Counting cells in bright-field images

Benjamin Midtvedt , Peter Sveningsson and Erik Sörstadius
benmid@chalmers.se
petsvenn@student.chalmers.se
eriksor@student.chalmers.se

Project course in mathematical and statistical modelling , Chalmers University of Technology

In collaboration with Fluicell AB and under supervision of Alexey Geynts and Felix Held

April 10, 2021

Estimating the number of cells in a sample is a necessary aid to efficiently conduct experiments with custom-built cell configuration for in vitro trails. In this report, we propose a machine learning based, annotation free cell counting method, using standard bright-field imaging techniques. A U-NET architecture was used to transform input images into a cell density map, such that the sum of the output image approximates the number of cells in the image. The model was trained on bright-field/fluorescence image pairs, using an auxiliary model to transform the fluorescence images into density labels. The auxiliary model was in turn trained on synthetic data composed of elliptical disks convolved with the point spread function of the microscope. The final model is capable of human level accuracy.

Contents

1 Introduction **1**

2 Method **2**
2.1 Network output and architecture . . . 2
2.2 Network loss . . . 2
2.3 Data collection and annotation . . . 2
2.4 Simulating synthetic data . . . 3
2.5 Expanding the data set . . . 3
2.6 Transfer learning . . . 4
2.7 Image stitching . . . 4
2.7.1 Correspondency detection . . . 4
2.7.2 Homography fitting . . . 5
2.7.3 RANSAC . . . 5
2.7.4 Fine tuning homography . . . 5

3 Results **5**
3.1 Image stitching . . . 5
3.2 Graphical user interface . . . 5

4 Discussion **6**
4.1 Image stitching . . . 7
4.2 Future work . . . 7

Appendices **8**

A The U-NET model **8**

1 Introduction

Accurate counting of cell samples finds applications in a wide range of fields. For example when ensuring optimal cell density for synthetic tissue growth, or when analyzing the stress induced by the effects of some compound. Traditional approaches face several challenges, since the appearance changes drastically with the optical setup, the type of cell used, the duration the cell has been allowed to grow, and the density of the cells. Moreover, since most such systems make binary choices to determine if a certain structure is a cell, they may tend to underestimate the number of cells in more complicated arrangements where individual cells are hard to distinguish.

A shift to machine learning based solution to biomedical problems has been seen in recent years[4]. These approaches are, however, inaccessible to most users due to the high technical expertise needed to adapt the solutions to problems at hand. Moreover, while easy-to-use packages to solve related problems exist[1], they typically cannot be easily retrained on the specific types of images the end user wishes to analyze. In a recently published article they solves this issue by utilizing transfer to retrain a model that analyzes the morphology of cells using a low number of annotated images[1].

The present work provides an easy-to-use software package for counting cells in bright-field images. Furthermore, the software package includes functionality for image stitching of microscopy images as characterized by an affine camera model not rotated between instances of image capture.

2 Method

2.1 Network output and architecture

Solving problems using machine learning often boils down to formulating the problem in a way that is conductive to the use of neural networks. The output of a network is limited, and certain operations that are hard to replicate in a machine learning model may be easy to do with conventional methods. Further, for a network to learn to solve a problem, one has to be able to formulate a corresponding loss function.

One might imagine the simplest network to be one that takes an input image and outputs a number corresponding to the number of cells. Indeed, this can be solved using a convolutional model followed by a densely connected top. However, such a network can only analyze images of a single predetermined size and are hard to validate on experimental images.

In an attempt to mitigate these issues, one could define a model that transform the input image to another image of the same size using a fully convolutional approach. The output of such a network could be a binary image, where each cell is represented by a small circles of 1s, while the background is represented by 0s. The problem of cell counting is then reduced to counting the number of these regions of 1s. However, while this solves both the problems of the previous method, it creates another set of problems. Notably, it has no way of informing the user of overlapping cells, meaning that very dense regions of cells will be underestimated.

