Using Convollutional Networks to Improve Biological Data Quality

Grant Sherrill
(Dated: November 26, 2023)

The study of cells is constrained by the unrestricted growth of data without advanced tools to label the data at a sufficient rate. Experiments are being designed to increase the number of pixels in a frame, leading to data sets becoming intensive to label and interpret. To aid in the process of data analysis, I am employing an augmented U-Net architecture for background subtraction. This convolutional neural network is developed for use in biological settings where images contain many different levels of illumination. There exist more robust networks able to track cells, but many run into generalizability issues and are computationally intensive to retrain. By removing the background of sample images, more traditional methods that are not computationally intensive such as thresholding can be used to segment cells where typical algorithms can process the data quickly. This approach is able to process all data coming from mother machine samples with little computation time aside from initial training. This method shows great promise in being able to remove the background, and the data can be augmented for future analysis easily. This project works to allow for data to be converted from images into quantitative measurements with greater efficiency than can be acquired without the use of deep learning.

I. INTRODUCTION

Cells and cell division are being tirelessly classified and understood by many different disciplines across the sciences, a collaborative effort to understand the human body in greater detail. As the field grows and matures, there is a rise in the need for quantitative data instead of qualitative data to describe cellular processes. Requiring numerical representations of cellular characteristics greatly increases the variety of knowledge one can get from antibiotics or gene deletion, but it introduces layers of complexity to data analysis.

Escherichia Coli doubles on average around 160 minutes, putting it out of the range of processes that are nearly instantaneous, yet changes in cell topology can happen in less than a second, such as the lifetime of cellular structures[1]. To be able to track cellular shape with accuracy, images taken of cells need to both be frequent and be of high resolution. The latter being absolutely crucial for segmentation of cellular bodies. Segmentation, the process of dividing cells into different objects, is a difficult task requiring single pixel accuracy[2]. To achieve this accuracy though would be infeasible as data sets get larger and include more pixels. Naturally, this leads one to consider using computers to automate the work. Biological data sets have many sources of error, making this a nontrivial task. To name a few, bubbles could be considered as cells, mutated cells could disguise as "model" cell, and measurement imprecision such as changes in microscope focus all contribute to sources of confusion for cell definition. Thus a computer needs to have a level of knowledge about cells and possible deformations while also being more efficient and accurate than a human. All of the above measurement ambiguities greatly affect human data analysis, and a few misclassified samples can change the conclusion of an experiment. Thus a tertiary goal is to make a program that is more accurate than a human researcher.

Cell segmentation has been a task for many long before neural networks became a prominent method of data processing, and one of the attempts to solve this issue called watershed. This method looks for regions of gray-scale images where there are edges between features via convolution algorithms. This methodology has large issues with noise in the data set, as it cannot distinguish beyond shape discrepancies. There exist some of the packages which have previously been developed to perform this task[3]. This idea of watershed is clearly akin to the principles that are employed in convolutional neural networks, they both share the techniques of using convolutional kernels to find boundaries in images, which leads one to see why they have such success in the field of biology.

There have been many attempts to use deep learning to try and classify cells, and many groups have had successes. The architecture I will be employing is the result of one of the first attempts at using convolutional neural networks to classify cells known as U-Net[2]. Getting its' name from the shape of the architecture, the U-Net was able to detect the intensity of light associated with a cell and where the cellular boundaries were. This network was not perfect by any means, and has been improved since its' release. Two recent groups have innovated on the U-Net producing Cellpose[4] and Omnipose[5]. Both projects make various improvements to the U-Net architecture, notably having support back end programs to allow users to input data of any size, the use of residual blocks, custom loss functions, and different methods of reconstructing cell shapes. The typical U-Net uses binary cross-entropy between a sample image and a target mask generated from other means (typically fluorescent tagging of cytosol). Cellpose introduces a boundary to the cell and works to calculate the center of the cell, and omnipose improves on this by introducing flow gradients which are output by the network. Both

of these improvements make the network quite robust and help solve the issue of cells typically forming oblong shapes and having large variability in shape across cell lines.

The above networks have been used by Dr. Mannik to segment cells, with great success. The segmentations are not perfect, but neither are human attempts to segment cells. Unfortunately, these networks struggle with cells that have recently divided or are in the process of dividing. The network learns that cells are pill shaped, and when they divide they appear almost dumbbell shaped. This causes problems as this stage of cell division is crucial for making measurements. Additionally the product Omnipose works when used as intended, but trying to adjust the scope of the code or the functionality is nearly impossible. Due to these factors I am working to use the U-Net to remove the backgrounds of images such that they can be processed by more elementary means such as a threshold and basic object tracking program.

