THE UNIVERSITY
of EDINBURGH

Computational analysis of single-cell dynamics

Protein localisation, cell cycle, and metabolic adaptation

Diane-Yayra Adjavon

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy
to the
University of Edinburgh

THE UNIVERSITY
of EDINBURGH

2021

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise. The work has not been submitted for any other degree or professional qualification.

Diane-Yayra Adjavon

Abstract

Cells need to be able to adapt quickly to changes in nutrient availability in their environment in order to survive. Budding yeasts constitute a convenient model to study how eukaryotic cells respond to sudden environmental change because of their fast growth and relative simplicity. Many of the intracellular changes needed for adaptation are spatial and transient; they can be captured experimentally using fluorescence time-lapse microscopy. These data are limited when only used for observation, and become most powerful when they can be used to extract quantitative, dynamic, single-cell information.

In this thesis we describe an analysis framework heavily based on deep learning methods that allows us to quantitatively describe different aspects of cells' response to a new environment from microscopy data. chapter 2 describes a start-to-finish pipeline for data access and pre-processing, cell segmentation, volume and growth rate estimation, and lineage extraction. We provide benchmarks of run time and describe how to speed up analysis using parallelisation. We then show how this pipeline can be extended with custom processing functions, and how it can be used for real-time analysis of microscopy experiments.

In chapter 3 we develop a method for predicting the location of the vacuole and nucleus from bright field images. We combine this method with cell segmentation to quantify the timing of three aspects of the cells' response to a sudden nutrient shift: a transient change in transcription factor nuclear localisation, a change in instantaneous growth rate, and the reorganisation of the plasma membrane through the endocytosis of certain membrane proteins. In particular, we quantify the relative timing of these processes and show that there is a consistent lag between the perception of the stress at the level of gene expression and the reorganisation of the cell membrane.

In chapter 4 we evaluate several methods to obtain cell cycle phase information in a label-free

manner. We begin by using the outputs of cell segmentation to predict cytokinesis with high accuracy. We then predict cell cycle phase at a higher granularity directly from bright field images. We show that bright field images contain information about the cell cycle which is not visible by eye. We use these methods to quantify the relationship between cell cycle phase length and growth rate.

Finally, in chapter 5 we look beyond microscopy to the bigger picture. We sketch an abstract description of how, at a genome-scale, cells might choose a strategy for adapting to a nutrient shift based on limited, noisy, and local information. Starting from a constraint-based model of metabolism, we propose an algorithm to navigate through metabolic space using only a lossy encoding of the full metabolic network. We show how this navigation can be used to adapt to a changing environment, and how its results differ from the global optimisation usually applied to metabolic models.

Lay Summary

In order to survive, every cell in every living thing needs to be able to read and respond to what's happening in the world around it. Just as trees make leaves and flowers during spring and summer, then shed them to become dormant during the winter, so too do all sorts of cells change their behaviour based on their environment. But how do they do this? To find out, scientists will recreate these environments in laboratory conditions, and observe how the cells react to them under the microscope. The more we want to know, the more complicated the experiments. Nowadays, experiments to answer a single question can take the form of hours-long videos of hundreds to thousands of cells. Finding out one by one what each cell is doing and summarising that into numbers would be long and tedious for scientists to do manually. Not to mention, sometimes two very different behaviours look identical; human eyes are not always good at identifying subtle differences in microscopy images, even with specialised training.

Luckily, computers can do long and tedious work where humans will not. In fact, they can often do it better than us. I looked at whether the technology that brought us self-driving cars, facial recognition, and Netflix recommendations, could also be used to interpret videos of cells. I taught an algorithm to recognise certain parts of the cell and its behaviour by showing it many examples of pictures of cells, each with a biological interpretation that I wanted it to learn. Once I could rely on its answers, I then gave it a whole new set of images where I didn't know the answer in advance myself.

I used the algorithm's predictions to help me find out in more detail, what happens to baker's yeast cells when they are starved of their favourite sugar. Baker's yeasts are small and easy to grow, but they are surprisingly similar in fundamental ways to many other cells, including our own. For this reason, we often use baker's yeast to generate and test our hypotheses about biological processes before testing them on more complex creatures.

I found that the yeast cells are able to know within 15 minutes that their favourite sugar is no longer available in their environment. Once they find out, some of their behaviour changes very quickly. For example: they begin to slow down their growth as they find out what kind of sugars they can use now. Other parts of their behaviour change surprisingly slowly. They can take up to two hours to replace the machinery that they use to pick up sugar from the environment into something more suited to their current situation.

Curious about this delay, I taught two more algorithms to give us an indication of what might be occurring during that time. In particular, my algorithms can now predict how far along the cell is in its cell cycle — the yeast's equivalent of pregnancy — based only one one look at it. If it loses access to its favourite sugar during certain parts of the cell cycle, it may be more difficult for it to recover.

It was surprising how well the algorithms were able to return answers. For the most part, they return results that are better than what a human could do. At the moment, we do not know exactly how the algorithms make these decisions. In the future, I would like to look in more detail at what's going on under the hood, in order to discover what it is about the images that makes the network pick one choice over another. In doing this, I may learn new information about how the cell's shape and contents change over time as it adapts to its environment — information that I could not have obtained without the help of computers.

By continuing to create tools like these, scientists will eventually be able to offload significant amounts of work to computers. Eventually, we will work in tandem with these algorithms: they will generate hypotheses about what is going on in the videos we give them, and we will be able to check and interpret these to further our understanding of cells, and how they adapt.

Acknowledgements

To be brave enough to attempt this, I had to be tricked into believing that I could, in fact, do it. I would therefore like to begin by thanking the people who did that job. First to Stan, who planted the idea in my head and then no doubt regretted it when he had to remind me, over and over. Next to Olivier and Nicolas, who saw a computer scientist show interest in Biology and decided to encourage me rather than run away. Congratulations; it worked.

Once the activation threshold had been crossed, I needed to find the right crew for the mission. I think it's safe to say that I found exactly that at the Swain lab. To Arin, Yu, and Nahuel for their companionship and commiseration throughout the years. To Kevin and Luis who imparted the wisdom of people on the other side, people who had made it. To Ish and Ivan who, to this day, who kept me well supplied with incredible data, interesting discussions, and an ever growing list of names and abbreviations to try and remember.

A particularly heartfelt thanks goes to those who went into the trenches with me: Alán and Julian. Julian was the impetus behind most of this work, and he held my hand as I was taking my baby steps (pun fully intended) into the world of microscopy and deep learning. Alán was the light but insistent breeze that pushed me towards the end when I was getting tired. Unlike me, he hardly needed any hand-holding at all, and I am sure that already he understands some of my work better than I do. I am delighted to leave it under his stewardship.

Finally, to Peter who has somehow managed to exhibit all of the above traits at once, along with a seemingly endless amount of patience. Peter left me the space to explore my interests, held me back when I was going off too far, and gently nudged me forward when I stalled. His insight and advice is what has turned my enthusiasm into what is now hopefully good, sound science. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Yeast glucose starvation as a model for cellular stress response

Evolution in variable environments has outfitted microorganisms with a wide variety of ways to obtain the energy and components necessary to grow and divide. Due to limited resources, they have simultaneously evolved decision-making strategies, which allow them to choose how best to respond when faced with a new condition. They need to recognise what has changed in the environment, determine how to reorganise to be better prepared to grow in this environment, and make these changes.

One of the most important adaptations for microbes is efficiently getting carbon out of the environment to use as an energy source. Yeast cells are able to metabolise many different carbon sources, but they prioritise growing on glucose. As a result they will repress the mechanisms that import other carbon sources when glucose is available in the environment, they have many different transporters for different concentrations of glucose in the environment, and they have a strong stress response to a sudden downshift in glucose. By observing yeasts responding to a sudden downshift in glucose, I can look at many different facets of this environmental stress response.

1.2 Three major signalling pathways

The internal reorganisation of budding yeast cells in response to a downshift in glucose is substantial. It involves three major signalling pathways in budding yeasts: the Ras2/cAMP/PKA pathway, the TORC/Sch9 pathway, and the Snf1/PP1(Glc7)-Reg1 pathway.

Protein kinase A (PKA) responds to cAMP levels in the cell, and is active in the presence of glucose. The level of cAMP in the cell is regulated by the Ras2 protein through interaction with the adenylyl cyclase that produces it. How Ras2 activity itself is regulated by glucose concentrations is not known, but it could be linked to the change in pH that accompanies a change in glucose concentrations [11].

Sch9, one of the kinases downstream of the target of rapamycin complex (TORC) is another primary regulator of this change in cell growth in response to glucose concentrations [11, 48]. Aside from its regulation by TORC in nitrogen-limiting conditions, the exact mechanism by which Sch9 is regulated in glucose starvation is unknown. Nevertheless, it both its localisation and its abundance are affected by glucose [52]. This could be through its interaction with Snf1 [11].

The yeast AMP Kinase Snf1 is activated in absence of glucose. Although it is activated by three protein kinases (Sak1, Tos3, and Elm1), its activity is likely more closely regulated by its deactivation by Protein Phosphatase 1, PP1(Glc7), in conjunction with Reg1 [11]. The Reg1-PP1(Glc7) complex forms in the presence of glucose. How glucose levels affect the formation or activation of the Reg1-PP1(Glc7) complex is still an open question.

There is significant overlap and cross-talk between these three regulatory pathways in glucose starvation. PKA and Sch9, for instance, have overlapping downstream targets [11]. This overlap may be due to a difference in timing. PKA can quickly regulate response to a sudden glucose downshift, whereas the slower response of the TORC pathway may be used to fine-tune the cell's steady-state based on the precise external conditions [64].

Furthermore, the kinases directly interact. Sch9 is phosphorylated by Snf1 [11]. Snf1 and PKA have cross-talk in both directions, with Snf1 regulating cAMP content in the cell [79] and PKA regulates the upstream kinase of Snf1, Sak1 [8]. The regulation of the glucose starvation response is therefore complex and involves many feedback loops; this complexity is likely the reason for the efficacy and reliability of the response.

1.3 A general stress response

To begin with, the cell undergoes a general stress response when there is a sudden downshift in glucose. This response is shared across several different types of stress, and consists in reducing growth and stopping expensive process in order to divert energy to the adaptation. Two important effectors of stress response linked to PKA are Msn2 and Msn4. These transcription factors trigger a coordinated change in gene expression in response to several different types of stress, including carbon stress. Msn2/4 affect over 200 genes with a wide range of functions including carbon and amino acid metabolism, protein folding and synthesis, transcription, and generation of energy [14]. Their effect is proportional to their concentration in the nucleus, but is also encoded in their dynamics, including the frequency of their translocation [36, 69]. In the presence of glucose PKA activity represses Msn2/4 nuclear localisation by phosphorylation. When glucose is removed from the environment, PKA activity decreases, allowing Msn2/4 to enter the nucleus and trigger the response.

A specific, systematic response to stress in yeast cells is a sudden reduction of growth. Cell growth is proportional to ribosome synthesis, specifically the expression of Ribosomal Proteins (RP) and Ribosome Biogenesis (RiBi) genes [103]. The expression of these genes is jointly regulated by PKA and Sch9, through the transcription factors Dot6, Tod6, and Sfp1 in particular [11, 47, 48].

Dot6 and Tod6 are paralogs. They are phosphorylated by both Sch9 and PKA. In high glucose, they are therefore localised in the cytosol. In response to the glucose downshift, they translocate to the nucleus where they repress both RiBi and RP genes. Conversely, Sfp1 will activate the expression of these genes. As such, it is localised in the nucleus in high glucose, and cytosolic in low glucose. PKA and Sch9 both activate Sfp1. These dual signals will ensure that the cell transiently slows down its protein production, and this is accompanied by a sharp decrease in growth rate.

1.4 Arresting the cell cycle

In addition to slowing down their growth, budding yeast cells may also try to arrest their cell cycle if they can.

1.4.1 The budding yeast cell cycle

The budding yeast cell cycle consists of four phases: G1, S, G2, and Mitosis. During Gap 1, or G1, cells increase in size but their DNA content remains the same. New-born cells tend to have a longer G1 phase than established mothers, as they need to reach a certain size before they can begin to produce daughters of their own [119]. The transition from G1 to S phase is determined by the regulatory checkpoint START, at which cells commit to a new cell cycle [41].

After START, the cell begins to form a bud on its membrane which will grow to become a daughter cell during S phase. A septin cap forms to mark the location of the bud, which then turns into a ring by polarised exocytosis as the bud begins to form [87]. This ring forms a diffusive barrier between mother and bud membranes [13], and a scaffold for the formation of an actin-myosin contractile ring [95]. This mother-daughter link is called the bud neck. From this point in the cell cycle, growth is directed mostly in the bud [67]. Growth occurs in an apical manner during G1/S, pushing growth out from the point where the septin cap first formed [96]. DNA replication occurs during S phase, along with a duplication of other chromatin organisation proteins including histones [9].

After DNA replication, the cell enters into another gap phase, G2. Bud growth becomes isotropic during this phase and into Mitosis. G2 is generally short in budding yeasts, especially during fast growth.

The daughter cell gains its independence from the mother during Mitosis. Mitosis occurs in several steps. Prophase and metaphase prepare the nucleus for separation, forming separating spindle and correctly orienting the nucleus in the cell. Unlike for many other eukaryotic cells, the nuclear membrane is kept intact during mitosis in budding yeasts [67,84]. During anaphase, the nucleus and its contents are split between the daughter and mother cells. After anaphase, the cell undergoes an additional checkpoint to ensure that all necessary components have migrated into the daughter. Then, the cell undergoes cytokinesis. The actin-myosin ring around the bud neck contracts and the mother and daughter plasma membranes and cell walls are split [70]. This point corresponds to the birth of a new cell.

1.4.2 Quiescence in response to glucose starvation

Cell cycle arrest in response to glucose starvation is most prevalent before START. In unfavourable conditions, a cell in G1 will forego passage through start, and instead enter the quiescent state G0 until the conditions change [37].

The G1/S transition is inhibited by transcriptional repressor Whi5 during G1, which represses the expression of hundreds of genes involved in DNA synthesis. In particular, it does so by repressing the Swi6 transcription factor which is required for the expression of these genes [51]. The cyclin Cln3, in a complex with Cdc28, hyperphosphorylates Whi5, causing it to leave the nucleus and de-repressing the G1/S genes. The activity of the Cdc28-Cln3 complex is enhanced in a positive feedback loop by the Cln1 and Cln2 cyclins.

Cln3 activity is proportional to cell growth, through a coupling with RiBi/RP protein levels [53]. In other words, a slow growth rate will prevent Cdc28-Cln3 from activating the G1/S transition. The coupling between Cln3 and growth rate is thought to rely on the chaperone protein Ydj1. Ydj1 is involved in the release of the Cln3-Cdc28 complex from the endoplasmic reticulum, and at the same time is involved in growth processes [2, 29].

Additionally, passage through START is regulated directly by Snf1, through direct phosphorylation of Swi6. Finally, cAMP levels affect activity of Cln1 and Cln2 cyclins [2, 12].

1.5 A change in nutrition strategy

Finally, there is an actuation phase to the cell's glucose starvation response, that consists in adapting to the new environment so that it may reverse the changes made above.

1.5.1 Regulation of transporter expression

Yeast cells contain many hexose transporters (Hxts) with different affinities to glucose, of which Hxt1-Hxt7 are the most important for growth on glucose. These are regulated in order to fit the current nutrient condition [12, 76]. Hxt1 and Hxt3 are low affinity transporters, they are most suited to high levels of glucose in the environment. Hxt6 and Hxt7 are high affinity transporters, better suited to low levels of glucose in the environment. Hxt2 and Hxt4 have medium-to-high affinity, and are particularly expressed during glucose down-shifts [76]. Hxt5

has medium affinity, and is expressed in low-glucose conditions, including no glucose.

Additionally, two sensors Snf3 and Rgt2 are present in the membrane to sense glucose. These are similar to Hxts, but they cannot transport sugars into the cell [12, 90]. They also have different affinities for glucose; low-affinity Rgt2 is used to sense high levels of glucose, whereas high-affinity Snf3 senses low levels of glucose [89].

Levels of glucose sensed at the membrane are then translated into the required expression of hexose transporters through Rgt1, Mth1, and Std1. Mth1 and Std1 are active in absence of glucose. When bound to the transcription factor Rgt1, they repress the transcription of Hxts through recruitment of global regulators [91, 93] When glucose binds to the Snf3 and Rgt2, Mth1 and Std1 are inactivated, Mth1 by degradation [30] and Std1 by formation into condensates [112]. Rgt1 activity is additionally regulated through phosphorylating by PKA. In the presence of glucose, PKA phosphorylates Rgt1, dissociating it from the global regulators [54, 102].