We instead propose a network that attempts to transform the input to another image of the same size, where each cell is represented by a 2D Gaussian curve, normalized such that the sum of the curve is exactly one. This can be imagined as a type of cell density map. In contrast to the segmentation approach, the network can now represent overlapping cells by outputting a Gaussian curve with twice as high a peak. In such a network, the number of cells is easily be retrieved by summing the values of each pixel in the output image.

The machine learning network used to solve this problem is based on the widely popular U-NET architecture, which was originally devised for segmentation of biomedical samples. The network is a encoder-decoder-style network, meaning it first encodes the input image to a lower-dimensional space where convolutions can be performed more efficiently, before up-sampling the encoded input to a transformed image of the same shape as the original image. For more information about the specifics of the architecture, see Appendix 4.2 Finally, the image is cropped such that the number of pixels is divisible by 16, as is required by the model architecture.

2.2 Network loss

The network is trained using a combination of two loss functions. First is a standard L1 loss,

$$l_1(\mathbf{x}, \mathbf{y}) :=< |\mathbf{x} - \mathbf{y}| >, \tag{1}$$

which enforces the desired structure of the output. To ensure that the image is correctly normalized, we also define the auxiliary loss function,

$$l_2(\mathbf{x}, \mathbf{y}) := |< \mathbf{x} > - < \mathbf{y} >|, \tag{2}$$

which directly enforces the sum of the output to be the number of cells present. The total loss is simply the sum of the two loss functions.

2.3 Data collection and annotation

Annotated images of real data were needed to train the network. A collection of bright-field/fluorescence image pairs served as the basis for the training. These images were, however, not annotated. Manually annotating images is a tedious task, which is why effort was made to bypass this step. We utilized the relative simplicity of simulating fluorescence imaging to create synthetic data which could be automatically annotated. This data was used to train a network, U-NET(A), to transform fluorescence images to a density map.

The bright-field images could then be automatically annotated by simply running the corresponding fluorescence image through U-NET(A), and using the output as a label. This, in turn, was used to train the final model, U-NET(B).

Figure 1 demonstrates this. Figure 1**a)** shows how simulated images are used to train U-NET(A). Figure 1**b)** shows how U-NET(B) is trained by using pairs of bright-field and fluorescence images, where U-NET(A) is used to convert the fluorescence image to a density label. Finally, Figure 1**c)** shows how U-NET(B) analyzes a bright-field image and outputs an estimated cell count.