As an aside, I will include here 1 which is the output from running the basic U-Net on a fairly noiseless data set. Here the labels are fluorescent tagged cytosol, which in this case was a GFP tag [1]. The masks are virtually noiseless as they are imaged in darkness and microscopes typically can filter wavelengths of light out that are not in some range as known from the flourophore being used. The outputs become subpar and there appear to be

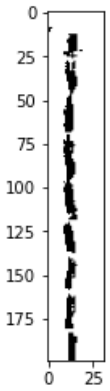


Figure 1: Output of U-Net trained on Binary Images

holes inside of the cells, which becomes a problem for data analysis. When making measurements, we need to reconstruct every point inside of a cell to be able to track the cell perfectly in time, and missing data inside of the cells or misrepresenting a boundary is unacceptable.

II. DATA SET

The data set is a set of images taken as a time lapse over the course of a couple hours of a "mother machine" device for cell analysis[6]. The mother machine device is a large number of channels running perpendicular to a central channel. The central channel can allow liquid to flow through it but that was not utilized this experiment. Each of the perpendicular channels were seeded with *e. coli* which grow and flow down the channel over the span of their life. The seeded cell typically stays inside of the channel at the dead end through the entire experiment. The channels are imaged in sets of 5, which get split into individual channels via custom MATLAB scripts[7]. Typical sizes of channels are 30×200 in pixels, corresponding to around 30×200 microns in the viewing plane.

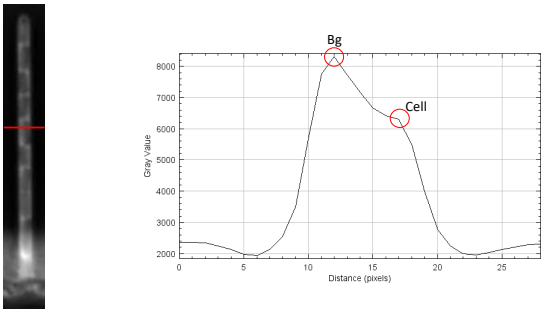
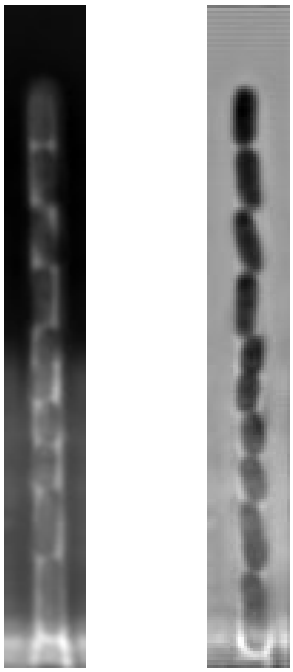


Figure 2: Comparison of Intensity Across Channel

The channels have 3 main characteristics I refer to in this work, the background, the channel background, and the cells. This can be visualized in 2 where the highest hump is the background, the middle is the cell and the background is the lowest level. The background is typically unimportant and does not affect measurement, but must be cropped near the edge leading into the main channel. The channel background is the main feature being removed by the neural network, and cells appear as the dark blobs inside of the channels. Note that channels can be either up or down, corresponding to which side the dead end is holding the mother cell. Typical data processing cannot be performed on this data set due to the different levels shown here. The map of the intensity across the length of the channel shows 3 specific regions, corresponding to the background, the channel noise, and the cells. The cells are less intense than the channel noise, but more intense than the background. Additionally the brightness of channels can sometime change parallel to the direction of motion, causing issues in typical processing.

For the network training data set, I am using the $JM161$ data set. This is unpublished data as of the time of writing this, and for access consult Dr. Mannik. This data collection was performed by Janna Mannik and is property of the Mannik lab. I have been allowed to use this data set

to try and help the lab with data processing. This data set was chosen in particular because the experiment is fairly standard without many anomalous cells. Background subtraction was previously being done by subtracting a select frame of an empty channel each frame of a data set.



(a) Original data (b) Target data

Figure 3: Input and Target Data

This works moderately well for some of the samples as seen in 3, but fails as channels drift during the course of an experiment. The target data was constructed by finding the best frames from this background subtracted data. Additionally an empty channel is included to show what generates the target data in 4. The sample has features at a relative minimum, which is inverted for feature learning despite not being necessary.

