again applied a linear scaling to fit values to $[0, 1]$. We took the absolute value because we are only interested in the magnitude of the image's derivative, and not its direction.

**Models trained for recognition of cells**

| | Fusions | Splits | FP | FN | Overshoot | Undershoot | True area | Predicted area |
|---|---|---|---|---|---|---|---|---|
| Big HP | 0.556 | 0.0 | 0.167 | 0.056 | 0.02 | 0.012 | 0.333 | 0.341 |
| Medium HP | 0.556 | 0.056 | 0.0 | 0.222 | 0.014 | 0.023 | 0.339 | 0.331 |
| Small HP | 0.222 | 0.167 | 0.0 | 0.0 | 0.012 | 0.012 | 0.352 | 0.352 |
| Big none | 15.222 | 0.222 | 1.333 | 6.222 | 0.044 | 0.055 | 0.142 | 0.131 |
| Medium none | 8.667 | 2.944 | 18.111 | 16.667 | 0.002 | 0.001 | 0.004 | 0.005 |
| Small none | 3.667 | 0.167 | 0.667 | 0.056 | 0.032 | 0.033 | 0.246 | 0.245 |

Figure 6. Evaluation of the watershed images, as compared to ground truth. The values have been normalized given the size of the output. Note: FP = False Positive, FN = False Negatives

**Test accuracy / precision / recall (%)**

| | Labeled cells: Small | Labeled cells: Medium | Labeled cells: Big | Labeled borders: Big |
|---|---|---|---|---|
| No filter | 94.0 / 50.0 / 40.0 | 93.9 / 38.7 / 2.0 | 99.9 / 94.5 / 96.8 | 99.11 / 74.4 / 67.8 |
| Sobel | - | - | - | 98.8 / 70.2 / 52.4 |
| Highpass | 99.7 / 97.1 / 97.0 | 99.4 / 95.9 / 93.8 | 99.9 / 94.5 / 96.8 | - |

Figure 7. Metrics of each model on the test set when a model was trained successfully. Note: while we were able to train a model for the highpass/labeled borders configuration, we were unable to test it due to Google Colaboratory crashing at a critical moment and discarding the data.

### B. Implementation

We developed our models in TensorFlow 2.0 [1][1]. They were trained using the Adam optimizer [6] and binary cross-entropy as a loss function, on Google Colaboratory[2] using its built-in GPU acceleration. Other libraries used were NumPy[3], imageio[4], and PIL[5].

For every training, our set of images was randomly split into 80% training images, 10% validation images, and 10% test images. The models trained for a maximum of 150 epochs, and stopped early after 20 epochs with no improvement to validation accuracy. In practice, training rarely reached 150 epochs.

## V. Test and results

*a) Cells detection:* Figure 6 evaluates the performance of our models on cell detections, averaged over all predictions. We could obvserve that without applying the high pass (HP) filter the predicted cells had the tendency to merge together, as predicted. The columns "Fusions" and "Splits" indicate the number of unnecessarily fused and splitted cells.

*b) Border detection:* Models trained on cell borders performed the best when detecting large cells, and in fact are capable of smoothing cell edges and even filling in small gaps between borders that don't quite touch. However, they struggle when there are many or smaller cells, sometimes missing a border entirely or failing to recognize any cell at all. In the latter case, we suspect that it is necessary to increase the resolution of the border mask image, so that borders of small

[1]http://tensorflow.org
[2]https://colab.research.google.com/
[3]https://numpy.org/
[4]https://imageio.github.io/
[5]https://pythonware.com/products/pil/

**Loss (binary cross-entropy)**

| | Labeled cells: Small | Labeled cells: Medium | Labeled cells: Big | Labeled borders: Big |
|---|---|---|---|---|
| No preprocessing | 0.0321 | 0.169 | 0.0131 | 0.0275 |
| Sobel | - | - | - | 0.0429 |
| Highpass | 0.0182 | 0.0147 | 0.543 | - |

Figure 8. Binary cross-entropy loss on the test set when a model was trained successfully. Note: see Figure 7