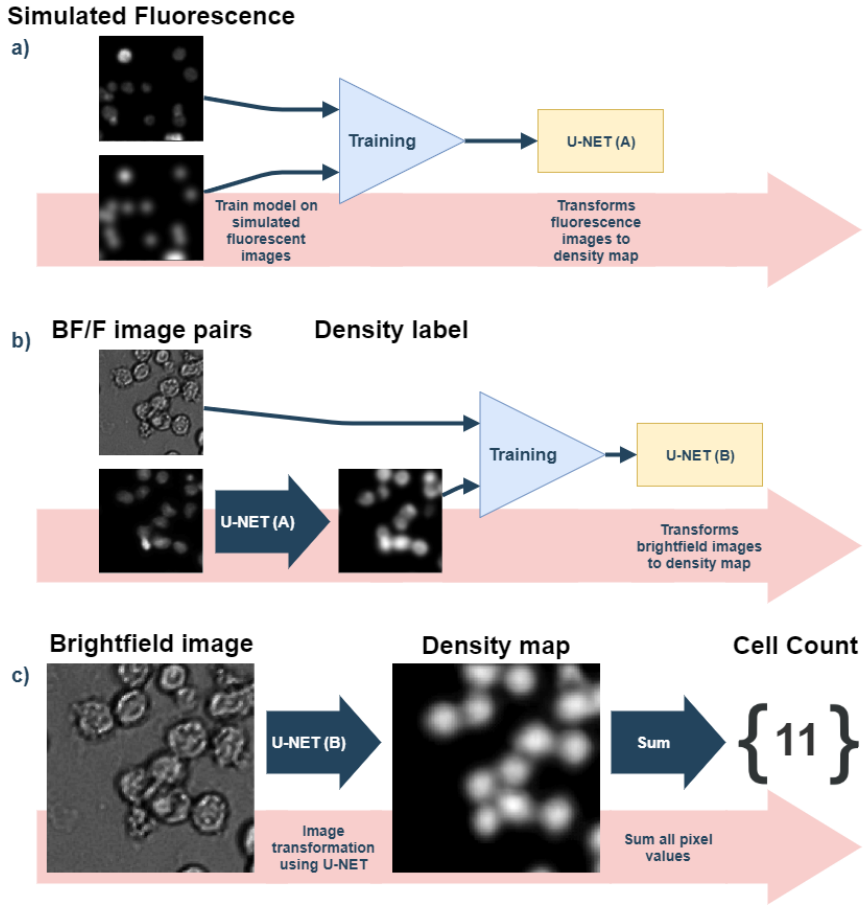

Figure 1: *Flowchart demonstrating how network models are trained and how images are analyzed.* ***a)*** *shows how simulated images are used to train U-NET(A).* ***b)*** *shows how U-NET(B) is trained by using pairs of bright-field and fluorescence images, where U-NET(A) is used to convert the fluorescence image to a density label.* ***c)*** *shows how U-NET(B) analyzes a bright-field image and outputs an estimated cell count.*

2.4 Simulating synthetic data

In the creation of synthetic fluorescence data, cells were approximated as simple elliptical disks of point sources. The intensity of the sources were simulated as highly spatially correlated noise, by

1. Providing each cell with a uniform intensity I, sampled from $I_{max} > I > I_{max}/2$
2. Introducing a Poisson distributed noise
3. Convolving this noisy image with a Gaussian kernel
4. Setting the background to zero

The final step is needed since the convolution will smear the edges of the cells, making them less sharp than in real images.

The optical system was simulated by calculating its point spread function (PSF). The PSF is a single point source, as viewed through the optical system. Fluorescence can be modelled as the sum of many point sources, which is why the intensity at the camera plane can be modelled as the distribution of point sources in the sample, convolved with the PSF of the microscope. Since the cells were all close to the focal plane, potential aberrations in the system were not modelled. The point spread function is simply the squared magnitude of the inverse fourier transform of the pupil function, which in turn can be approximated by

$$P(\rho) = \begin{cases} 1, & \rho < \frac{\mathrm{NA} \cdot d}{\lambda M} \\ 0, & \text{else} \end{cases} \tag{3}$$

where ρ is $\sqrt{k_x^2 + k_y^2}$, NA is the NA of the limiting aperture, d is the pixel size in the camera, λ is the emitted wavelength, and M is the magnification.

2.5 Expanding the data set

The training images for U-NET(B) consisted of images of real cell configurations. For every cell configuration a pair of images were supplied, a fluorescence image and a bright-field image. For many of the cell configurations several bright-field images were supplied of the same object, but with varying brightness and camera focus.

Some preprocessing was necessary to make the bright-field and fluorescent images overlap. Since the network produces a density map, non-overlapping images would result in a irreducible error, increasing the

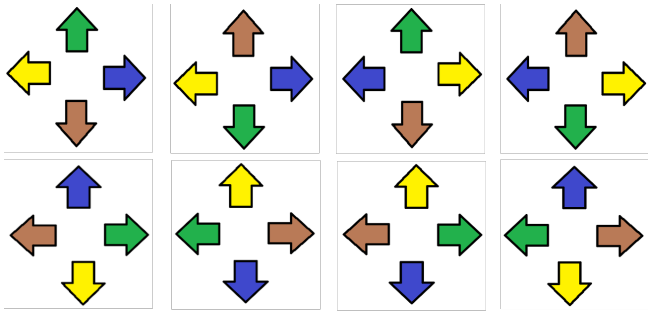

Figure 2: *The 8 euclidean transformations used on a configuration of 4 different colored arrows*

risk of over-fitting. All images were cut and fitted manually to ensure that the network could train correctly.

To increase the number of images used for training, euclidean transforms were used to increase the number of data. Euclidean transforms are characterized by preserving distances and angles. For the two dimensional images 4 mirror transformations and 4 rotation transformations were used, for a total of 16 combination of these. Because of symmetries in the transformations only 8 of these are unique. These transformations do not change the Manhattan distance between any two pixels, meaning for example that all pixels will have the same neighbours after the transformation. For the network this essentially means that the training data can be increased eightfold since the convolutional neural network only looks at local features.