Figure 4: Empty Channel

There are other data sets available to me, which I will be referencing as unseen data, which is the $JM57$ data set and $JM219$. $JM57$ is another quite normal data set, but it is not very diverse and does not have many samples. $JM219$ has a large number of irregularities such as cells disappearing during an experiment and tilting occurring over the time lapse. These will be supplementary evaluation data, but the background subtraction for these data sets was quite poor. With this we cannot evaluate how well the network did, but qualitative judgements about resulting images can be made. This can inform decisions about the ability to use this network for processing other data sets which have not been seen by the network.

III. MODEL

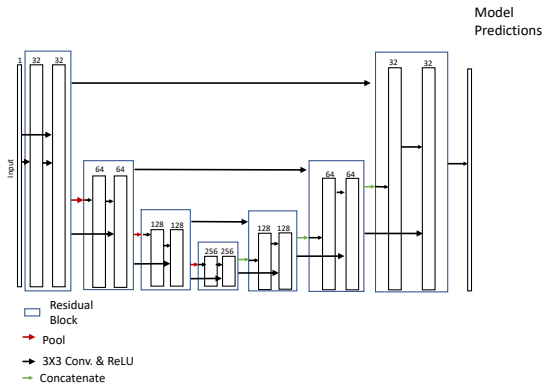


Figure 5: U-Net Architecture

The model being used in this project is a derivative of the U-Net architecture[2]. Looking at our network schematic, we have the convolutional network as described by the paper introducing this concept. Aptly named, this U shapes architecture takes in an image interpreted as a 2-dimensional array and performs many convolutional operations on it. The number at the top references the number of filters in the layer, and the corresponding block size shows the relative size of the image itself. The network condenses down to a much smaller size, learning features on the way down, and then uses nearest neighbors interpolation to return to the regular size, and convolves the features learned on the way down with the features learned on the way up. I have included schematics for the changes in dimensionality in 6 as they were not discusses as thoroughly in class with the examples showing step and filter sizes of 2 which is included in the network. In the papers that improve upon the U-Net, they implement a residual block which I include as an improvement over the U-Net. A schematic of the U-Net with the residual blocks

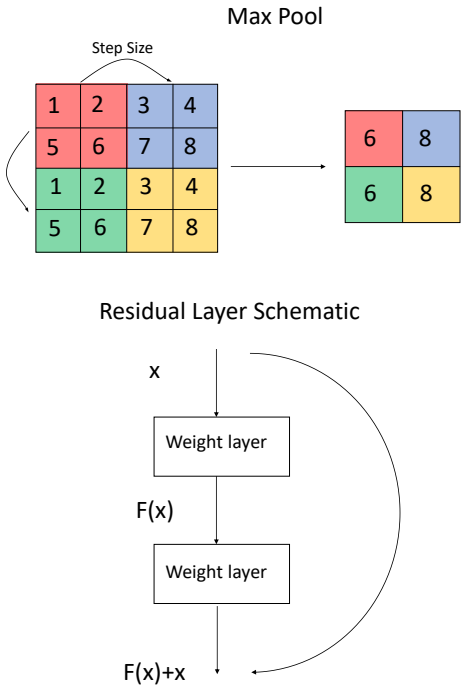


Figure 6: Max-Pool and Residual Block

included can be seen in 6. The residual block works as:

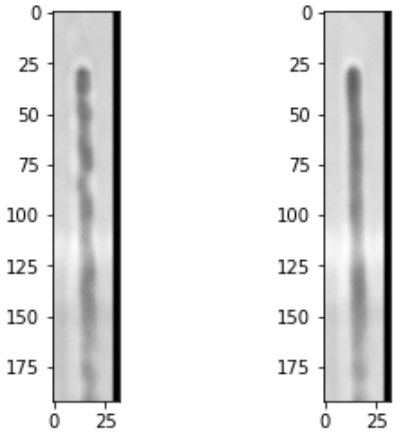
$$\begin{aligned} &\text{input} = x \\ &f(x) = \mathcal{F}(x) \\ &f(x) = \mathcal{F}(f(x)) \\ &x = f(x) + x \\ &\text{output} = \mathcal{H}(x), \end{aligned}$$

where $f(x)$ is denoting the changes to the input, $\mathcal{F}$ denotes our 2-D convolution, and $\mathcal{H}$ is our *ReLU* activation which I have chosen for the residual block. This has been shown by the paper introducing this concept to improve networks by adding some constant which the network is simulatneously training to optimize the filters[8].