1.5.2 Arrestin-mediated endocytosis of transporters

In addition to changes in expression, hexose transporters which are no longer optimal are removed from the membrane to the vacuole for degradation. The transporters are marked for endocytosis by ubiquitylation. This process requires the involvement of alpha-arrestins: adaptor proteins that bind to hexose transporters in specific conditions, bringing E3-ubiquitin ligase Rsp5 to the membrane. All of the hexose transporters 1 through 7 except for Hxt5 have known interactions with alpha-arrestins in changing glucose conditions. In all cases, the hexose transporters subsequently undergo endocytosis; they do not when the alpha-arrestin, or Rsp5, is deleted. Alpha-arrestins are also involved in the endocytosis of other plamsa membrane proteins [81, 86].

There is no unified understanding of the regulation of hexose transporter endocytosis in yeast. Nevertheless, complete mechanisms have been elucidated in glucose-starvation conditions for some transporters. Arrestin activity is regulated both transcriptionally and post-transcriptionally, through ubiquitylation and/or phosphorylation. Their regulation involves Snf1 and PP1(Glc7)-Reg1, as well as the PKA pathway [45, 85, 86, 101, 113].

Alpha-arrestin activity is regulated by Snf1 both transcriptionally and post-transcriptionally [45, 85, 86], balanced with regulation involving PP1(Glc7)-Reg1 [45]. Endocytosis of Hxt3 is also

regulated by the Ras2/cAMP/PKA pathway [113] in glucose to ethanol shifts, as is that of Hxt1 in glucose starvation [101].

1.5.3 De-repression of alternate carbon pathways

As yeasts preferentially use glucose over other sugars, the presence of glucose inhibits the metabolism of other sugars including galactose, maltose, and sucrose. Unlike the transcription factors above, this response is glucose-specific rather than general [36]. The transcription factors Mig1 and Mig2 are responsible for much of this inhibition [57]. In presence of glucose, the Snf3/Rgt2 pathway activates their expression [55] and localise in the nucleus. They inhibit the GAL, MAL, and SUC pathways, the oxidative carbon metabolism pathway required to metabolise non-fermentable carbon sources such as ethanol and amino acids [22, 108], as well as some of the hexose transporters [55]. When glucose is removed from the medium, Mig1 undergo phosphorylation by Snf1 that causes its removal from the nucleus [121], while Mig2 is degraded [68].

1.6 Fluorescence time-lapse microscopy data

Many of the phenomena involved in the budding yeast carbon stress response are inherently spatial: molecules move to and from the various compartments in the cell in response to internal and external stimuli. Furthermore, they can occur over a wide array of time periods, from minutes to hours. Fluorescence time-lapse microscopy, combined with micro-fluidics devices, allows us to follow the full response in single-cells: from the pre-growth in a glucose-rich environment, to the transition into a glucose-poor environment, to the adaptation, until recovery.

High-throughput micro-fluidics devices give the ability to follow single yeast cells over an entire experiment [17,23,126]. They can be used to collect single-cell time-series of fluorescent signals, and bright field images showing the general morphology and growth of the cell. We can follow cells as they bud and divide.

Fluorescence imaging adds information on the internal workings of the cell. By tagging proteins of interest with a fluorescent marker, we are able to see when they are expressed as well as where they are located within the cell. For a quantitative description of their localisation dynamics, additional fluorescent markers are often used to tag different compartments of the cell.

These types of experiments produce large quantities of rich data, too much for manual analysis. As such, we need to turn to automated methods of analysis. Although classical forms of image analysis have been used to analyse these types of data in the past [7], they make only limited use of the richness of the data. I sought instead to apply deep learning methods to automate and enhance the analysis of time-lapse microscopy images. In particular, I attempted to maximally use the information available in bright field images to increase the range of analyses available for time-lapse fluorescence microscopy. This work includes both automation of image analysis, and the development of novel analysis methods.

1.7 Deep learning

Deep learning is a subset of machine learning, whereby neural networks are used to model relationships between data modalities. These relationships are usually described as a machine learning task, among which are classification and generation.

Classification tasks consist in labelling input data with a pre-determined class. A common classification task is ImageNet [104], which takes images as an input and outputs one of thousands of labels to describe the object in the image. Some examples of classification tasks for bio-image analysis are give in subsection 1.8.3, and evaluate the performance some classification networks for cell-cycle annotation in chapter 4.

Segmentation tasks are an extension of classification. In segmentation, each pixel of the input image is given a label, so that objects of a given class are not only classified but also located within the input image. This is a common task in bio-image analysis, used to find cells or sub-cellular compartments within images. Some examples are given in subsection 1.8.1. In chapter 3, I develop two segmentation models to find organelles in images of budding yeast cells.

Generation tasks consists in creating synthetic examples of a given type of data. A classical generation task takes noise as input, and generates images of animals. The goal is to create an image that is as realistic as possible. In some cases, we want to generate an output that is conditional on a specific input; this task is called image translation. In image translation, the network simultaneously needs to learn to stay true to the input, while generating an output image that is realistic. I describe specific image generation tasks in bio-image analysis in subsection 1.8.2, and develop an image generation model in chapter 4.

1.7.1 Components of Neural Networks

The general model for deep learning is an artificial neural network (ANN), or simply neural network. A neural network consists of mathematical operations applied sequentially to an input to obtain a desired output. The atomic unit of such an network is called a neuron. It consists of a set of inputs, weights, and a bias. The output of a neuron is a weighted sum of its inputs and its bias. It is often combined with a non-linearity or activation function, which is applied to the above sum. This process is illustrated in Figure 1.2a.

I plot three examples of activation functions in Figure 1.1 The standard activation function is a Rectified Linear Unit (ReLU); it is usually used in the middle of networks. At the output of the network, other activation functions may be used depending on the desired numerical range of the output. For example, the Sigmoid activation function is often used for classifications and segmentations, where the output needs to have values between 0 and 1. When the output needs to be between -1 and 1, the Tanh activation function is used instead.

The weights of a neuron determine the relative importance of the inputs, and often go to zero when a given input is not useful to determine the output. The bias serves as a decision threshold for the activation function. For example, if using the ReLU activation function, all inputs that are below 0 are squashed to 0. By adding a bias to the neuron, we can shift this decision threshold. For example, if the bias is equal to -1, then the weighted inputs need to be greater than 1 in order for the neuron to have a positive output value.

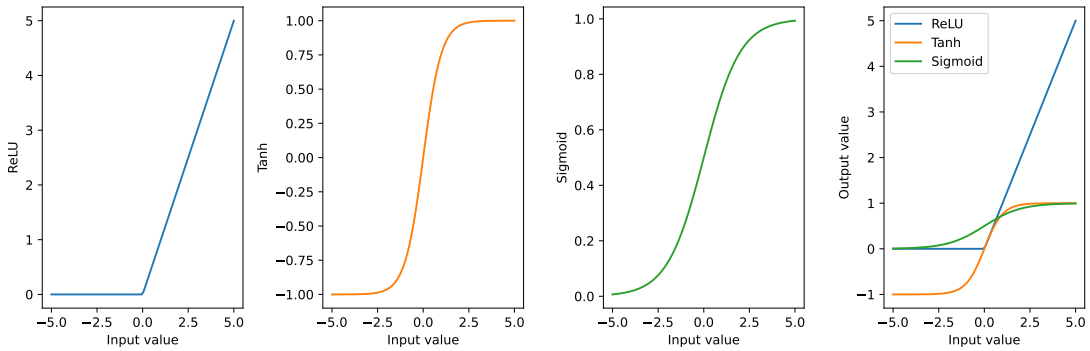


Figure 1.1: **Activation functions.** Three commonly-used activation functions. Activation functions introduce non-linearities into the network. (a) The rectified linear unit (ReLU) removes all negative values from the output, pushing them to 0. Above zero, it is proportional its input. (b) The hyperbolic tangent (Tanh) forces values into a range of -1 to 1. This range is tractable for the neural network, and is often used as a data normalisation range. (c) The Sigmoid activation function forces values into a range of 0 to 1. This is useful when the output of the network is a probability, such as in classification and segmentation tasks.

Neurons are combined into layers. A layer consists of a set of neurons that all take the same inputs. Layers of neurons are stacked into networks; often the output of one layer becomes the input of the next. The superposition of layers is what makes deep learning "deep". When making predictions with a neural network, we see only the data from the input and the output. All other layers in the network are referred to as hidden layers. This creates a network like the one seen in Figure 1.2b.

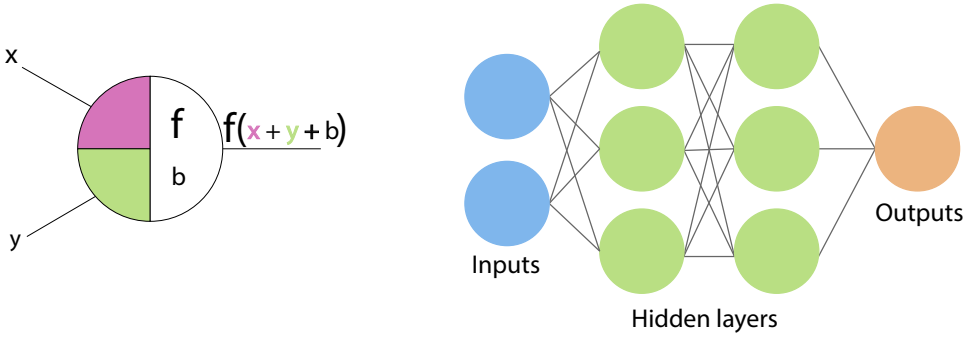


Figure 1.2: **A single neuron and a schematic neural network.** (a) Illustration of a single neuron, with two inputs and an activation function f. The inputs are weighted (colours) and summed together with the bias b before being pass through the activation function. (b) An illustration of a small neural network with two inputs, two hidden layers, and a single output. This network is fully connected or dense.

Different data types and different tasks require different types of layers. There are some standard layers used in the neural networks in this thesis. The simplest is a Dense Layer, also known as a Fully Connected. In a Dense layer, all of the outputs in one layer are connected to all of the inputs in the next layer. The example in Figure 1.2b is made up of only dense layers. Each neuron in the first hidden layer has two inputs, because there are two inputs in the input layer. Similarly, each neuron in the second hidden layer has three inputs, because there are three neurons in the first hidden layer. Additionally, these neurons have as many weights as there are values in their inputs. This kind of layer is often used toward the end of a network for classification tasks, but it is ill suited to tasks like generation or segmentation where the input can be large and does not have a pre-defined organisation, like images.

The next type of layer is a Convolutional layer. This is particularly important for analysis of images. It is made up of convolutions. A convolution consists of a kernel and a set of biases. Unlike a dense neuron where there needs to be as many weights as there are inputs, in a convolution the kernel is of a fixed size and can cover any size of input. This is done by moving the kernel across the input, essentially sharing weights across the input. Convolutions are a great way of matching a pattern, described by the kernel, at any point in the input. An

illustration of a convolution in two dimensions is shown in Figure 1.3. An intuitive idea of what this looks like is given later, in chapter 2(Pipeline chapter), section 2.3(BABY section). Networks containing convolutions are called Convolutional Neural Networks (CNN).

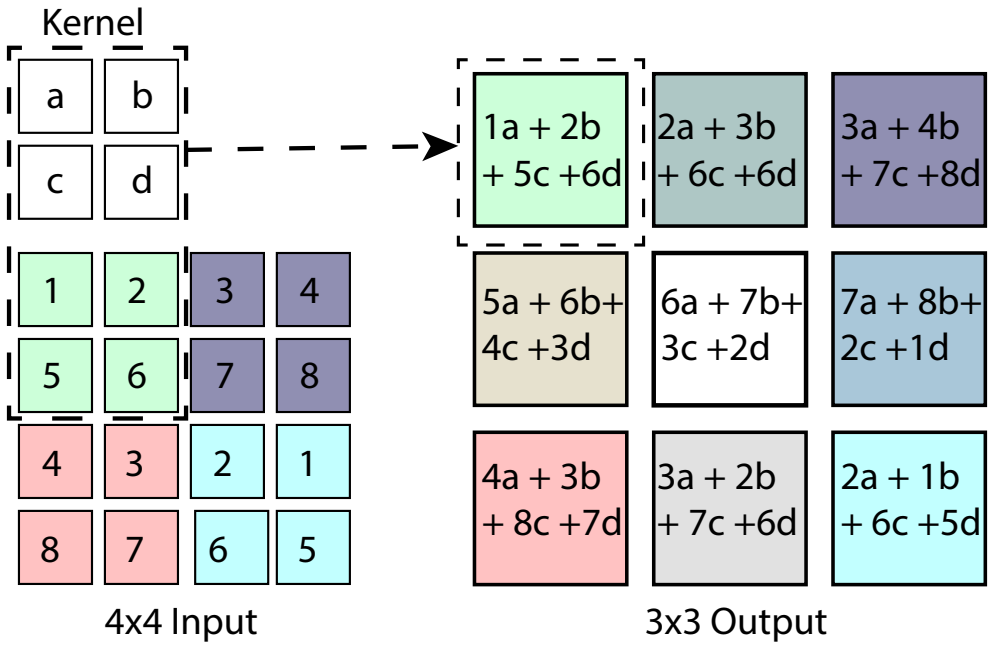
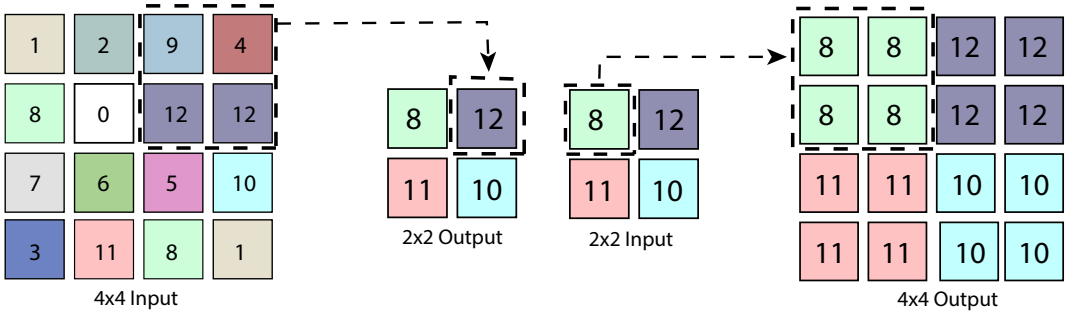


Figure 1.3: **A convolution.** On the left is the input and a convolutional kernel. The kernel is applied to 2 by 2 sections of the input, and slid over the input one step at a time to produce the output. Each pixel of the output is therefore a weighted sum of the inputs. Not shown are biases or activation functions.

Neural networks often also include changes in scale. They contain down-sampling layers called pooling layers. This operation consists in reducing the size of its input by pooling together adjacent values in the input; in this thesis I exclusively use maximum pooling. In maximum pooling, a set of adjacent values in the input are represented in the output by only the maximum value. I show an example of maximum pooling in Figure 1.4a. The opposite upsampling operation is used when we want to increase the size of the input. This is usually done by simply repeating the value of the input in multiple adjacent parts of the output, as shown in Figure 1.4b, although there are more complex alternatives to this [27].

1.7.2 Stochastic gradient descent

Training a neural network like those in this thesis consists of giving it a large number of input-target pairs, and updating its parameters such that it can recreate the target from the input. A loss function is used to determine how well the network is recreating the target.



(a) Maximum pooling operator.

(b) Up sampling operator.

Figure 1.4: **Scale-changing operations.** (a) Down-sampling using a Max-pooling operation. Each pixel of the output is the maximum value of a 2 by 2 subset of the input. Here the operator is slid over the input with a stride of two pixels at a time. (b) Up-sampling by repetition of the input. Each individual pixel in the input is turned into a 2 by 2 section of the output. It is a lossy inverse of the Max-pooling operation.

The parameters of the network are updated using a stochastic gradient descent algorithm. A forward pass through the network creates a prediction from the input. The loss compares the target to the prediction; it corresponds to the gradient at the output layer. The gradient with respect to the network's internal parameters are obtained by back-propagation. This consists of a backward pass through the network, starting from the loss at the output. At each step, the gradient with respects to a neuron's parameters is obtained from the gradients at its outputs, and then back-propagated to its inputs.

There are different optimisers which apply different methods of updating the parameters based on the gradients. In all of the examples of this thesis, I use the same optimiser with fixed hyper-parameters. In order to speed up training, the network is not shown the entire data set at each optimisation step. Instead, it is trained on batches of data. As such, its gradient it uses is a noisy estimation of the actual gradient over the entire dataset. This is what make the gradient descent stochastic.