2.6 Transfer learning

By full utilization of image augmentation, a pre-trained model can easily be refined to accurately analyze images outside its original scope with just a few annotated images. This is a type of transfer learning, where a trained model is adapted to perform a new task. We will also use this to train U-NET(B) by using U-NET(A) as a starting point.

One could even imagine simulating a rough approximation of the bright-field images, which could initialize the model closer to convergence, significantly reducing the amount of real data needed. This is, however, outside the scope of this project.

2.7 Image stitching

When a sample is too large to be captured in single image image stitching can be used to combine several images taken of the same scene. The methodology requires that there exists some shared features between the images.

The aim of image stitching is to find a homography H so that for any correspondence pair (x, y) in the two images

$$x \sim Hy, \quad H \in \mathbb{R}^{3\times 3},\, x \in \mathbb{P}^2, y \in \mathbb{P}^2.$$

$\mathbb{P}$ denoting the projective space. In the setting of microscopy imaging the distances to scene points are approximately constant. This lessens the effect of perspective distortions in the image and allows the camera to be modelled as affine. The affinity approximation is most easily understood as parallel lines in the scene are preserved as parallel lines in the image. Under the additional assumption that neither the scene points nor the camera are rotated between image capture we can choose a coordinate system such that the homography H takes the form

$$H = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}.$$

H is therefore an euclidean transform with two degrees of freedom. It follows that to determine H one correspondence between the two images is needed.

2.7.1 Correspondency detection

Correspondency detection is the task of finding points in two different images which depict the same object. Two methods of correspondency detection have been implemented, template matching and scale-invariant feature transform, SIFT. SIFT is the gold standard in industry. However, the algorithm is patented and this may not suit the needs of Fluicell. Furthermore, template matching provides features which are not scale- or rotation-invariant which will decrease the number of false-positive matches in the microscopy setting.

Template Matching

A template is a small sub-image cropped from one of the two images. A template is sampled and is compared to patches of equal size in the second image in a sliding window manner. For image patch I_1 and template I_2 the statistic used for the comparison is the normalised cross correlation

$$NCC\left(I_1, I_2\right) = \frac{1}{n-1}\sum_{i=1}^{n}\frac{\left(I_1\left(x_i\right) - \bar{I}_1\right)\left(I_2\left(x_i\right) - \bar{I}_2\right)}{\sigma\left(I_1\right)\sigma\left(I_2\right)}$$

which is bounded between $(-1, 1)$ with the value 1 indicating a complete match.

Correspondencies are found by pairing a template with the patch in the second image which has the highest NCC value. If the highest NCC value is smaller than 0.9 the template is discarded. Enough templates are sampled so that 30 correspondencies are found. Templates are sized at 150×150 pixels. The template size and the NCC thresh-hold are lowered dynamically if matches are not found.

SIFT

Keypoints are descriptive points in a image. These are found by applying a Laplacian of Gaussian convolution

to the image which in effect finds areas of large change in the image such as edges.

For each keypoint a descriptor is created by taking a patch of 16×16 pixels, grouping the patch into 4×4 sub-patches and calculating the numerical gradients in the sub-patches. A normalized histogram of the directionality of these numerical gradients are used as descriptors of the keypoint. These descriptors are robust to changes in illumination changes and rotation.

Correspondencies are created by matching keypoints in the image pair by their nearest neighbour in euclidean distance. If a keypoint can be closely matched to several keypoints in the paired image the keypoint is discarded. The work of Lowe et. al. found that this filtering removed $\approx 90\%$ of false matches and $\approx 10\%$ of true matches[3].

2.7.2 Homography fitting

For each correspondency pair k a homography is fitted and the squared error

$$L_2(H^{(k)}) = \sum_i \|x_i - H^{(k)} x_i\|^2 \tag{4}$$

is calculated.

The homography H is then chosen as the homography $H^{(k)}$ which minimizes the squared error (4).