Each block is a convolution or some change in dimensionality, so this network is fully convolutional. All of the activations are ReLU functions, with the final activation being a sigmoid function. An Adam optimizer is being used for this network with learning rate 0.0001.
Typically this network would be employed as a binary classifier, with the according loss function. However I am choosing to use a mean squared error loss function and have implements a sigmoid final activation. This choice was made as the target itself if not binary and the network is trying to approximate what the background subtracted data would appear as. Thus the mean square error was chosen. There are potentially other cost functions in the parameter space to be chosen, along with different optimizers and learning rates which I did not explore thoroughly in this project. This was due to the run time of the network being roughly $5min$ per epoch without the use of GPUs.

IV. BENCHMARKING

To analyze error in the predictions and evaluation data, we cannot consult a confusion matrix the same way we would for a classifier. I have taken the approach to look at a couple different quantities for accuracy: network loss and the normalized square root of the mean square estimate. These should give an idea of how far away from the target each pixel in the prediction is. This was chosen because the loss is the only quantitative way we have to investigate how well the network is learning. The choice for the NSMSE is that we want some unbiased estimator of the data that is not dependent on the normalization of the data nor the quantities of the data. The normalization was chosen as an average of the data. If most of our pixels are sufficiently close, this project will have been a success.



(a) Shallower Network (b) Deep Neural Network

Figure 7: Benchmark Outputs

Additionally, this is a qualitative project to some degree as a "good" background subtraction is what I determine to be good. During training I used the prediction function of the network to look at some of the outputs and compared them to the actual evaluation pictures as a benchmark as in 7. During training I first used the U-Net architecture, then I simplified the model down at various steps to try and reduce computational time but it did not seem to work well. I did not record the loss for these as I found them unusable regardless of loss. I am quite pleased with the U-net output, and recorded the loss at the end. I implemented the residual blocks and have some losses associated with these training steps as well. For training I used all of the same conditions except for epochs, which

started at 10 but was increased to find when the network stopped learning. I have included 8, showing that the network could potentially learn more but due to time limitations I did not push it to learn any more as it seems to converge.

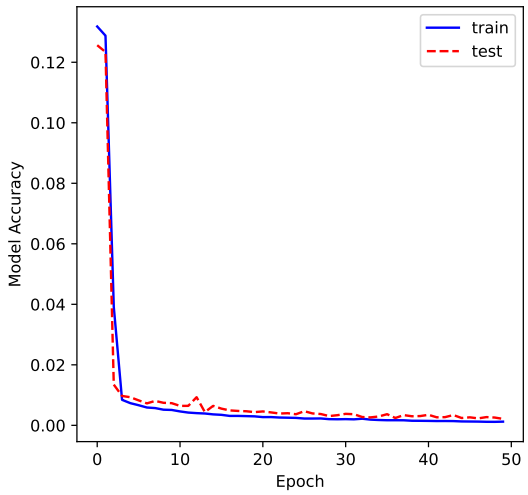


Figure 8: Caption

V. RESULTS

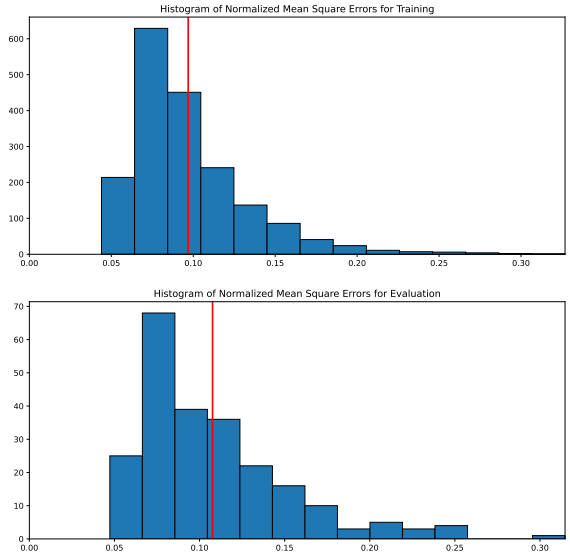


Figure 9: Histograms of Normalized Root Mean Square Difference for Training and Evaluation Data

The error in training the network was: 0.0007, with the average normalized mean squared error being 0.1. The normalized average mean squared error for the training set and the evaluation data is shown in 9, with this measure being chosen as it is an unbiased way to estimate the accuracy of the fit. These were achieved using 50 epochs to train the network, with 2200 training samples being split into 80/10/10 train, validation, and evaluation. These numbers lead us to believe that the average pixel is fairly close to what the evaluation data target is. I do not believe the network can achieve lower loss without increasing the run time even further. I should note that many of the pixels are going to be close to the target, since the pixels outside of the channel do not change with the background subtraction by a significant amount, which pollutes the estimation measure with many small numbers as the input is axiomatically close to the background. This however has no simple solution as choosing specific pixels to investigate for the measure requires us to find pixels associated with cells, which is the goal of the network and would not be an independent measurement.