When it is trained like this, the network learns the relationship between input and output from the set of examples that it is given, called the training data set. If left to train too long, the network may over-fit the data. This means that in addition to learning the requested task, the network also learns the noise in the training data. Over-fitting improves the loss of the network on the training data set, but it decreases generalisability. The network will have a worse performance on inputs that are not a part of the training data. To avoid this, an additional data set is included for validation. This validation data is not used in the updating of weights, but is regularly used to verify the performance of the network. When the loss on

the validation data stops decreasing, training stops even if the loss on the training data is still decreasing, to avoid over-fitting.

1.8 Deep learning for bio-medical image analysis

Convolutional neural networks produce state-of-the-art results in many computer vision tasks, and they are increasingly being used to analyse microscopy data with a wide range of applications. Several image denoising algorithms have been recently developed with the aim of improving the signal-to-noise ratio of fluorescence images, replacing deconvolution [63,65,129]. Existing work has used deep learning for segmentation of objects in fluorescence images [28, 73, 100, 106]. At even higher resolution, deep learning methods have allowed segmentation of organelles in Electron Microscopy volumes [43, 44], and of neurons in the same data [83]. Classification networks have also been used, to classify cell types by looking at localisation dynamics [60, 62]. For recent review on deep learning for bio-medical image analysis see [16, 65, 75].

1.8.1 Segmentation of yeast cells in micro-fluidics devices

In this thesis, I will be focusing on the task of image segmentation. Segmentation of cells with convolutional neural networks has been applied to budding yeast cells. Two types of segmentation are considered. Semantic segmentation consists of assigning a class to each pixel in an image. In the case of semantic segmentation of cells, we obtain a mask of values 0 for background pixels and 1 for pixels that correspond to a cell. Instance segmentation additionally separates different instances of cells: it returns a set of labels where each cell is individually numbered.

The YeaZ pipeline used a semantic segmentation network called U-Net to segment budding yeast grown in colonies [26]. From the semantic segmentation obtained by U-Net, they then obtain instance segmentations by a three-step post-processing pipeline. Their pipeline is trained on both bright field and phase contrast data.

Two segmentation methods are compared in [94]: a semantic segmentation based on U-Net, and an instance segmentation based on a Mask-RCNN architecture. Their data consists of yeast cells in micro-fluidics devices, and they train their networks to segment cells, traps and background pixels in their images. Using the Mask-RCNN architecture, they are also able to do

instance segmentation directly through the network, without loss of performance on the class segmentation.

We developed an alternative instance segmentation pipeline for budding yeast, based on U-Net, trained on images of yeast cells in micro-fluidics devices with various trap morphologies. I briefly describe this method in section 2.3.

1.8.2 Fluorescence from bright field

Several studies have used image generation to obtain predictions of fluorescence images from transmitted light microscopy.

Christiansen *et al.* [21] use a modified version of the Inception network [118] to predict fluorescent labels from transmitted light images. The labels they predict define the nucleus position, cell types, cell health, cell membrane, and some sub-cellular labels. Their training data includes wide field and confocal images of human motor neurons, confocal images of primary rat cortical cultures, and confocal images of human breast cancer cells.

Multiple U-Net networks are trained in [88] to predict the location of different sub-cellular compartments from bright field, in three-dimensional volumes. Their tool includes 14 different models, predicting targets markers of the DNA, nucleoli, nuclear envelope, actin filaments, mitochondria, cell membrane, and endoplasmic reticulum, among others. They work on human embryonic kidney cells and human fibrosarcoma cells. Their input is three-dimensional z-stacks of data acquired on a spinning-disk confocal microscope; their output is also in three dimensions.

1.8.3 Cell cycle annotation

In the domain of ageing studies, several works have attempted to classify cells into various stages of their replicative life spans from bright field images.

[32] compared different classifier for cell counting in micro-fluidics devices. Specifically, they trained classifiers to classify images with a single mother cell, a mother and a daughter cell, or a trap with more than two cells.

The very recent DetecDiv [5] framework determines budding yeast cell replicative life spans by counting cell divisions. Their cells are grown in micro-fluidics devices, and can be followed

throughout their entire life-span. They combine a convolutional neural network for extraction of features from individual images, with a long short-term memory network to combine features across time points and predict points of division, as well as other time-related information. In particular, they are able to classify cells of interest into several classes including mother, budding, daughter, and dead. They further combine this set up with semantic segmentation networks to simultaneously obtain cell segmentations from bright field images, and nucleus segmentation from fluorescence.

1.9 Neural networks beyond images

Deep learning is also able to make associations across modalities that would be too complex for human annotators. Yang and colleagues [133] were able to translate across modalities, generating RNAseq data from images of Chromatin and vice-versa. As such they correlated gene expression patterns corresponding to two sub-populations of T-Cells to phenotypic markers in chromatin images: both generating a hypothesis as to how the chromatin formation enables a given cell type and creating a marker for separating live cells into the two sub-populations.

Deep learning can also be used to directly model biological processes. Decision-making in cells is usually accepted to occur within the gene regulatory network. Watson and colleagues [128] used neural networks to model the evolution of a gene regulatory network in response to selective environmental pressures. They find that the selective pressures that occur in the creation of a diverse set of phenotypes are analogous to cognitive learning constraints.

1.10 Using Deep Learning to extract information from complex data

Deep learning methods have shown promise in their ability to interpret complex biological data. In this thesis, I will show how this ability can be leveraged to increase the scope of information obtained from bright field images in time-lapse microscopy.

In chapter 2, I introduce the experimental platform in more detail, and describe the software suite designed to access and pre-process the data.

In chapter 3 I describe a deep learning algorithm based on the U-Net architecture to predict the location of organelles in bright field images of yeast cells in micro-fluidics devices, acquired on a wide-field microscope. Our network is trained on segmentation labels obtained from fluorescence images. I then apply it to quantify the dynamics of transcription factor localisation, and hexose transporter endocytosis, in response to sudden glucose starvation in budding yeast.

In chapter 4 I develop various computational methods for the label-free annotation of cell cycle phases in budding yeast cells at various granularities. I discuss the limitations of these techniques and give directions for future work.

In chapter 5, I create a framework to describe adaptation from the point of view of metabolism. From a genome-scale constraint based model of metabolism, I use a neural network to generate a small, tractable space that faithfully describes the full metabolic space of the cell. I then describe an algorithm to traverse this space using limited, noisy information.

Chapter 2

A pipeline for image analysis

From raw pixel data to quantitative time-series

2.1 Introduction

With micro-fluidics devices we can perform time-lapse microscopy of single budding yeast cells across long time-scales, with high time-resolution [23]. We routinely image cells for up to 18 hours, with an image taken every two to five minutes on multiple channels. The cost of this is increased analysis. The data is complex, contains various levels of hierarchy, and there is too much for manual annotation. On the other hand, plentiful, information-rich data is ideal for the application of deep learning. Deep learning algorithms require large amounts of training data. A large part of setting up deep learning algorithms for analysis is therefore ensuring reliable and automated methods to access, clean, pre-process, and annotate data [109].

To this end, I designed and implemented an image analysis pipeline in python to retrieve data from storage and pre-process it for segmentation. I contributed to BABY, a cell segmentation, tracking, and lineage-assignment framework built around a convolutional neural network. I designed and implemented a storage format for the data output from BABY and the pipeline that is shareable, reasonable in size, and has fast read-write access. I further contributed to

extraction and signal post-processing pipelines to fully automate image analysis from raw data to quantitative time series, both within and without the context of training neural networks. The pipeline is partially backwards compatible: it is able to read the results from previous experiments analysed using the MATLAB pipeline [7].

In the following chapter, I begin by describing the data that we work with, and give the requirements for the pipeline. I then describe the first steps of data access, organisation, and pre-processing before segmentation. Next I describe the steps of BABY, and how the pipeline interacts with it in order to obtain segmented and tracked cells with mother-daughter relationships. I then justify the main design choices of the storage format used, including some benchmarks. I finish by giving examples of time series extraction and post-processing, to show how they fit into the framework above.

2.1.1 The micro-fluidics device

We run fluorescence time-lapse microscopy experiments in the ALCATRAS micro-fluidics device [23]. This device contains multiple chambers, each with a media input and output. In this thesis, all chambers take in the same media (single input), and the waste out-flow is collected in the same place as well. The device is illustrated at different hierarchical levels in Figure 2.1; the arrows show the direction of media flow. The media input to the device is connected to an external switch and several pumps, which allow fast switching from one condition to another [36]. This can also be leveraged to create slow changes (ramps), but this is outside of the scope of this thesis.

Each chamber of the device can hold a separate strain. When working with single-tagged strains, we can therefore image as many different proteins of interest as there are chambers, under the same media conditions. Assuming that the cells are genetically identical aside from the fluorescent tag, we can make inferences on the interactions between proteins at a population level. For more direct interactions at a single-cell level, we use strains with multiple fluorescent markers (up to two, in this thesis).

The device is mounted on a moving stage that can be shifted in three dimensions. The field of view of the microscope encompasses a sub-part of a chamber, which we refer to as a (stage) Position. A Position is fully determined by the stage coordinates in the x-y plane. The movement of the stage over time is imperfect, so there can be a drift in the true x-y coordinates of

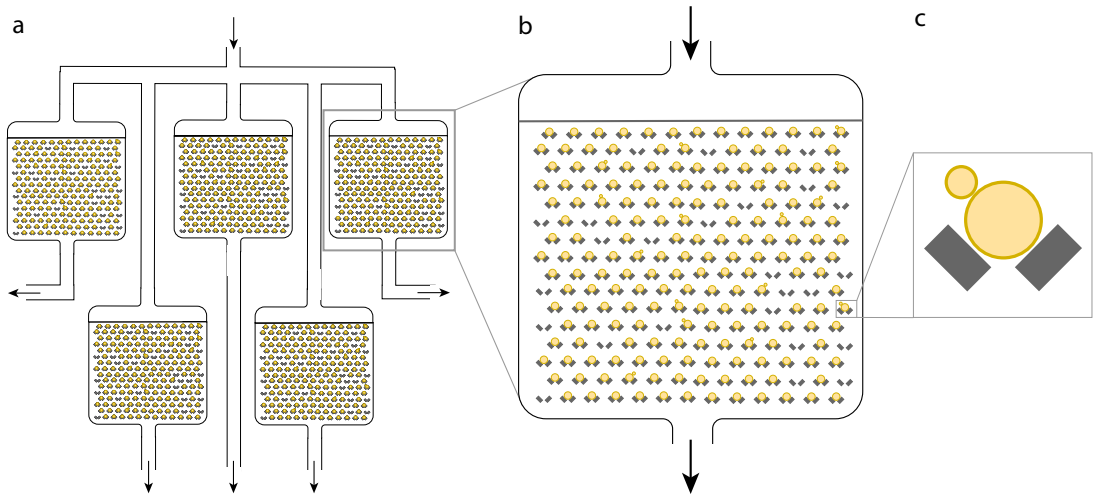


Figure 2.1: **ALCATRAS micro-fluidics device.**(a) Diagram of a full 5-chamber device, with media input and output. Arrows show the continuous flow of media. (b) Zoom into a single chamber. Cells are loaded into the chamber through the output and gathered at the wall at the top of the chamber. The flow of media is then reversed to settle cells into the traps. (c) Zoom into a single traps. The trap is made to hold a single cell throughout the entire experiment. Cells bud and divide, and daughter cells flow out of the device with waste media.

a Position over time: these need to be rectified in processing.

At a single position, we take multiple images at each time point. These include various channels (bright field, one or more fluorescent channels, and possible phase contrast), and several z-planes for each channel. We image multiple Positions per chamber (usually four to five): they are named according to strain and can be analysed in groups.

Each position can be further sub-divided into Traps (Figure 2.1c). A trap holds a single cell throughout the experiment. We can follow a cell as it buds and the daughter cell is born. Once separated, the daughter cell often flows out of the device with the media. This prevents the device from becoming over-crowded with cells, but it does bias the sample population towards older cells.

Although the traps are organised in a grid-like manner, their coordinates within a given field of view is not known. This is because the coordinates of a Position are related to stage movement, not to a location within the device. As such, the coordinates of the traps need to be obtained in processing.

Cells are loaded through the output tube of a chamber, with flow from output to input. They settle at a wall at the top of the chamber which prevents them from flowing up into the input. Once there are enough cells at the wall, we slowly reverse the flow of media and control the

cells' descent into the chamber so that they settle into as many traps as possible. After this, the flow of media is kept from input to output, continuously, such that the cells always have fresh media available.

2.1.2 The OMERO image database

Once acquired, the images are stored on an OMERO [3] server. The OMERO project was designed explicitly for storage of biological data, as such there is a good correspondence between OMERO objects and the hierarchy in the ALCATRAS device.

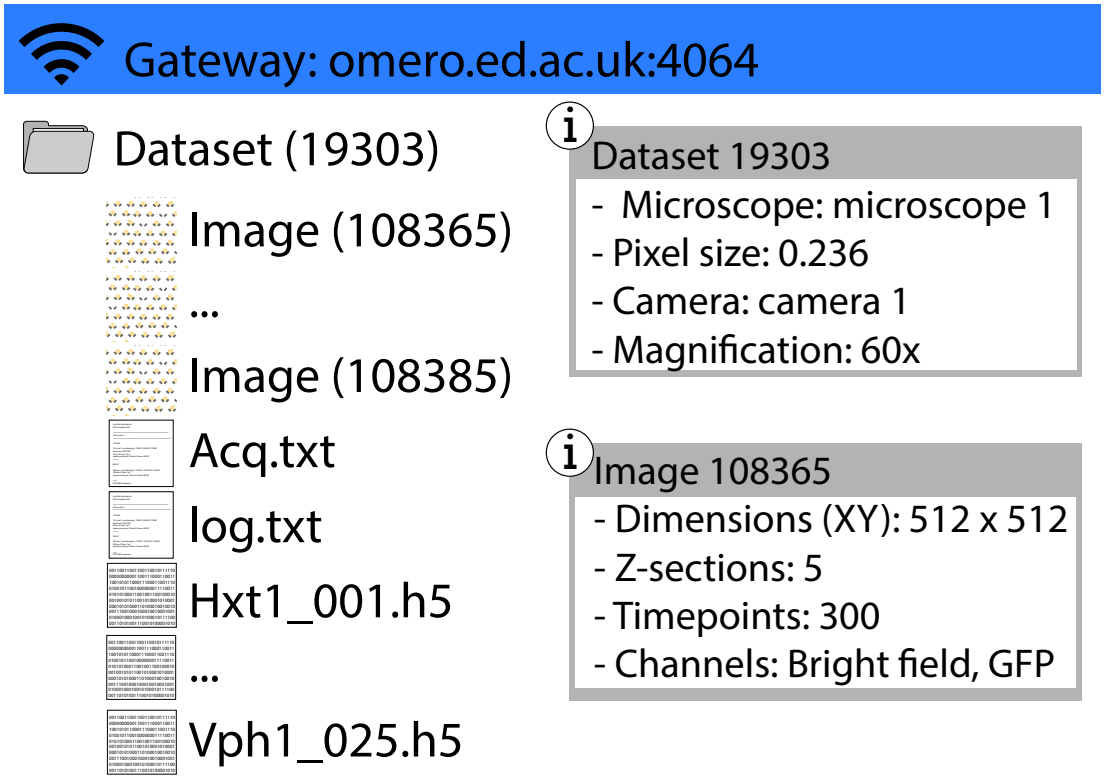


Figure 2.2: **The organisation of data on the OMERO server.** We ignore the Project level, and describe only the data with which we work for a single experiment. A single experiment is stored as an OMERO Dataset. Metadata associated with the dataset includes all of the acquisition parameters, and is mostly stored in text-based log files. Each position of the stage is stored as an OMERO Image. This object is directly queried for raw pixel data that makes up the images. Metadata associated an Image is stored either directly in the object, or as tags on OMERO. Additionally, HDF5 files corresponding to each Image hold the results of the pipeline.

Figure 2.2 shows a description of the files and metadata that correspond to a single experiment on OMERO.

OMERO objects are accessed through a connection gateway. Each object has a type and a unique ID, and can be requested from the gateway using this information. The highest-level

of abstraction on an OMERO server is a Project: this corresponds to the data for multiple experiments with a single focus. In the processing pipeline, we work one experiment at a time. We therefore ignore the project level.

An experiment is fully stored in an OMERO Dataset. This is essentially a folder that holds all the images, and file annotations of the experiment such as log files and analysis result files. Although most metadata is stored in log files, the Dataset object can also be tagged with metadata that can be directly queried from the server. We use these tags to filter through experiments when browsing OMERO: they include experimental information as seen in the info box in Figure 2.2, but also categorising information such as upload date, names of lab members, and strain and project descriptors. These tags are manually set by the experimenter upon upload.

In the automated pipeline, we assume that filtering has already been done and that the user can directly input the experiment ID. The OMERO server includes a searchable web interface for parsing through projects and tags in order to find the correct experiment. The images can even be browsed from the web interface to ensure that there are no issues before beginning the analysis pipeline.