Since only one correspondency is needed to fit the homography the exhaustive search approach as described above has a linear asymptotical time complexity with regards to the number of matches. The expected time complexity is decreased by employing a random sample consensus approach as described by Fischler et. al[2].

2.7.3 RANSAC

Given a set of correspondencies it is assumed that these consists mainly of low noise correct correspondencies, inliers, and a low amount of incorrect correspondencies, outliers. The RANSAC methodology samples the minimal amount of correspondencies to calculate a model. Inliers are defined as having a residual lesser than some predefined threshold. A large amount of models are fit and the model which maximizes the number of inliers is selected.

2.7.4 Fine tuning homography

The best solution found by RANSAC is a minimal solution of three correspondencies. This solution is robust against outliers but is sensative to noise. Therefore a definition of inliers is made and a least squares of the re-projection errors of the inlier points is presented as a fine-tuned homography.

3 Results

Figure 3 shows the percentage error of U-NET(A) compared to the simulated ground truth, and U-NET(B) compared to the labels provided by U-NET(A). The points appear to follow set curves due to the discrete nature of counting. In other words, the central line corresponds to a correct count, while the first curve corresponds to miss-counting by one, the second curve to miss-counting by two, etc. The fluorescence model seems to stay within two-three percent of the correct value, while the bright-field model stays within five-six percent of the value provided by U-NET(A).

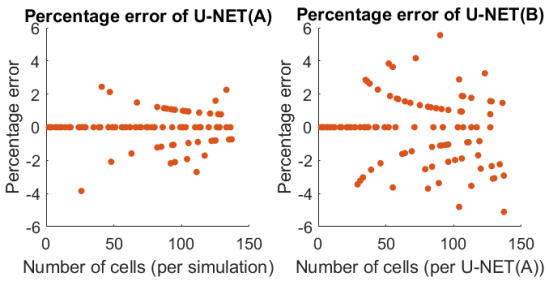

Figure 3: *The percentage error of U-NET(A) and U-NET(B) over the number of cells in the image. For U-NET(A), simulated labels were used as ground truth, while for U-NET(B), U-NET(A) provided the ground truth by counting the number of cells in the corresponding fluorescence image. Note that the curves in the image appear since counting can only be wrong in increments of whole steps.*

Figure 4 shows a few typical bright-field images and the corresponding density map generated by U-NET(B).

3.1 Image stitching

In figure 5 and 6 one can see a cropped template matched in another microscopy image. Provided one can find many of these correspondences the algorithm is able to stitch together image 7. For the dataset available the stitching algorithm using template matching was able to stitch together all sets of images. This included several examples of brightfield imaging and fluorescent imaging. The parameters template size and matching threshold proved to be important for the robustness of the algorithm. For images with a small overlapping segment the template size needed to be smaller. For fluorescent images the template needed to be large to capture many features since fluorescent features are not distinct as can be seen in figure 8. This was well handled by a adaptive decrease of this parameter if a low amount of matches was found.

3.2 Graphical user interface

The GUI is hosted by running GUI.py which is a single file implementation of a flask-based interface. The user

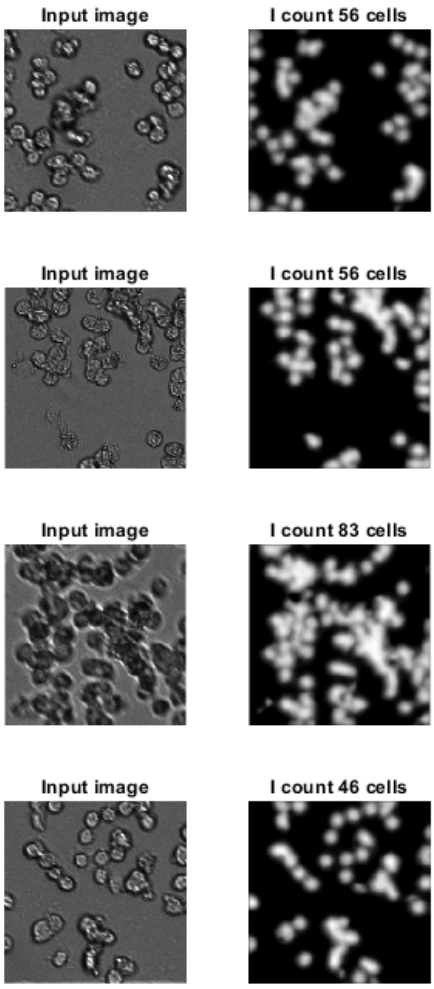

Figure 4: *Four typical bright-field images and their corresponding density map as calculated by U-NET(B). Note the large variability in the appearance of a cell.*