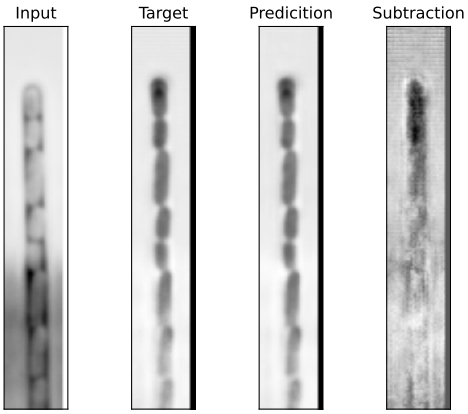


Figure 10: Input, Network Output, Target, and Difference between Output and Target

10 shows how the network predicted one of the images, compared with the target image and the target subtracted from the output. Ideally this would give us only noise, and have no identifiable structure. This does not seem to be the case, but I attribute this to the network not producing the exact values as the target. Below in 12 are two images from other data sets which had been previously unseen to the network. We can immediately see that there are still some discrepancies, especially at this cell-channel interface. The cells that the network is isolating appear to be smaller than regular cells, and may misrepresent them when taking quantitative measurements of features such as cell diameter and circumference. This being noted, it is predicting the centers of the cells quite adeptly, which is a more important feature to extract from the experiments. Analysis of mother machine devices focuses on the velocity and relative movement of cells inside of the channels, and a slightly smaller cell will still have the same relative movement to a larger cell, giving the right conclusion from the data set. There may be some issues if this data is used to determine cell health as cell

morphology is a defining characteristic of health, but it could be investigated to ensure each cell is shrunk by the network the same way leaving the conclusions about the data set unperturbed.

We can also see from the flipped channel this network is

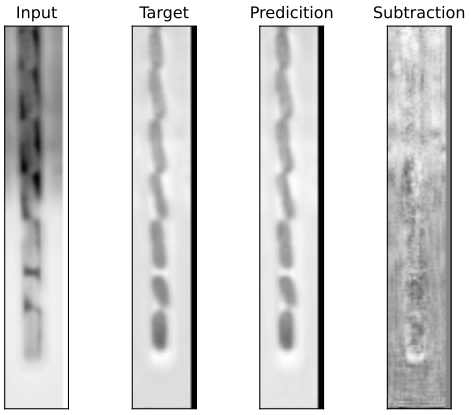


Figure 11: Dead-end Down Prediction

fairly robust, being able to look at different configurations of channels. There could be some improvements made by augmenting the data via rotations and even by performing affine transformations[9]. Considering the time constraint this project was already under this was not taken as increasing the training data set greatly increases the training time. It is not seen as clearly

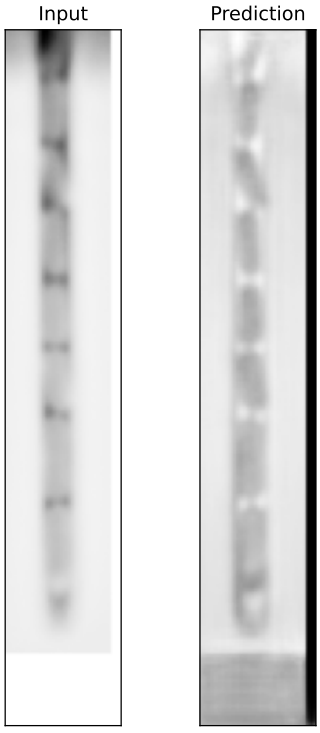


Figure 12: Unseen Data

here, but many samples have different exposure amounts, which this network is robust against. From the unseen data sets as in 12, there is not any issues with changes in exposure. The network does not seem to work as well when the cells go out of focus, but I hypothesize this is not the fault of the network but the fault of the data collection.