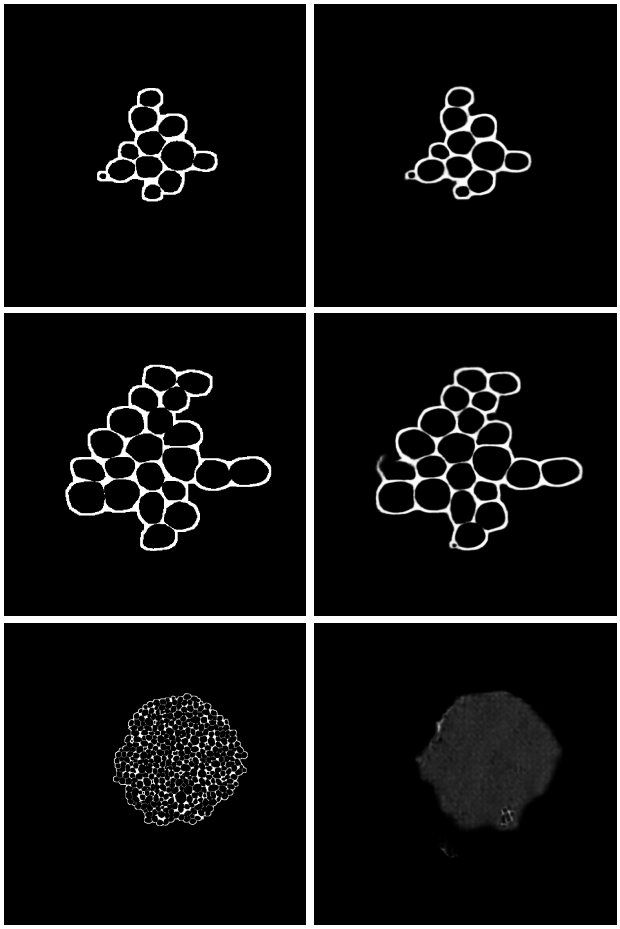

Figure 9. Left column: masks for cell borders derived from manually-annotated segmentations. Right column: the associated predictions. Notice that, in the second row, the model is capable of filling a few gaps in between borders, though it creates some gaps of its own. This particular model used a Sobel filter and the absolute value normalization discussed earlier in this paper (model no. 8).

cells are continuous and have sufficient thickness so as to be distinct from noise. Figure 9 shows results for three different inputs with Sobel filtering and the border detection.

Though we were successful in training a model that correctly identifies cell borders with the high-pass filter, our ML platform, Google Colaboratory, crashed before the model was saved, and we were not able to train another correct model in time. However, the results of all three border-detection models were very similar.

*c) Sensitivity to initialization:* We found that our models are quite sensitive to the initial weights. When performing a training run under identical conditions, we obtained drastically different results, many of which were utterly unusable.

We expect that we can reduce this sensitivity in the future by training on a more diverse set of inputs. We discuss more about this in the next section.

The weights were initialized randomly from Gaussian distribution with standard deviation $\sqrt{2/N}$, where $N$ denotes number of signals incoming to neuron from previous layer [4]. But still in some of the cases the net quickly converged to a local optimum, when all the outputs were classified as "non-cell". To prevent from this behaviour, we added Dropout [11] between every second and third convolutional layer in each block. To ensure steady convergence, we have divided learning rate by 2 every 10 training epochs.

## VI. Summary and Future Directions

We successfully demonstrated that U-Net can be used to segment a yeast cell microscopy image with very high accuracy — 97% to 99.5% of correctly-classified pixels. We extended the model of [9] by trying different sizes of the U-Net, as well as training the network on the cell border regions instead of cell bodies, obtaining very similar results. With better data, ideally of cell colonies with different shapes and from many different time-series, we expect that the accuracy can be increased even more.

One of the weakest points of our work was the training data. It consisted of three different time-series, in which several images were taken a short time apart, and thus were very similar. We cannot discount the possibility that our models are not actually learning to segment cells — if the input images were split such that every image in the validation and test sets have a similar counterpart in the training set, the models may simply be mimicking training images and "faking" high accuracy. In any case, a much larger quantity of more diverse images are needed to properly characterize these models.

We had thought that cell borders would be a more conspicuous feature than cell bodies, and so we used Sobel and high-pass filters to accentuate them, and trained directly on cell borders as well. However, we did not find that these strategies gave a significantly higher accuracy than the "plain" model.

Our data consisted of very dim microscopy images, which we linearly normalized and fed through Sobel and high-pass filters. However, this normalization exaggerates artifacts in the images. Particularly, the dynamic range of a cell border structure is a lot higher than the dynamic range of cell bodies or the background field, and normalization should account for that. A possible improvement in the future is to apply a custom luminance curve, or normalize the histogram of the image instead.

A clear deficiency of our model is that it very poorly classifies cells that are close to the edges of the image. In our view, this is not a significant problem, as images can be digitally processed to move cells into the centre of the field before predicting.

## References


[1] Martín Abadi et al. "Tensorflow: A system for large-scale machine learning". In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 265–283.

[2] Md Zahangir Alom et al. *Recurrent Residual Convolutional Neural Network based on U-Net (R2U-Net) for Medical Image Segmentation*. 2018. arXiv: 1802.06955 [cs.CV].

[3] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[4] Kaiming He et al. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. arXiv: 1502.01852 [cs.CV].

[5] Yoon Kim. *Convolutional Neural Networks for Sentence Classification*. 2014. arXiv: 1408.5882 [cs.CL].

[6] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: 1412.6980 [cs.LG].

[7] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. *V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation*. 2016. arXiv: 1606.04797 [cs.CV].

[8] Ming Liang and Xiaolin Hu. *Recurrent convolutional neural network for object recognition*. June 2015. DOI: 10.1109/CVPR.2015.7298958.

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Cham: Springer International Publishing, 2015, pp. 234–241. ISBN: 978-3-319-24574-4.

[10] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. arXiv: 1409.1556 [cs.CV].

[11] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2627435.2670313.