can then access the GUI by entering 'localhost:5000' into any modern web browser. Figure 9 shows the graphical interface.

4 Discussion

The strategy to train the U-NET(A) network to count cells in a fluorescent image has probably been the implementation that has saved the most time. Having to manually label cell images and counting the number of cells in tens of photos would be very tedious and time consuming. Realizing that fluorescent images easily can be simulated and using this unlimited number of images has contributed greatly to the accuracy of the model. Another benefit of this is that, as mentioned earlier, the initialization of the weights for U-NET(B)

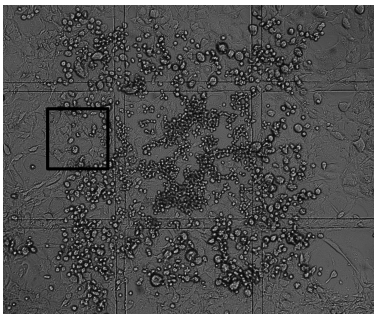

Figure 5: *A example of a template cropped out of a microscopy image.*

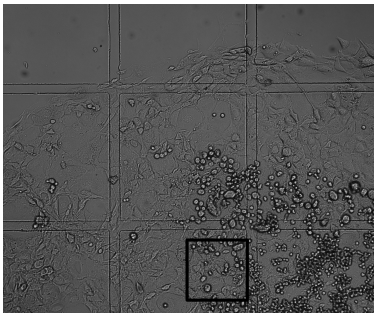

Figure 6: *A example of a template matched to another microscopy image.*

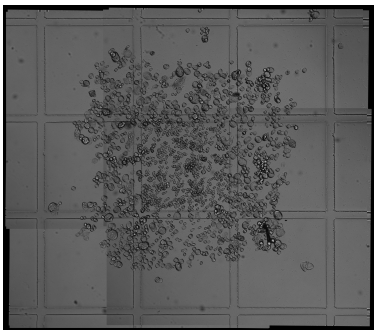

Figure 7: *A example of a fully stitched brightfield image.*

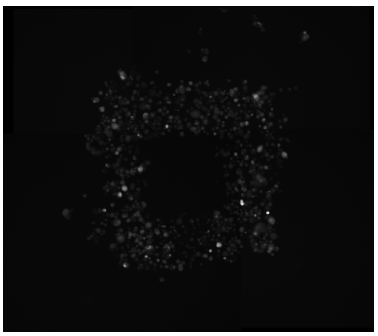

Figure 8: *A example of a fully stitched fluorescent image.*

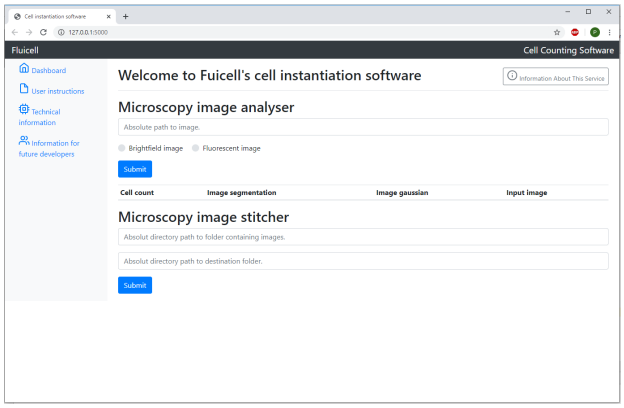

Figure 9: *Image of the developed GUI.*

was a lot better than other simpler methods.