VI. DISCUSSION

The network demonstrated here clearly has some flaws, but are relatively minor. It is performing as well and is transforming the data into the targets with good efficiency. I believe that any sources of error or missed predictions are coming from subpar target data. Many of the samples in the target data are deemed "good" by myself, while many of them may not be of the best quality. These quality issues could include cells being out of frame, artificially subtracted, the channels drifting, and my own incorrect identification of cells. The network being trained on incorrect data will give incorrect results no matter the efficiency or accuracy of the network. I have worked to minimize this by hand curating all of the data that was included in the training set from the original source of the data. This of course introduces human based error which I cannot eliminate from the process, denying perfection to my network.

With this, I have been able to take images of cells inside a channel with a large amount of noise and remove a majority of the noise leaving the cells on a homogeneous background. This was my goal and I feel this project was successful in reaching it. There are a few natural expansions of the project that would be useful to investigate further. The first is to run this network in a sequential manner where an image can be predicted, and then this prediction can inform the network about how to predict the next image. This idea has validity because our data is not randomly created and is not arbitrary. The cells move in linear paths, and frequently do not rotate. This should mean that if a cell can be recognized in frame 1, it should be close and similar in shape to the cell in frame 2. A similar approach has been explored in another paper, but there wsa not enough time to develop this feature for this project[10].

There is another way to go from this result, which is to take the background subtracted images and track them based on the network output. The Mannik lab does most of their coding in MATLAB, and has procedures for tracking shapes from binary images. The output from the network resembles ovals on a background which can be filtered out, which would be acceptable for use with the existing scripts. This approach has issues already discussed where the shapes are not exact, but it would be a good approximation.

This has been an excellent project as I succeeded in my goals and the network is predicting what I would like it to with great efficiency. This process has also been illuminating in understanding data analysis and the cost balance associated with it. There is a balance between picking the best training data, how long the network needs to run, the accuracy of the network, and how much time is available to explore parameter space. The whole project itself is also an exercise in cost balancing, where there are programs that track cells, but they are not customizable. Dr. Mannik and I have approached machine learning to work around this while producing high quality data. Additionally I could recreate the kind of programs used by others, but that project is completely infeasible for the time frame given. All of these factors are important to recognize and think about when completing projects, and this project required me to give them all sufficient thought. With the time given and constraints intrinsic to the project, a sufficiently capable network was created to analyze data with greater efficiency.

VII. ACKNOWLEDGEMENTS

I would like to thank Dr. Mannik for providing the data for making this project possible, along with the guidance on choosing this topic and how to appraoch this problem. Additionally the computer and various scripts have been provided for this project by the Mannik lab.

[1] B. E. Walker, J. Männik, and J. Männik, Transient Membrane-Linked FtsZ Assemblies Precede Z-Ring Formation in Escherichia coli, Current Biology **30**, 499 (2020).
[2] O. Ronneberger, P. Fischer, and T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation (2015), arXiv:1505.04597 [cs].
[3] S. Stylianidou, C. Brennan, S. B. Nissen, N. J. Kuwada, and P. A. Wiggins, *SuperSegger* : robust image segmentation, analysis and lineage tracking of bacterial cells: Robust segmentation and analysis of bacteria, Molecular Microbiology **102**, 690 (2016).
[4] C. Stringer, T. Wang, M. Michaelos, and M. Pachitariu, Cellpose: a generalist algorithm for cellular segmentation, Nature Methods **18**, 100 (2021).
[5] K. J. Cutler, C. Stringer, T. W. Lo, L. Rappez, N. Stroustrup, S. Brook Peterson, P. A. Wiggins, and J. D. Mougous, Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation, Nature Methods **19**, 1438 (2022).
[6] J. Männik, T. F. Teshima, B. Wolfrum, and D. Yang, Lab-on-a-chip based mechanical actuators and sensors for single-cell and organoid culture studies, Journal of Applied Physics **129**, 210905 (2021).
[7] S. Tiruvadi-Krishnan, J. Männik, P. Kar, J. Lin, A. Amir, and J. Männik, Coupling between DNA replication, segregation, and the onset of constriction in Escherichia coli, Cell Reports **38**, 110539 (2022).
[8] K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition (2015), arXiv:1512.03385 [cs].
[9] A. Mikolajczyk and M. Grochowski, Data augmentation for improving deep learning in image classification problem, in *2018 International Interdisciplinary PhD Workshop (IIPhDW)* (IEEE, Swinoujście, 2018) pp. 117–122.
[10] M. Pachitariu and C. Stringer, Cellpose 2.0: how to train your own model, Nature Methods **19**, 1634 (2022).