From the Dataset object, we can obtain a list of unique IDs of all of the Images of an experiment. An Image on OMERO corresponds to a stage Position in the experiments: it is essentially a field of view of the microscope. It is five dimensional, holding all channels, time points, and z-stacks for a given field of view. This is the most used object in the pipeline, as it is what allows us to directly access to raw pixels. An OMERO Image is format-agnostic, meaning that the underlying data can be stored in many different image formats. In our case, the data is stored in an OMERO-specific Raw Pixel Data format. This has some speed advantages for slicing data across non-x,y dimensions.

Finally, additional (non-image) files can be stored as File Annotations on the OMERO server. The log files from image acquisition are stored at the same time as the images. After image processing and data extraction, we store the results in the form of HDF5 file annotations on OMERO as well. These can then be re-used for further analysis: this way we only run analysis once per experiment.

2.1.3 Contributions

The results presented in the rest of this chapter have been the work of many people.

- section 2.2: software by Diane-Yayra Adjavon, maintained by Diane-Yayra Adjavonand Alán F. Muñoz.
- section 2.3: software by Julian M.J. Pietsch, Alán F. Muñoz, and Diane-Yayra Adjavon
- section 2.4: the described methods were written by Diane-Yayra Adjavon, the surrounding package was written by Alán Muñoz
- section 2.5: merging and picking software by Alàn Muñoz, Growth rate estimation software by Peter S. Swain, adapted by Diane-Yayra Adjavonand Julian M.J. Pietsch

2.2 Automating image processing

2.2.1 Access to images

Image access is the start of the pipeline. Each Position consists of several gigabytes of data so downloading the entire Position into memory before beginning processing is not an option. I therefore opt for lazy loading of the data: pixel data is only loaded upon request. I leverage the Dask [98] `array` and `delayed` packages to do this. Dask arrays act similarly to NumPy [40] arrays, but combined with `delayed` can be made to lazy-load data upon request with very little extra effort for the user.

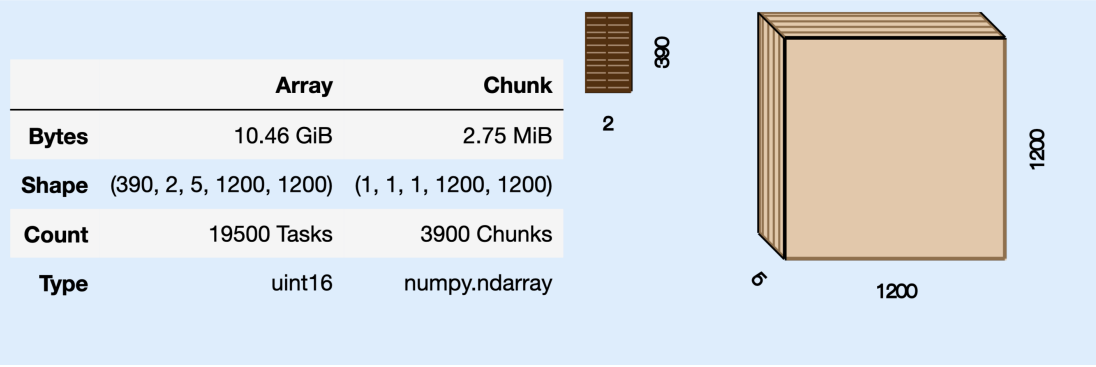


Figure 2.3: **Dask array corresponding to one Position/Image.** It is chunked into single x-y planes. The data is loaded, chunk by chunk, only when it is requested thanks to Dask's delayed function.

I create a Dask array for each OMERO Image, or experiment Position. Using the OMERO metadata, I define the shape of the array based on the number of time points, channels, z-positions, and the x-y size of the image. The Dask array is separated into chunks: atomic units that will be loaded at once. I set the chunk size of the arrays to be a single x-y plane, for which there is a standard request through the OMERO Image's API. I give Dask the instructions for how to retrieve any given chunk (based on its time, channel, and z indices). When Dask computes any part of the array, it loads only the selected chunks from OMERO.

An example of one of the Position arrays is given in Figure 2.3. It has 390 time points, two channels, five z-positions, and the x-y plane size is 1200 pixels by 1200 pixels. We can see that the full array is over 10GB of data, which is larger than we would want to store in memory. Each chunk corresponds to one time point, one channel, one z-position, and the full x-y plane. If we request time points 0 through 9, channel 0, and z-position 2, Dask will make only 10 requests to the OMERO Image object. Note that because the chunk is the full plane, the amount of data downloaded is the same if we request the full x-y plane or only part of it. Downloading the full field of view for a single time point is fast, but downloading a full time-lapse for a small portion of the field of view (e.g. a trap) is slow as a trade-off. This will affect the order of operations in the rest of the image processing. Namely, we will run processing in a time point by time point manner rather than a trap by trap manner to avoid double downloads. This order is also consistent with real-time processing of still-running experiments.

2.2.2 Texture-based trap segmentation

With image access implemented, the next step of the pipeline is to segment the field of view into individual traps. As stated previously, trap coordinates within a position are not known at imaging despite their grid-like organisation. Previously, we used a method of template-matching to find traps in the image. This required user input: the user needed to click on an empty trap in the image and this empty trap was then used as a template to find all of the others. Our new pipeline does not include a GUI, however, so providing a template for each position, or even each experiment, is cumbersome. On the other hand, a single template cannot be used across experiments due to significant difference in images. Even with the same device, microscope, magnification, and camera, using a trap template from a different experiment failed to robustly locate all of the traps in the experiments. I therefore needed to find traps by using only the information available within the image.

Luckily, the difference between traps/cells and background in the bright field images is stark. The background is smooth and homogeneous, whereas the cells and traps are textured. By measuring the entropy of the image, I segment textured areas from smooth areas.

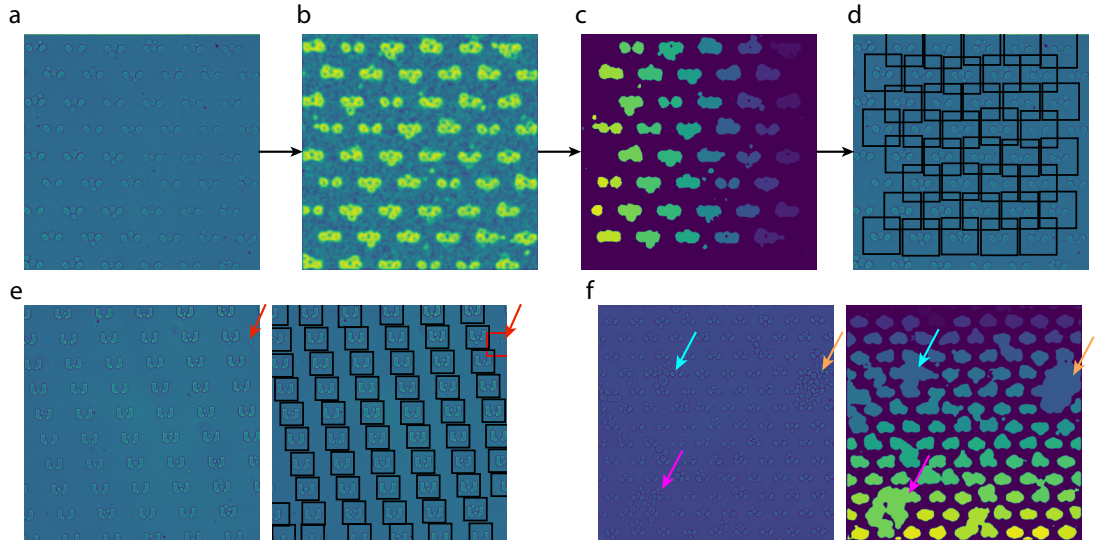


Figure 2.4: **Texture-based trap segmentation.**(a-d) Steps for standard segmentation. Start from a bright field image (a), measure the entropy (b), apply Otsu-thresholding and label connected regions (c), find the centre of large enough regions and assign traps with a fixed size (d). In (e) I show that this method picks up large contaminations as traps. If the device is over-crowded some areas cannot be split into individual traps (f).

In Figure 2.4 I show the steps of segmentation. Starting from a bright field image (Figure 2.4 a), I measure the entropy (Figure 2.4b). Entropy is an information theoretic measure that measures the level of randomness in some data. In an image, entropy is measured relative to a structuring element: in this case, I use a small circle whose radius is proportional to the total size of the image. For each pixel in the image, the entropy is measured over the histogram of values of all of its neighbouring pixels: these are the pixels within the circle. If the pixels within the circle have similar values, the entropy value is low. If there are many different values within the circle, then it is high. This allows us to distinguish between the homogeneous background (low entropy) and textured cells and traps (high entropy) in the image.

I then turn the entropy image into a binary mask using Otsu-thresholding. This method splits all of the pixels in the image into two classes, the background and the foreground, based on their brightness. Briefly, this is done by choosing a threshold that minimises intra-class variance (all pixels in the same class are similar). The pixels that are found to be background are set to 0, and those that are foreground are set to 1. This mask gives us regions of the image that contain cells and traps as 1-valued. I then label each connected region of the mask, to obtain the locations of individual traps (Figure 2.4c). Using the centre of these regions, I create uniform

(single-size) bounding boxes around each trap (Figure 2.4d).

This method has a few known issues. First, it will pick up any texture in the images, including contamination. Although the regions are filtered by size to get rid of smaller contamination areas, larger contaminations will slip through the cracks and be picked up as a trap. An example of this is shown in Figure 2.4e, where an arrow points towards the contamination in the bright field (left) and its corresponding "trap" (right). A possible mitigation would be to find the traps in the images at two different time points and only keep those that match across time. Contaminations should move due to the flow of media.

The second issue is one of overloaded devices. On rare occasions, cells accumulate in the device and are not washed out by media flow. In these cases, textured areas can encompass multiple traps, as in Figure 2.4f. This can be mitigated by filtering regions that are too large, but we risk losing data. Another option would be to filter out large regions, then recover the traps in those regions by adding an extra template-matching step using one of the found traps.

I use an image from the first time point to find traps, relying on image-registration to track traps across time points. Experimenters are able to see the state of their device before starting imaging. As such, they can control the state of the first time point. Devices usually only become over-loaded around the end of the experiment, after significant cell growth. Similarly, fields of view can be chosen to avoid significant contamination. I therefore do not implement these mitigations, relying instead on the experimenter to ensure the cleanliness of the images.

2.3 Birth Annotator for Budding Yeast

In order to run single-cell analysis from an experiment in an ALCATRAS device, a fundamental step is locating and segmenting the cells. The state-of-the-art of cell segmentation in our lab is the DISCO system [7]: a set of pre-determined filters applied to trap images to segment the inside of cells, as well as edges of cells.

These segmentations are not binary, rather they assign to each pixel a probability of being a cell interior (respectively, a cell edge). From the cell interior, the highest scoring pixel is used as a "seed", essentially representing the centre of the cell. It is used in conjunction with the edge image to create an cell outline.

The main issues with DISCO are its limited speed and its ability to locate small buds. The

image processing steps necessary to apply the DISCO filters are expensive, and need to be run for a large number of filters. As such, DISCO cannot be used for real-time analysis; its results cannot be used to inform the running experiment.

Second, buds under a certain size are invisible to the DISCO filters. An accurate measure of growth rate at each time point requires finding these small buds as soon as they appear. The appearance of a bud is also an accurate estimation of the irreversible START point in the cell cycle. We improved upon the accuracy and performance of the state-of-the-art using deep learning and a small amount of fast post-processing.

2.3.1 U-Net

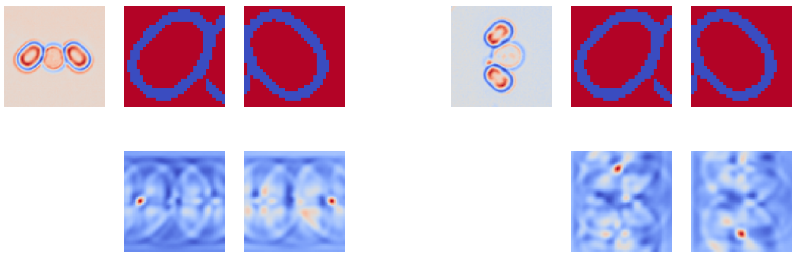
The basis for segmentation is a U-Net deep neural network. The U-Net architecture was originally created for biomedical segmentation [100], and has been successfully used in many segmentation tasks since [16, 26, 73, 88]. It is a convolutional neural network that consists of two parts, an encoder that reduces the input into many small features, and a decoder that upscales these features into an output.

Each step of the network consists of a convolutional block. The convolution element in this block is similar to DISCO filters: it has a kernel that corresponds to a specific pattern matching task. Unlike in DISCO, however, these kernels are automatically learned rather than manually curated. As such, we can stack far more of these filters together than were available in DISCO: this is the "deep" aspect of the neural network. The output of one block becomes the input of the next block, linking them all together in a network.

An intuitive example of how a convolution works is shown in Figure 2.5.

I manually curated a filter that represents a trap in the training image (Figure 2.5a) by cropping out the outline of the trap. We see in the convolution results that the output returns a bright spot in the position of the traps. Using the same filter on a previously unseen image, or testing set (Figure 2.5b), with the added difficulty of a rotation, we see that we once again recover the position of the traps. Note that this is actually a stroke of luck in this case as the traps look the same with any rotation that is a multiple of 90 degrees, but this should illustrate how powerful the method is.

Each convolutional block of the U-Net creates a new, larger set of features from its input. To



(a) Training set. (b) Testing set.

Figure 2.5: **Examples of convolutions.** (a) An image (top left) and two manually curated trap filters on the top. The bottom corresponds to convolution of the filters over the image. We see a bright red spot at the position of the trap in the image. (b)An image from a different experiment (top left), and the same filters as before. The results of the convolution are at the bottom, and still show bright spots at the position of the traps in the image.

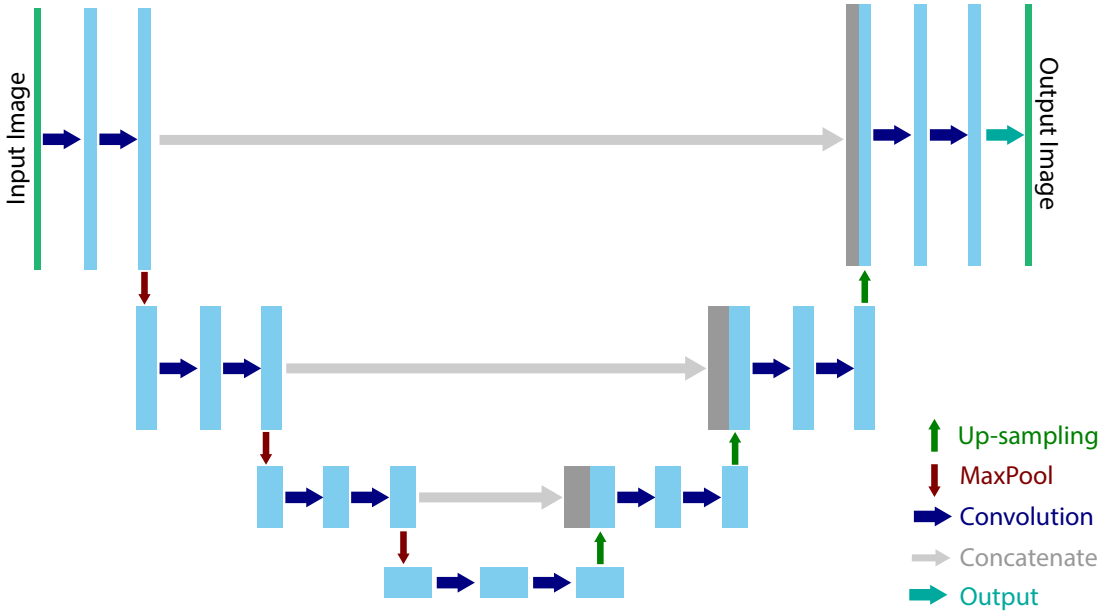


Figure 2.6: **The U-Net.** Figure from the original paper introducing the U-Net [100]. The depicted network is of depth 5, and consists of an encoder (left side) and a decoder (right side). Each depth level consists of three convolutions followed by non-linearities, then a max pooling (respectively up-convolution) to go down (respectively up) to the next level. The encoder and decoder are connected by skip layers, where the output of the encoder at a given depth is copied and concatenated to the input of the decoder at the same depth.

force the network to keep only small, relevant features, a down-sampling step is added after three convolutional blocks. This maximum pooling layer reduces the size of the features by half by replacing each 2 by 2 block of pixels by the maximum value within that block. We further include a downsampling step for the number of features used during training called drop-out. This step randomly removes a certain number of features at each pass through the network, forcing the network to return a reasonable output despite the missing information.