4.1 Image stitching

For numerical reasons the RANSAC algorithm was not directly implemented in python but a C implementation from the library openCV was used. The openCV algorithm calculates an affine homography between the correspondences which includes rotations, scale and translation. This makes the algorithm less robust since rotations and scale operations are infeasible under the assumptions used for the correspondency detection.

4.2 Future work

One request by Fluicell was to stitch together images taken by a camera during printing. However, this setting provided two problems, static equipment was present in the image and because of the low video quality. The feature extraction methods of SIFT and template matching were not a successful approach in this setting. However, one could attempt to first pass the images through the U-net classification network and then attempt to stitch together the images, provided that the segmentation feature are consistent between image capture.

To count the cells in a video, another approach could be used. Cells could be tracked outside of the current frame, by using the fact that the cells are static on a moving stage. This could give a more accurate result than counting the cells in one image, since each cell would be viewed in many different frames.

The only manual processing of images were the preprocessing step where pairs of images were ensured to be overlapping. It would be beneficial to automate this step by implementing a code to recognize structures in the two images and cutting them accordingly. The fluorescent images and bright-field images are of the same configuration, meaning both will have the same inherent structure. It is therefore very plausible that such an algorithm could be implemented.

Some bright-field images supplied contained different types of cells, fluorescing in different wavelengths. These could also be preprocessed by joining two flourescing images to further increase the data, this was however not done, somewhat limiting the training data.

To further increase the data used for training the augmentation techniques could be expanded to include image distortions, arbitrary rotations, added noise, intensity transformations and rescaling.

Bibliography

[1] et al Falk T. “U-Net: deep learning for cell counting, detection, and morphometry”. In: *Nature Methods* 16 (2018), pp. 67–70.

[2] M Fischler and R Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24 (1981), pp. 381–395.

[3] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60 (2004), 91–110.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *ArXiv* abs/1505.04597 (2015).

Appendices

A The U-NET model

The U-NET model was originally developed for segmentation tasks of biological samples. Fundamentally, however, it is best imagined as performing image to image transformation. Figure 10 is a visualization of the U-NET architecture. It is similar to the older, fully convolutional network, in that it first downsamples the input using a convolutional layer followed by some pooling operation. The downsampling allows the network kernels to use information from parts further away in the image. It also reduces the memory size of the network. This is followed by an upsampling step in order to return an output of the same shape as the input. The upsampling is performed using a deconvolution kernel followed by a convolutional kernel.

The difference between the fully convolutional architecture and the U-NET architecture is this upsampling step. In U-NET, the output of the convolutional kernel of the same size during the downsampling step is concatenated to the input of the convolutional kernel after the deconvolution step. This is represented by gray arrows in figure 10. This allows the network to more easily retain positional information which may be lost during the pooling step. This makes the U-NET architecture especially useful where the output is somewhat visually similar to the input image.

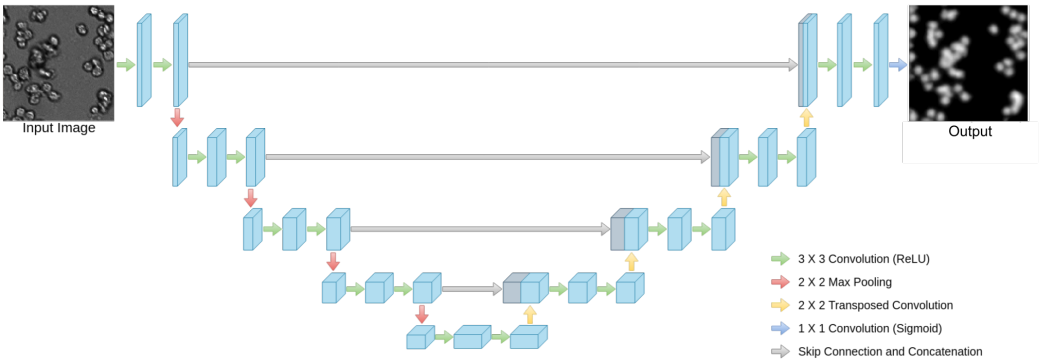

Figure 10: *A diagram of he U-NET network architecture.*