This improves generalisability, meaning it makes the network better at predictions on data that isn't in the training set.

The decoding side of the U-Net also consists of convolutional blocks, but it has an up-sampling step instead of the down-sampling. This up-sampling step is simply the inverse of the down-sampling step: each pixel is turned into a 2 by 2 block by simply repeating the value of the initial pixel.

Finally, the most characteristic part of the U-Net is its skip layers. As stated earlier, the encoder-decoder structure forces the network to focus on smaller, local features. In order not to lose information that is encoded in the global organisation of the pixels, we add larger-scale information to the decoder after each up-sampling step. This is done by concatenating the same-size layer of the encoder into the decoder layers, and using those as inputs for the next step of the decoder. The decoder is therefore able to create an output from both the local features that it upsampled from the bottleneck region, and global features that it obtains from the skip layers.

2.3.2 Semantically separable outputs

U-Net is normally used for semantic segmentation, that is separating cell from background/trap. We create an instance segmentation framework from this by creating a multi-output U-Net that improves the separability of cells. Instead of outputting a mask of cells (foreground) vs not-cell (background and traps), we curate a 3-part output separated by size.

The three output types are cell interior, cell outline, and bud neck. The cell outlines are manually curated using a special-made graphical user interface. Filled outlines create a mask of each cell. The cell interior output that we train the U-Net to retrieve is an eroded version of the cell mask: this reduces the chance of an overlap between targets. This type of output has been called seed in other studies [106]. Then, by looking at the intersection between mother and daughter cell outlines, we obtain the bud neck output prediction. This output will allow us to determine lineage relationship between cells.

Additionally, each of the output types is separated into three sizes: small, medium and large. Cells in micro-fluidics devices tend to overlap with each other when they are of different size: small (bud) cells overlapping with medium (daughter) cells and with larger (mother) cells. By separating by size, we improve detection of small buds, and can therefore catch buds much

earlier. This also improves lineage assignment.

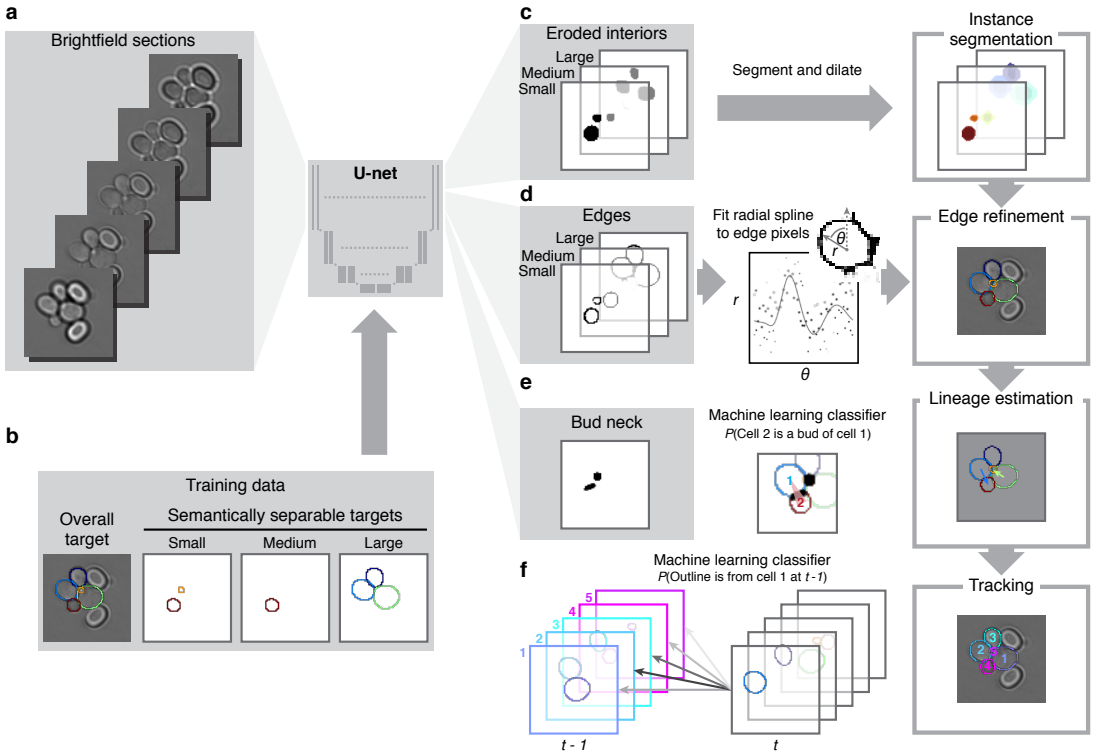


Figure 2.7: **Birth Annotator for Budding Yeast.** (a) The input is a z-stack of bright field image, centred around a trap. (b) Training data was manually annotated with outlines. The outlines are then separated into semantically separable targets, with size-thresholds hyper parameters chosen to minimise intersection across target sizes. (c - e) The U-Net outputs size-separated interiors, size-separated edges, and a bud neck prediction. The edges are directly compared to the ground truth labels, the interiors to a filled and then eroded version of these labels, and the bud neck to manually annotated bud neck labels (based on intersection between cell objects). (c) The eroded interiors are segmented into instances of cells. Duplicates at the border between sizes are removed based on Intersection-over-Union thresholds. (d) Edge predictions are used to refine the edges of the segmentations obtained from eroded interiors. (e) Bud necks and segmented cells are input into a classifier to determine mother-daughter relationship. (f) Another classifier tracks cell outline across time. Figure courtesy of J. Pietsch.

2.3.3 Instance segmentation

From the output of the U-Net, we are able to obtain instance segmentations of single cells with very little post-processing. The steps of post-processing include separation of instances (using watershed) within each size-group based on the cell interior or seed. We then cross-reference across groups to ensure that there are no duplicate cells. A duplicate is found if two cells in separate groups have a large enough overlap. In case of duplicates, we choose the cell with the largest segmentation.

The description of cells then needs to be translated into a radial spline. We get cell centres

from the obtained cell interior masks. We then dilate the centres to obtain the full cell mask, to get an estimate of the cell outline. From the centre, we project lines outwards at multiple angles: this gives us a set of angle-radius pairs to which to fit a radial spline. Using the cell outlines output by the U-Net, we refine the radial spline so that the fit spline has a maximal overlap with the outline mask obtained by the U-Net. Thus, we obtain the final description of the cell outline. We save both the spline representation (centre, angles, radii) and a sparse mask containing only the outline into the results store.

2.3.4 Lineage assignment and tracking

The algorithm above returns a set of cells for each snapshot of a trap, but does not yet link outlines from one time point to another into a time series for each cell. This cell tracking is done additionally using an XGBoost estimator. This model is given a set of features as input (cell centre, cell area, major and minor axis of the cell's ellipse representation, among others). It connects cells at time point t to cells at time point $t-1$ by comparing each $(t, t-1)$ pair of cells and outputting a probability that this pair corresponds to the same cell at two different time points. We then link the pair with the highest probability into a single-cell time-lapse. The output of this model is a cell label for each cell in each trap. We both store the cell label, and organise the segmentations by cell labels in the results store.

Lineage assignment works in a similar way, but uses the bud neck output of the U-Net as an input. It determines mother-daughter relationship for pairs of cells within a snapshot. This is refined at each time point so long as both cells are still available. Unlike the previous data for which a result needs to be stored for each time point, the mother-daughter relationship only needs to be stored once. We therefore store only the last, most refined estimation. It consists of an array of labels, where the index is the cell's unique label (1-indexed) and the value is the label of that cell's mother. A zero-value means that the mother is unknown.

2.4 Data extraction

Once the single-cell segmentations are available, we can begin the process of extraction. This process turns the images and accompanying segmentations into meaningful quantitative data. Namely, we extract brightness within the cell for each fluorescence channel, the volume of the

cell and bud separately, and their instantaneous growth rate. We also require some information about a full trap image in fluorescent channels, for example an estimation of the background fluorescence. This is the final step in the pipeline that requires access to the images.

Unlike the processes above, however, there isn't a standard work-flow for extraction. The fluorescent channels available are experiment-specific, and the choice of projection (max, mean, median) and of data types (max 5 pixels, mean fluorescence within the cell, proportion of fluorescence in different parts of the cell) depend on the questions asked as well. It is also the analysis step where there are most likely to be extensions. This step of processing is therefore done following a tree-like description of parameters. First the channel is chosen, then the projection, then the function. This structure is also followed for storage of the results.

In the following, I describe two examples of data extraction that will be used later on in this thesis: membrane fluorescence and cell volume.

2.4.1 Membrane Fluorescence

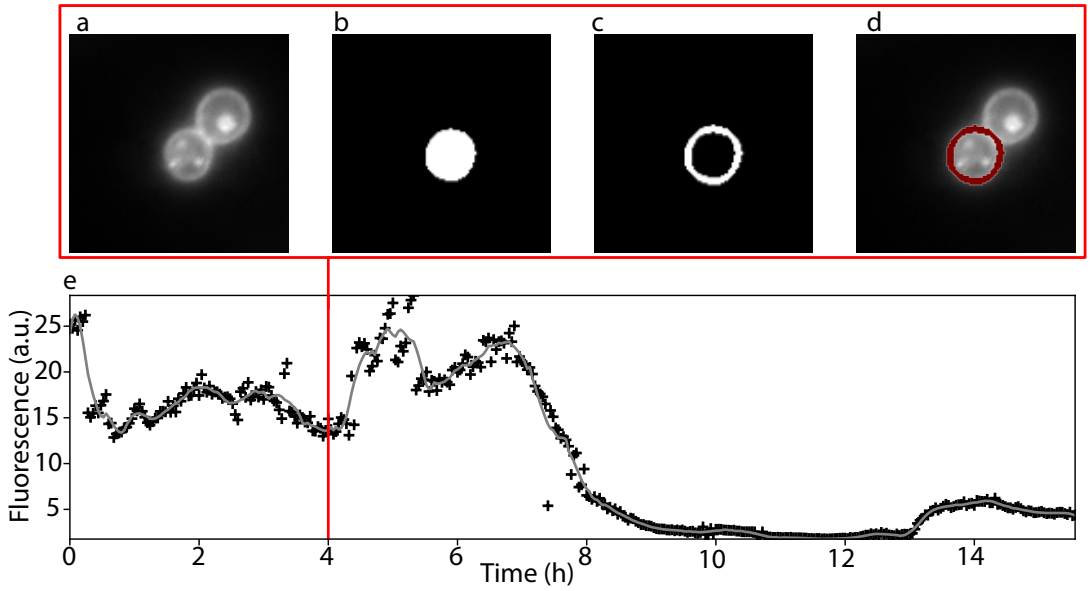


Figure 2.8: Membrane fluorescence extraction. (a) The maximum projection of a fluorescence channel (here, GFP). (b) A cell mask, obtained from BABY segmentation. (c) The membrane mask obtained from processing the cell mask. (d) An overly of the fluorescence channel and the mask, shown in red. (e) Membrane fluorescence for the cell shown in (a-d) across a 16h long experiment. The red line shows the time point at which the images in (a-d) taken. A Savitsky-Golay filter of the data points is shown in grey.

Membrane fluorescence is an extraction function that uses both an image and a cell mask. It is applied to a fluorescence channel, here GFP; the projection used is maximum projec-

tion. The function for fluorescence extraction therefore takes as input a 2D fluorescence image (Figure 2.8a) and a cell mask (Figure 2.8b).

We create a cell interior mask by eroding the cell mask 2 times, and an enlarged cell mask by double dilation of the cell mask. We remove the cell interior mask from the enlarged cell mask to obtain the membrane mask (Figure 2.8c). In Figure 2.8a we see that the fluorescent protein is localised preferentially on the membrane (bright ring) and in small vesicles (bright spots inside the cell. Figure 2.8d shows that the membrane mask covers the bright ring of the membrane in the fluorescent image. The membrane fluorescence is taken as the median of masked membrane pixels. Figure 2.8e shows the evolution of the membrane fluorescence over a 16h long experiment: we see that the fluorescent protein is progressively removed from the membrane after 7h (see chapter 3).

2.4.2 Volume

Some extraction functions only require the cell mask, for instance the estimation of area and volume. Area is straight-forward: it is the sum of pixels in the cell mask. In the following I describe the estimation of volume from the cell mask. We follow the conical method described in [35] to estimate cell volume from an outline in a way that is robust to various common cell shapes. The volume output of this function is in (anisotropic) voxels, and can be converted to cubic micrometers using metadata information on pixel sizes and distance between z-stacks.

Volume estimation requires a few assumptions. We assume that the cell is roughly ellipsoidal; this is in line with assumptions made by BABY in predicting the outline. We further assume that the mask that we get describes the central plane of the cell, the largest contour of the cell. For larger, mother cells, this is a reasonable assumption since these cells are constrained on either side by the micro-fluidics device. For smaller cell which can move in the z direction, a shift in their position can lead to errors in the volume estimation. For these smaller cells, we rely on BABY segmentation, which had access to multiple z-planes, to choose the largest available cell outline. Remaining noise in volume estimation will be smoothed out later on in analysis using Gaussian processes. Finally, we assume that the cell is rotationally symmetric along its major axis. This means that the cell is as high as it is wide.

We estimate the volume of the cell as four times the volume of an elliptical cone whose base is the cell mask. We get the height of the cone from the rotational symmetry assumption: the

height of the cone is the same as the length of the cell's minor axis. We estimate the length of the minor axis by finding the maximum distance of any pixel in the mask from the edge of the mask. The volume of the ellipse is then $\frac{4}{3}Bh$, where B is the base of the cone i.e. the area of the mask, and h is the height i.e. the length of the minor axis.

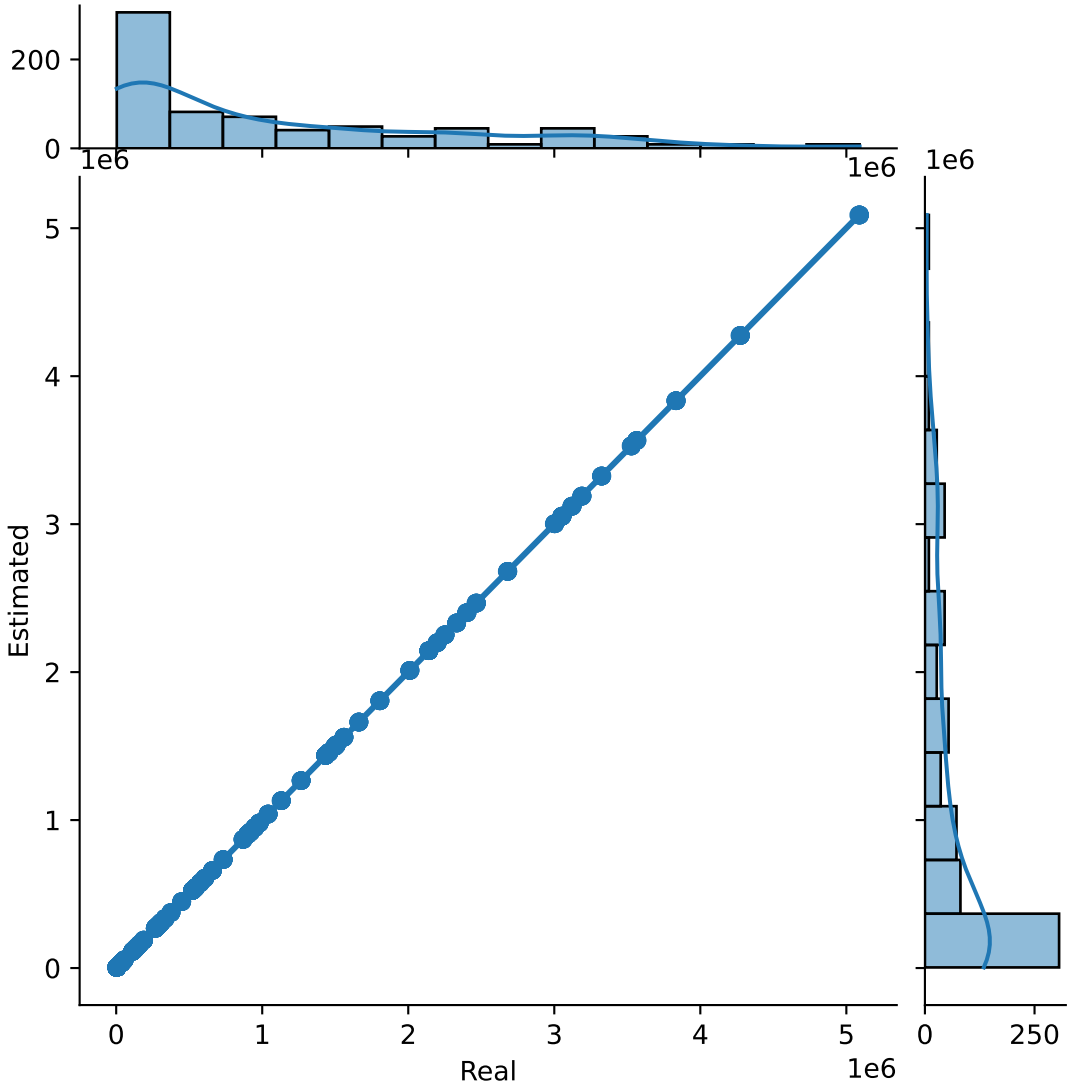


Figure 2.9: Analytical vs estimated volume of a synthetic set of elliptical masks with various sizes, eccentricities, and rotations.

I compared the estimations to the analytical volume for a synthetic set of elliptical masks. I varied the size of the ellipse (as determined by the length of the minor axis), its eccentricity (which, from the minor axis, gives the length of the major axis), and its rotation in the plane. There is perfect correlation between the two (Figure 2.9), with the estimated usually within 2% and always within 5% error of the analytical value. Nevertheless the time series of volume remains rather noisy because of the area and height both being in whole pixel units. The

volume is smoothed during post-processing.

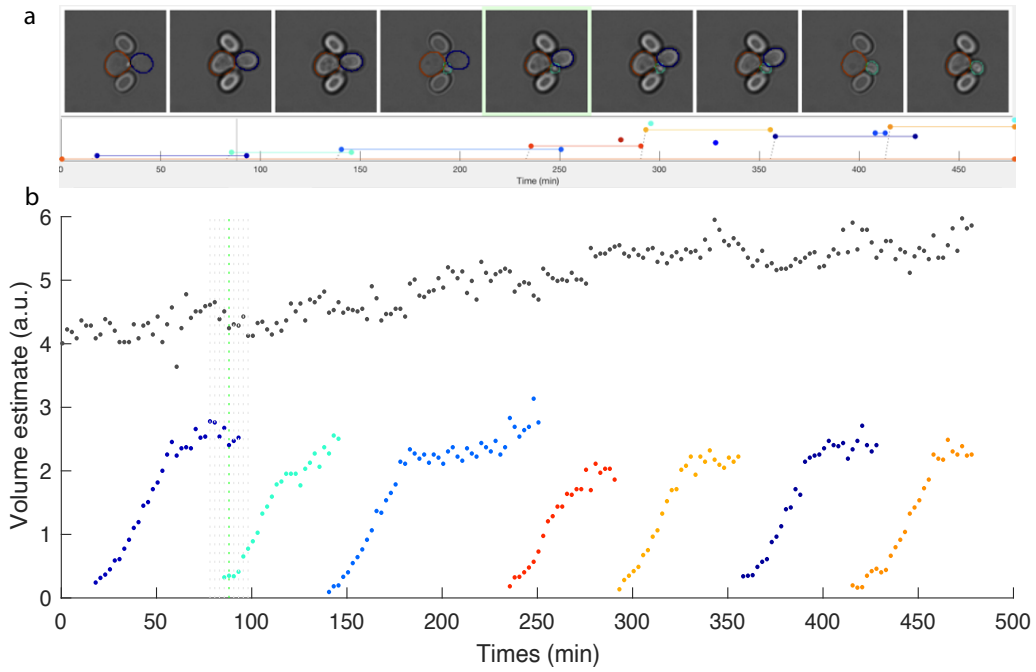


Figure 2.10: **Volume of a cell and its daughters.** (a)Bright field images of a trapped cell in a time lapse, taken from the Swain Lab's custom MATLAB Graphical User Interface, written by J. Pietsch. (b)The volume of the trapped mother cell (black) and its daughters (coloured) over time. The images in (a) correspond to the time period with the vertical dotted lines.

2.5 Post-Processing

We refer to post-processing as any modification of time series data obtained from or before extraction, that does not require access to the images nor to the segmentation masks. Post-processing can run separately from the rest of the pipeline, and is highly customisable. Nevertheless there are some post-processing functions that are a standard part of the pipeline: merging, picking, and growth rate estimation.

2.5.1 Merging and picking

After the processing described above, we are left with time series of extracted data for a set of cells. Merging and picking are filtering processes that determine which time series we consider for further analysis, and how.

Merging is introduced to fix any remaining BABY tracking errors. Using of of the extracted

signals (usually cell area), the merging process combines tracks that seem to correspond to the same cell. Tracks are merged if one starts soon after the other ends, and if the values of the merging signal at the connection point are similar. The thresholds for "soon" and "similar" are manually chosen.

Picking introduces further filtering to remove tracks that do not correspond to our expectations. This includes tracks that are too short, even after merging; and cells with no growth as determined by total change in volume. Custom filters can also be applied, for example to choose certain lineages.

2.5.2 Growth rate estimation

Instantaneous growth rate is both an indicator of fitness and a useful metric for observing the cell cycle. It is therefore a standard function during post-process and is applied to every experiment. We use two different methods for growth rate estimations, based on the speed requirements.

For long-term (stored) analysis, we estimate growth rates by fitting a Gaussian Process to the volume, following a method previously developed in our lab [117]. We set the bounds on the hyper-parameters to ensure that the Gaussian Process does not fit the noise in the data. From the fit, we are able to obtain a smoother volume time series with errors, as well as first and second derivatives of the volumes: the growth rate and change in growth rate, respectively. All of these time series come with a standard deviation, with the noise assumed to be Gaussian.

During prototyping and real-time processing where the growth rate may be used as an input for microscope control, fitting a Gaussian process is too slow. Instead, we fit a spline to the volume data using the Savitsky-Golay method. This method is much faster than Gaussian process fitting, but it is less reliable and does not include a confidence interval. It is reliable enough for coarse-grained control, such as determining whether cells have recovered from a stress condition by seeing if their growth rate has increased past a low threshold. We used fixed hyper-parameters for the Savitsky-Golay method, fitting a cubic spline with a small window so as not to overly smooth the signal.

2.6 Technical Implementation

2.6.1 Storage and size benchmarks

Large amounts of data are generated by each step of the pipeline. On the one hand, they are complementary to each other and should therefore be stored together for retrieval of results post-analysis. On the other hand the data is too large to fully read into memory and the different stages of analysis are rarely accessed at the same time. I therefore decided to store the data into Hierarchical Data Format (HDF5) files [61].

An HDF5 file is internally organised into groups and datasets. Data is stored in chunks in the file and can be compressed. A big advantage of this file format is that it can be partially read, using metadata to navigate through the structure and only reading one chunk at a time. With appropriately chosen data structures and chunk sizes, we can therefore obtain a very small file with fast read and write times.

The group and dataset structure of an HDF5 file is similar to a directory and file structure of most file systems. Each group is analogous to a directory: it is simply an organisational component and does not directly contain any data, although it can contain metadata in the form of attributes. Groups can be nested into a hierarchical structure. Datasets are analogous to files: they contain the actual data, although they can also have some metadata. Each dataset also has a fixed data type, which needs to be carefully chosen in order to conserve space. They are essentially large arrays of any number of dimensions. Datasets usually have a fixed size on all dimensions but one, to which data is appended. Variable-length data types (required in our case for describing cell outlines as radial splines) are available as well, but are costly in both storage space and metadata overhead.

We store the results in per-position HDF5 files, with each step of the pipeline in its own top-level group. The metadata overhead of nested data is high, so we store data by type instead of by cell. In order to organise it, we also store a set of identifiers in the same order as the other data. For example, for cell segmentation which is stored under the `cell info` group, we store the trap ID, the cell ID, and the time point for each data point. We can filter through the data using these IDs. The first steps of the pipeline, which are standardised, only have one group level. For steps which are not standardised and have the tree-like structure described above, a hierarchy of groups is used to describe the parameters and can be followed as they would be in

a tree.

2.6.2 Shape of datasets

During processing, new results are introduced time point by time point. Some results also have a second dimension that changes over time: the number of cells. There are two possible ways of storing this data. The first is appending results along one dimension only, and adding metadata to determine where to index into the array. For instance, when storing cell masks after segmentation, this method would create an three dimensional array where the first dimension is extensible to add new cells and new time points, and the two other dimensions correspond to the x-y dimensions of the trap. Additionally, we would create three one-dimensional indexing arrays — time point, trap, and cell label — to be able to correctly index into the segmentation mask array. At first glance this method seems efficient, because resizing is expensive in HDF5 and we are only resizing in one dimension; it also has a very simple implementation. We use this method for small data.

The second method creates an array with a slightly more intuitive shape: the first dimension corresponds to the cells, the second dimension to time, and the rest to the dimensions of the data (the x-y dimension of the segmentation mask, for example). A smaller additional indexing array would be required, that mapped each index in the cell dimension to a trap and cell label description. We call this the folded method. With this method, we need to resize in two dimensions: the time and the cell dimension, and the implementation requires more care. Unlike the linear method, we cannot simply append data at each pass, but rather need to find out where to add it, and whether to reshape on the cell dimensions, based on the indices. In the following, I describe benchmarks to justify why we store cell edge masks in this format rather than the linear format like other data.

I tested these two methods of storing this data, with benchmarks for reading speed, writing speed, and file size. First, I ran benchmarks on a small synthetic dataset, where I only stored a single array. I varied the number of cells and the number of time points (the first two dimensions) in the dataset. Counter-intuitively, the linear method was more expensive in both reading and writing time, as well as in file size (Figure 2.13). I found that this was because the operations in the linear method were much more sensitive to the amount of data, both the number of time points and the number of cells. More specifically, the timing of operations, as well as the file size, increased linearly with both cells and time points; the slope of increase

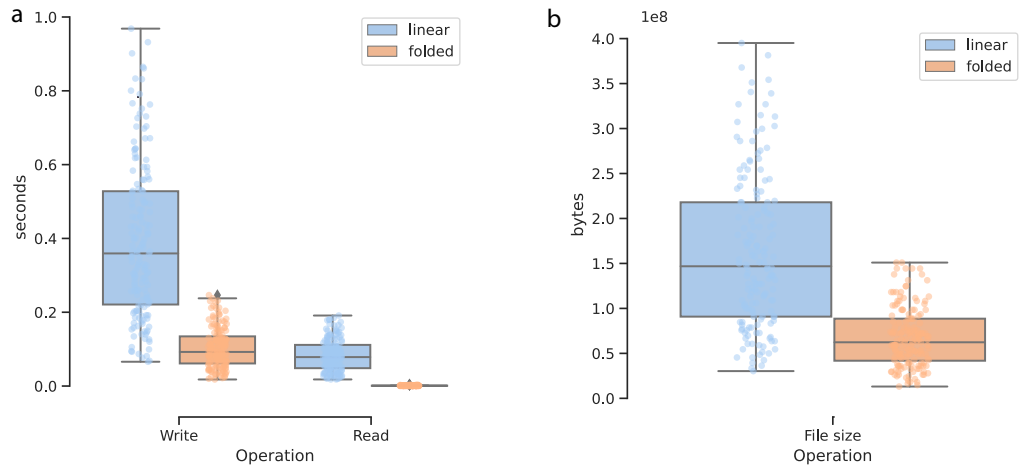


Figure 2.11: **Test storage benchmark.** Benchmarks on a synthetic dataset stored either flattened (linear) or as a 4-dimensional array (folded) with the first two dimensions corresponding to cell number and time, and the last two a fixed shaped example outline. (a) Shows the time taken by read and write operations, both faster in the folded method. (b) shows the corresponding file sizes after storage; once again the folded method is better in this test.

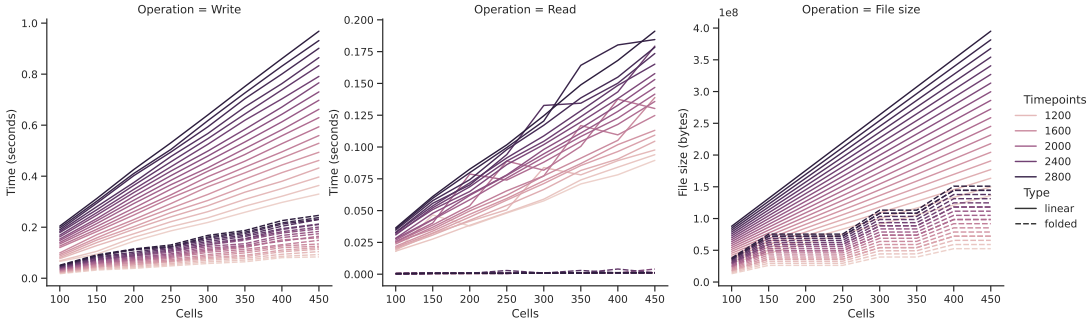
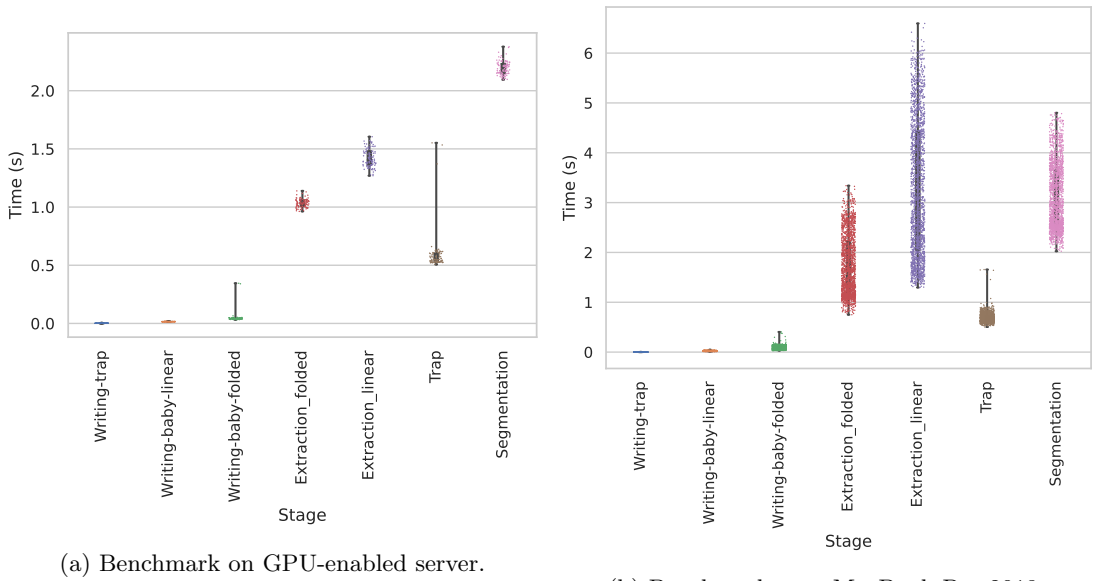


Figure 2.12: **Linearly increasing run time.** A benchmark of time time taken by operations based on the dimensions of the array, in particular the number of cells and the number of time points. These are the first two dimensions in the array in the folded method. Both read and write time, as well as file size, increase linearly with both number of cells (x-axis) and the number of time points (colour).The slope is much steeper in the linear method than in the folded method.

is much larger for the linear method (Figure 2.12). This is likely because both cells and time point increases happen along the same dimension in the linear method, meaning more data to traverse. When applied to a running pipeline, this would mean that processing would become significantly more expensive over time.

I next ran benchmarks on real data, processing five positions within one experiment. The writing operation of the unit test above corresponds to the BABY Writer, and the reading operation occurs during Extraction. The timing benchmarks are summarised in Figure 2.13. With this more complex data, the writing in the folded method is slightly longer and noisier than in the linear method, but both are negligible compared to the time taken by extraction.



(a) Benchmark on GPU-enabled server.

(b) Benchmark on a MacBook Pro 2018.

Figure 2.13: **Folding speeds up extraction.** Benchmarks on the effect of linear vs folded storage of cell outlines on real data. The steps of the pipeline that are affected are the Writing the results of the BABY segmentation (orange, green) and the extraction step (red, purple). The speed up in reading cell outlines is significant in extraction.

As expected, in the linear method the time taken by extraction increased over time, leading to a wide distribution of slow extraction times. Extraction becomes the bottleneck of the pipeline in the linear method because of this. By contrast, in the folded method extraction times are faster and more consistent; the bottleneck of the pipeline is the segmentation, as expected.

I also compared file sizes. Initially, the trade-off for this increased speed in the folded method was very large file sizes: about twice as large as in the linear methods. This was due to the way that the files were compressed. HDF5 compression works on chunks of data, rather than on full datasets, to enable partial reads and writes. When explicit chunk sizes are not given, the software attempts to guess a chunk size based on the shape of the data set. There is a time-storage trade-off in chunk sizes: smaller chunks are faster to decompress and read on request, but require more metadata information. I found that in the folded method, the automatic chunk size was small compared to the typical query of the extractor: individual masks were being divided into sub-parts, whereas they are always read in their entirety. I therefore modified the implementation to specify a chunk size of 25 cells, one time point, and the full mask dimension. This significantly reduced the file size to the numbers in Figure 2.14. Although the folded files are still larger than their linear counterparts, the overhead (mostly due to empty masks) is minimal. Moving forward, all complex data sets were stored using the folded method.

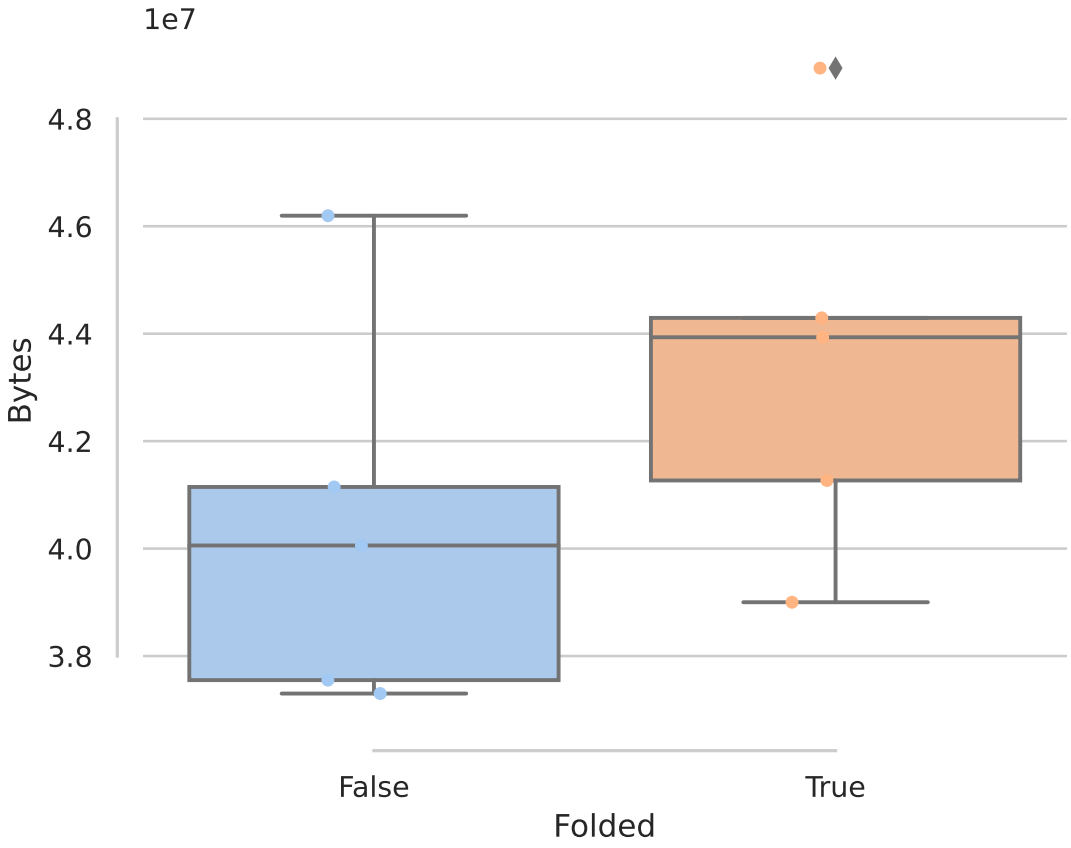


Figure 2.14: **Folded and linear data of similar size on disk.** File size benchmarks on real data. The size of the files where edge masks are stored with the folded method (orange) are larger than the files using linear storage, but they are of a similar order of magnitude (around 40MB).

2.6.3 Parallelisation and speed benchmarks

The positions of an experiment are completely independent from each other for all of the processing steps described above. We run processing on multiple positions in parallel to speed up processing. The speed-up obtained through parallelisation is strongly dependent on the hardware set-up: number of CPU cores, size of RAM, and GPU memory in particular. Nevertheless, we were able to obtain a 3 to 5 time speed-up using parallelisation as compared to sequential processing on the experiments used in the benchmarks above.

2.6.4 Abstraction and extension

One of the main goals of this pipeline is to standardise data access and extraction. Although the current steps of processing were previously well defined, there are likely to be additional steps included within the pipeline as the work of the lab evolves. New steps need to be able to use the existing data and functions, and follow the same logic. As such, we built the processing steps of the pipeline with a defined set of abstractions and rules. All extensions to the pipeline that follow the rules above should fit seamlessly.

First, every step that performs analysis and creates data needs to follow the basic structure of a Process. A Process is simply a class that takes in a set of Parameters (equivalent to a dictionary), and contains a `run` function. This function simply performs processing on a requested time point, for real-time processes, or list of time points, for post-processing.

Process classes are initialised with a Parameters dictionary. It wraps a regular python dictionary, but can also read from and write to a human-readable configuration file. This file describes the steps and sub-steps of processing, as well as their parameters, as nested bullet points in plain text. In order to fully understand the data in a results file, this configuration file is also necessary, and it should encapsulate all of the processing steps that were run.

Finally, we soft-enforce a separation of tasks across Processes. In particular, direct access to the raw images should be limited to the (existing) early steps of the pipeline. Access to the trap time-lapses should be done through the Tiler object. All analysis that requires images should occur before extraction. The images are the most memory intensive and difficult to access part of the pipeline, and with these rules we aim to reduce download time and number of copies. Additionally, there are special processes for access to HDF5 files. This ensures that the files

are suitably closed after access, and prevents an run-time error from corrupting the results file. Thanks to this, we can re-start processing after errors when they occur.

2.7 Discussion

Modern microscopy methods are creating large amounts of information-rich data that are impractical to manually annotate. As such, deep learning methods are commonly being used to automate annotation, with promising results [75]. There is a necessary step to developing this sort of method that is often overlooked, however, and this consists in creating the framework for accessing, processing, and storing the data in a way that is compatible with the deep learning frameworks used.

In our case, this meant creating a pipeline to programmatically access and pre-process images in python, replacing an existing MATLAB framework, including segmentation and post-processing into the pipeline, and providing the abstractions for this pipeline to be extensible in the future. It also included deciding upon a standardised storage format for the results of this pipeline to be stored in.

One may ask, why re-create a pipeline from scratch rather than introduce the new methods into the existing one?

The first reason to re-write was to create neater code. The MATLAB software used up until now in the lab was grown organically out of necessity by various lab members, most of which are no longer in the lab. As such, it contained a large amount of dead code, that is code that is never used. Refactoring out the dead code would have been difficult, however, as it was difficult to determine the dependency graph of the software without any coverage tests or explicit documentation. The re-write into Python couldn't use existing code, however it could use the MATLAB framework as a conceptual base. The existence of known use cases, as defined by how the previous software grew over time and is currently being used, made it easier to define what the new software should be able to do. The re-writing simply allowed us to re-think how to meet these specifications more efficiently.

The second main reason for a re-write was opening the software to wider audiences. Although it is commonly used in science, MATLAB has the disadvantage over Python of being proprietary software. This made MATLAB-based results more difficult to share. The storage of results into

MATLAB classes, written as .mat files, was particularly limiting. These files can only be read from MATLAB and requires the user to have access to the source code of the classes that were saved. Reading these files in other languages is cumbersome and unreliable, as demonstrated by our need to write a full translator. By converting to HDF5 storage with human-readable metadata and an openly defined documentation, we can now make results directly available with a single file (per position). HDF5 readers and writers have been implemented in most programming languages, MATLAB included. Anybody with the specification should be able to read our files, as well as create files of their own in that format which would seamlessly integrate into our pipeline.

These first two reasons amount to reducing technical debt [109].

Additionally, this switch to python with both real-time analysis and offline analysis planned opens the door to many new advances in image processing, deep learning, and microscopy visualisation. Micro-manager now has a python open-source Python equivalent [92] which could be used to replace the current MATLAB interface used in the lab. The MATLAB pipeline contains quality GUIs for data visualisation which are still missing in pipeline. The napari viewer [114] has objects that can be used to encapsulate much of the logic and hierarchy of our pipeline, and could be used as a base for the development of GUIs for our python pipeline, especially in combination with its Magic-GUI package. Napari has already been used to directly integrate deep learning tools, and there is an OMERO plugin in the making. By using python, we open access to a thriving community of bio-image analysis tool developers. This reduces the maintenance burden, increases opportunity for collaboration, and may give others quicker access to our tools.

I built the rest of the work described in this thesis around this pipeline. I will further describe in chapter 3 how simplified access to the trap images and segmentations allowed us to train a neural network for sub-cellular segmentation from bright field images. In chapter 4, I will show how the growth rate information obtained from this pipeline can be used to robustly predict cytokinesis, across various conditions and through transitions between media.

High-throughput microscopy methods are producing increasingly large amounts of data: up to 10GB of data in a routine experiment. Accessing this data, performing analysis on it, and parsing it into an interpretable form has become a technically difficult problem. Just as modern sequencing methods required novel computational tools, our framework contributes to the set of tools that are becoming essential to modern microscopy. Without such frameworks, the rich

potential of high-throughput microscopy may be wasted, with biological insights lost beneath the sheer quantity of data.

Chapter 3

Label-free sub-cellular segmentation

To establish the relative timing of protein localisation processes in response to carbon stress

3.1 Introduction

3.1.1 Limitations of fluorescence

Many of the processes occurring within a cell in response to a downshift in external glucose are inherently spatial. They take the form of a change in localisation of certain proteins within the cell. When studying these, the movement of proteins from one sub-cellular component to another is more informative than the activation or deactivation of their expression. The standard method for observing these protein localisation phenomena is using fluorescence microscopy. The protein of interest is tagged with a fluorescent marker, and usually the compartments that we expect the protein to occupy are tagged as well. By co-localisation measurements, we can

then quantify the proportion of the protein of interest inside, versus outside, of the tagged compartment.

There are multiple issues with this kind of experimental setup. To begin with, each additionally tagged protein requires modifications of the cell which are costly to produce. Furthermore, as fluorescence accumulates in the cells, so does the potential phototoxicity: this limits the number of different colours we can observe in cells without killing them. There is therefore a trade-off between the number of colours imaged and length or time-resolution of an experiment.

Additionally, there are technical limits to how many fluorescent proteins we can observe at the same time. The ideal fluorescent protein would only be excited by a single wavelength, and emit a single wavelength. We would then be able to stack an infinite number of fluorescent signals, given the right equipment. Instead, fluorescent proteins are excited by, and emit, a range of wavelengths with a peak at the ideal wavelength. As such, the spectra of two fluorescent proteins can partially overlap. This means that when we stack multiple fluorescent proteins with similar wavelengths, we have cross-talk in the signals which cannot be disentangled. Although the state-of-the-art allows the imaging of up to six different markers at once [4], in practice for long time-lapse experiments we are limited to two.

We are therefore limited in the number of proteins we can tag at once; when we tag organelle markers we reduce the spectral space and photo-toxicity space available for tagging proteins of interest. Often, we would like to observe the relative localisation dynamics of multiple proteins at once [36]. We would therefore like to remove the need for organelle markers without compromising on having quantitative, rather than qualitative, data.

Bright field images are available without much additional cost in most microscopy experiments. These are often used to infer the context of the fluorescent images that are taken of the same field of view. For instance, we use them in previously (Section section 2.3) to locate the cells, and determine the lineage relationships between cells. Convolutional neural networks allowed us to segment cells and bud-neck from bright field images. Similarly, these methods have been used to replace fluorescent channels with neural network predictions [21, 88].

In this chapter I trained two U-Net [100] neural networks to predict the location of the vacuole and the nucleus in budding yeast cells from bright field images. Using the predictions from the U-Net for nuclear segmentation, I am able to reliably quantify transcription factor localisation dynamics. I then use the U-Net for vacuolar segmentation to obtain single-cell quantifications

of the endocytosis of hexose transporters in response to a sudden downshift in glucose. I show that this dynamic, single-cell analysis provides extra information on cellular response to glucose starvation.

3.1.2 Contributions

- Data in section 3.3 was obtained by Julian M.J. Pietsch
- Data in section 3.4 was obtained by Ivan B.N. Clark
- Methods and results by Diane-Yayra Adjavon

3.2 Segmenting organelles from bright-field images

First, I use the pipeline described in chapter 2 to generate the training data. Unlike for BABY, this data does not need to be manually annotated as we have paired bright field and fluorescence images for both tasks.

3.2.1 Training data.

The training set for vacuole segmentation has bright field images with 5 z-stacks, and Vph1-GFP as a vacuolar marker. Vph1 is a subunit of the Vacuole ATPase complex, and is located on the vacuole membrane [138]. In order to compensate for the z-planes where the focus slices through the vacuole(s), I apply pre-processing to the fluorescent image to obtain the network's ground truth. First, I apply maximum projection to the 5 z-stacks of the fluorescent channel to obtain a single plane. Next, I correct for background fluorescence using a median filter. I then smooth the image with a Gaussian filter, and normalise it between 0 and 1, such that 0 corresponds to the second percentile value in the image, and 1 corresponds to the 98^{th} percentile. This removes outliers: arbitrary bright pixels which can be introduced during imaging [78]. Finally, I apply Otsu thresholding (see subsection 2.2.2) to obtain a binary mask.

The nuclear marker I use is Nhp6a, fused with red fluorescent protein mCherry. Nhp6a is localised inside the nucleus. It binds to the nucleosomes and is involved in chromatin organisation [115]. mCherry is generally less bright than GFP, and appears more noisy in measurements. I apply the same pre-processing routine to obtain masks from the fluorescence images as I did

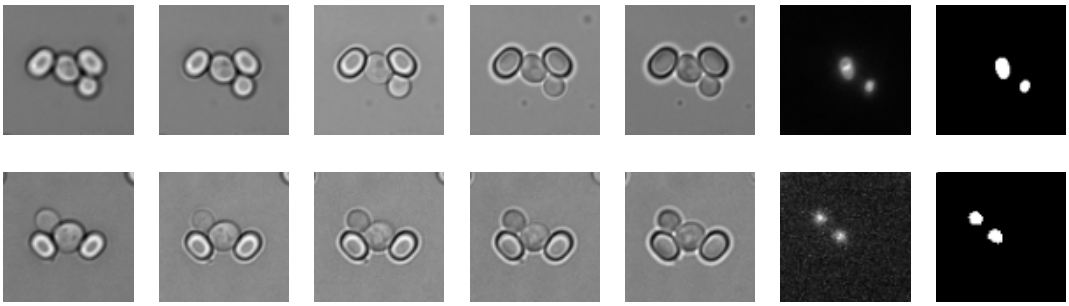


Figure 3.1: **Training data.** (a) Example training data for vacuole segmentation. From left to right, the 5 z-stacks of bright field used as input, the corresponding Vph1-GFP fluorescence image, and the Otsu-thresholded fluorescence image used as a target for the network. (b) Same as (a) but for nucleus segmentation. The fluorescence data is Nhp6a-mCherry.

for Vph1, but with hyper-parameters modified to fit the size of the nucleus, and the specificities of mCherry.

The output of the U-Net is, once again, a semantic segmentation. This means that it simply returns, for a full trap image, which pixels are predicted to be vacuole (respectively nucleus) and which are not; it does not separate instances. Unlike in BABY, the network does not need to be modified to get an instance segmentation. I simply use the output of BABY to mask the output of the vacuole-prediction U-Net during post-processing. This does not exactly correspond to vacuole instance segmentation, as there can be multiple vacuoles in the cell at once. This does not affect the analysis, however, as I am only interested in per-cell information.

3.2.2 Augmentation

Deep learning models learn best with a large training data set. As I have limited training data, I artificially increase the size of the training data using a processed called augmentation. Augmentation consists of a set of modifications made to the training images to increase the diversity of the training set and better represent the unknown data on which the network will be evaluated. On each training example, I apply a randomly chosen set of augmentations shown in Figure 3.2.

The first set of augmentations are random flips and rotations that are a multiple of 90 degrees (Figure 3.2b). This represents the various directions that the micro-fluidics device can be put onto the stage. In particular it helps the network learn to locate and remove the traps from the images. Additionally, I simulate translational and rotational drift of the stage by adding

(a) Original images.

(b) Horizontal flip.

(c) Vertical shift.

(d) Rotation.

(e) Noise.

(f) Elastic deformation.

Figure 3.2: **Augmentation operations on training data.** For each augmentation I show one bright field z-stack and the Max-projection of GFP, as well as the final mask that will be given as a target to the U-Net.

randomly chosen rotations (Figure 3.2d) and small translations of up to 10 pixels(Figure 3.2c).

Finally I introduce noise (Figure 3.2e) slight elastic deformations (Figure 3.2f) in the image. This essentially generates a new cell from the current cell image. It needs to be used sparingly, because it also introduces deformations in the background and trap pixels. As such, I apply a lower probability of to this augmentation than to the others in the list.

All z-stacks of the bright field input, as well as the fluorescence image are augmented in the

same manner. The images are then normalised (for the bright field input) and pre-processed into a mask (for the fluorescence image).

3.2.3 Training

In both cases, the network was written in Tensorflow using Keras [1, 20]. I use the Adam [58] optimiser with a fixed learning rate of 0.001 and a beta value of 0.9. The training set is split into training (70%) and validation (30%). Training is run for a maximum of 100 epochs, with a batch size of 16 and running through the full training set (with augmentations) at each epoch. I include an Early Stopping checkpoint in the training loop, with a patience of 10. This checkpoint stops the training if the validation loss does not improve over 10 epochs, helping to avoid over-fitting. It also reduces training time. As such, training rarely continues for a full 100 epochs: the network is very quick to train.

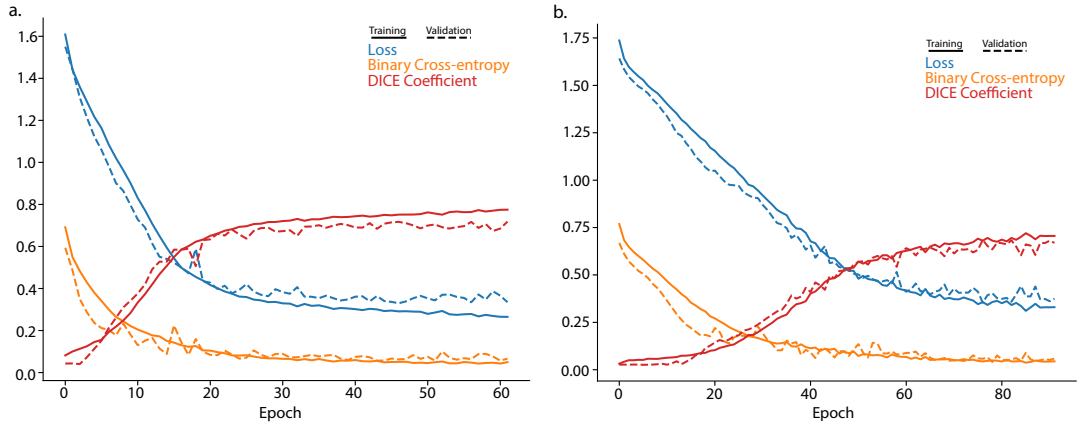


Figure 3.3: **Training loss and metrics for sub-cellular segmentation.** (a) Loss and metrics for vacuole segmentation. Training (filled) and validation (dashed) loss, binary cross-entropy, Dice coefficient, and Dice loss over training epochs. The loss is a combination of binary cross-entropy and Dice loss, each with an equal weight. The binary cross entropy is dominant during a first phase of training, where the network learns to roughly recognise the location of the cell. The Dice coefficient is $1 - d_{loss}$ where d_{loss} is the Dice loss. It increases when the dice loss decreases. It is dominant during a second phase of training where the network refines its outline of the vacuole. (b) Same as (a), but for nucleus segmentation.

The loss function is a combination of binary cross-entropy and DICE loss, with equal weights. Binary-cross entropy is a measure of the difference between two probability distributions, the true t and predicted p:

$$L_{BCE}(t, p) = -(tlog(p) + (1 - t)log(1 - p)) \tag{3.1}$$

It is a classification loss, and works on each individual pixel in the image. The loss used in

training takes the average binary cross-entropy over all of the pixels in the image. It has nice gradients, but is disadvantaged when classes are imbalanced [50]. This is the case with our data, where the organelle is much smaller than the background.

I balance it with a loss based on the Dice coefficient. The Dice coefficient works on the full true and predicted images, and measures their similarity [50]. It is computed from positive pixels, or pixels in the binary images that are set to 1. These pixels fall under three categories:

- True Positives TP: pixels that are positive in both the prediction and the true image
- False Positives FP: pixels that are positive in the prediction, but not in the true image
- False Negatives FN: pixels that are positive in the true image, but not in the prediction

From these, I can compute the Dice coefficient D.

$$D = \frac{2TP}{2TP + FP + FN}$$
$$L_{DICE} = 1 - D \tag{3.2}$$

It is essentially a measure of how many positive pixels the prediction got right as a fraction of all of the positive pixels. The Dice coefficient increases as the predictions improve: the number of true positives TP increases, whereas the number of false positives FP and false negatives FN decreases. If the prediction is perfect, then the Dice coefficient reaches 1. As the stochastic gradient descent minimises rather than maximises the loss function, I turn the Dice coefficient into a usable loss L_{Dice} as seen in Equation 3.2.

Binary cross entropy dominates at the beginning of training, where the network seems to learn low-frequency, global features. There is then a second phase of training where the Dice loss dominates. At this point, the network is fine-tuning the high-frequency features around the approximate location of the target organelle. If the first phase of learning on Binary cross entropy goes on too long, the network seems to learn either to segment the whole cell (for the vacuole) or a blank output (for the nucleus). These are local minima for the binary cross entropy that are easy to reach and difficult to get out of. The use of a pre-processing function (to learn masks, rather than actual fluorescence values), along with the addition of the Dice loss, steers the network away from that local minimum.

In addition to the loss, I evaluate a set of metrics to determine which kinds of mistakes the network is making. Of particular interest are Precision and Recall. Recall measures the network's

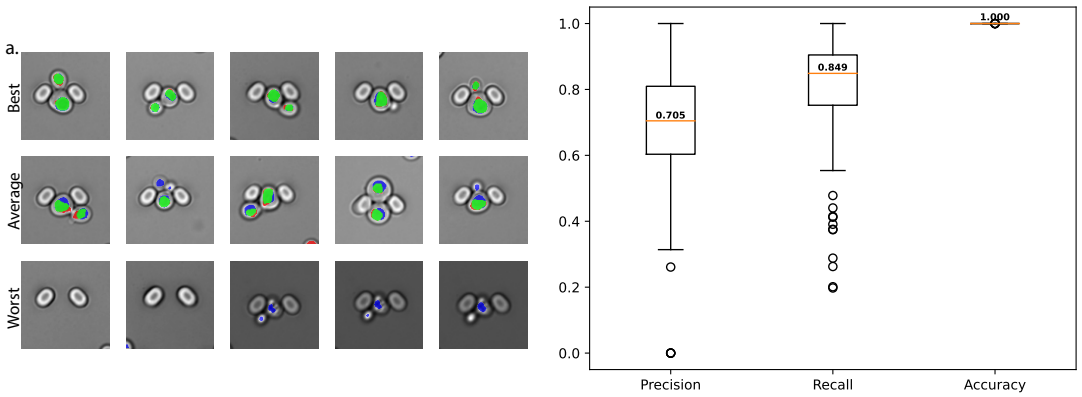


Figure 3.4: **Performance of vacuole segmentation.** (a) Best, average, and worst examples of segmentation plotted as an overlay. The mask is green on correctly predicted pixels, blue on false positives, and red on false negatives. (b) Precision and recall of the network across the testing set.

ability to find all of the targets in the image: of all of the target pixels, how many are retrieved in the prediction. A high recall means that the network is not missing any of the organelles. Precision, on the other hand, measures the network's ability to fine-tune: of all of the predicted pixels, how many are were actually in the target. A high precision means the network is not over-estimating the size of the organelles. The Dice loss ensures a balance between the two. I report these metrics, along with a few examples of predictions made by the networks, for the vacuole segmentation (Figure 3.4) and the nucleus segmentation(Figure 3.5). I plot some samples based on their performance with regards to loss, with best performing on top, average in the middle, and worst on the bottom. This sheds light on the kinds of problems the network may be having. In particular: the errors made by the vacuole network in the worst case are predictable. It does not do well on empty traps, or sickly cells (worst). It overestimates the size of the vacuole (average), likely due to the pre-processing function. It gives buds vacuoles too early (average); this is easily remediated by only using the mother cells in processing. The errors made by the nucleus network are less predictable. There is no clear issue with the images where the network performs worse.

In the next sections, I evaluate the ability of these networks to produce usable predictions for quantifying the localisation dynamics of fluorescent proteins.

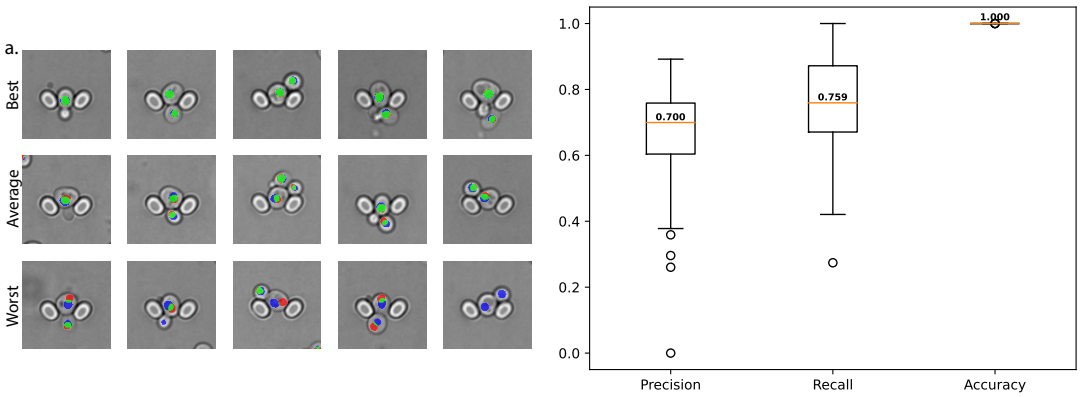


Figure 3.5: **Performance of nucleus segmentation.** (a) Best, average, and worst examples of segmentation plotted as an overlay. The mask is green on correctly predicted pixels, blue on false positives, and red on false negatives. (b) Precision and recall of the network across the testing set.

Id	GFP	mCherry	Nuclear Marker	Data set
1	Dot6	Nhp6a	Yes	Training, testing
2	Tod6	Nhp6a	Yes	Testing
3	Sfp1	Nhp6a	Yes	Testing
4	Sfp1	Msn2	No	Analysis

Table 3.1: **Strains for transcription factor localisation.**

3.3 Transcription factor localisation dynamics

3.3.1 Background

Tens of transcription factors localise to the nucleus in response to glucose starvation [11]. These transcription factors go on to regulate various parts of the cell's stress response. The dynamics of their localisation holds information on the type of stress and induces different strategies for response [36, 69, 132]. I used the predicted nucleus masks to quantify the localisation of several of these transcription factors.

Msn2 is a general factor of stress response, and is though to regulate up to 150 genes [25]. Dot6 and Tod6 are paralogs, both involved in rRNA and ribosome biogenesis [46]. Sfp1 is also involved in regulating ribosomal biogenesis, however it is usually localised in the nucleus, and de-localises during stress conditions. These transcription factors are considered generalists: they encode for multiple kinds of stress [36].

The strains used in this chapter are described in Table 3.1. They are used in one of three ways:

- Training: one position with this strain is used as a training set. This position is not used in the remaining analysis.
- Testing: for strains with a nuclear marker, I compare the predicted localisation dynamics with those obtained from co-localisation of the transcription factor and the nuclear marker
- Analysis: these strains do not have a nuclear marker. I verify that the predicted dynamics match what is expected from literature.

3.3.2 Analysis in glucose depletion.

I evaluated the nucleus segmentation network on an existing glucose depletion experiment (data by Julian M.J Pietsch [36]). The experiment contains a shift in glucose concentration in the media. The cells begin growth in 2% glucose and then are shifted into 0.1% glucose media. Cells are able to recover to almost the same growth rate after a switch to 0.1% glucose from 2% glucose, after a short adaptation period. The switch does, however, induce a stress response in the cells, which I measure as the localisation of certain transcription factors to and from the nucleus.

I trained the network on one field of view coming from the Dot6/Nhp6a strain, ignoring the Dot6-GFP channel. I verify the nucleus prediction network by quantifying transcription factor localisation dynamics of the remaining fields of view in the Dot6 chamber, as well as the cells from all other strains. In this experiment, doubly-tagged strains are available: I are therefore able to directly compare the predicted nuclear localisation curves with the same ones obtained through conventional co-localisation methods.

The strains imaged in the experiment are summarized in Table 3.1; the results are shown in Figure 3.6.

I estimated the fluorescence within the nucleus using either Nhp6a or the network's prediction to determine the location of the nucleus. For each time point, I obtain the fluorescence z-stack, and turn it into a 2-dimensional image by maximum projection. This image represents the location of the transcription factor. I then multiply the nuclear mask with the fluorescence image to obtain the nuclear fluorescence; all of the pixels outside of the nucleus are set to 0. This is reduced to a single value of nuclear fluorescence by taking the sum of the fluorescence across all of the remaining pixels.

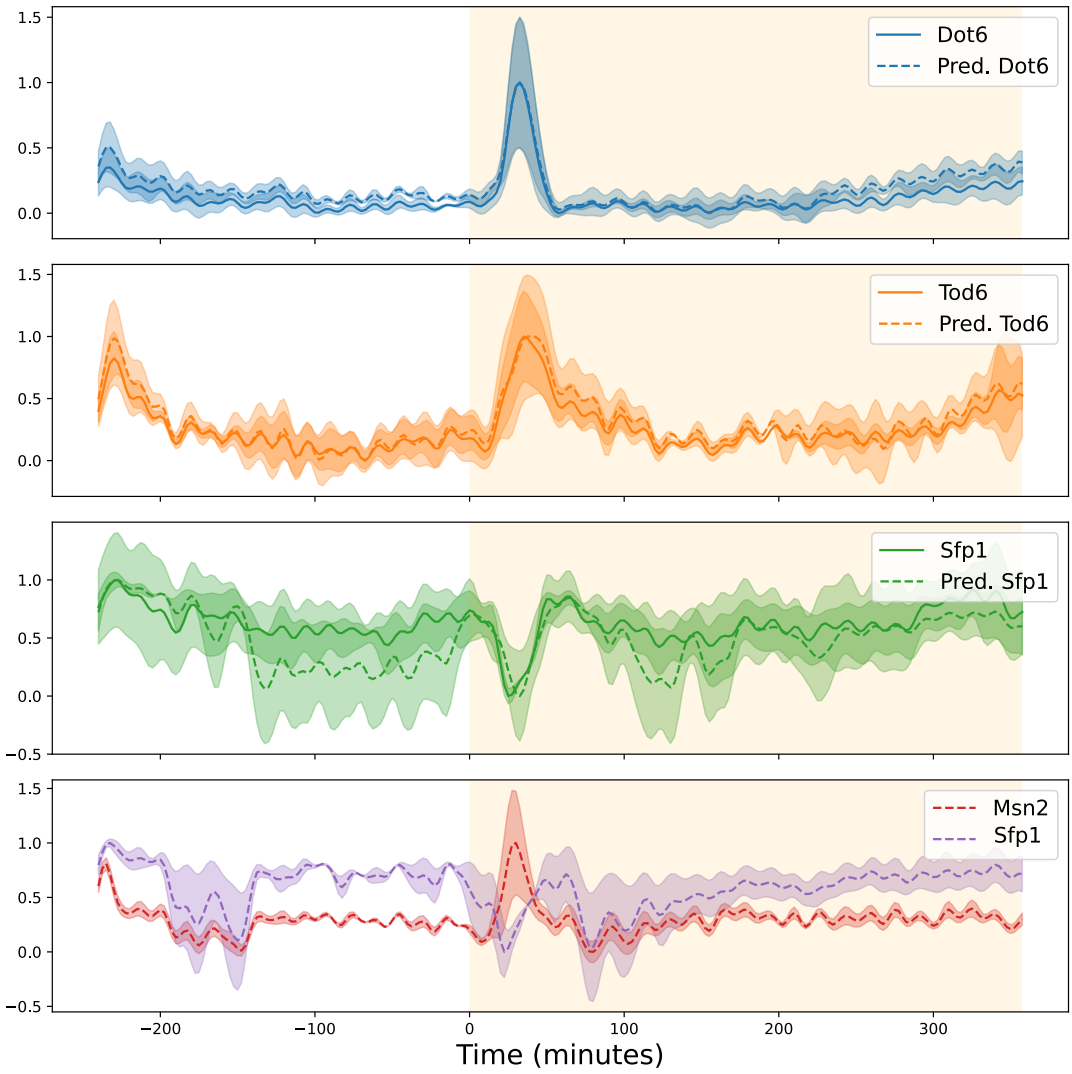


Figure 3.6: **Nuclear localisation of transcription factors.** The mean fluorescence in the nucleus of three transcription factors, evaluated user the nuclear marker Nhp6a (filled) or the bright field nucleus segmentation (dotted). Cells were grown in 2% glucose, then shifted to 0.1% glucose in SC. Tod6 and Dot6 localise transiently to the nucleus after the shift to 0.1% glucose. Sfp1, in contrast, de-localises. The predicted nucleus returns localisation results similar to the nuclear marker.

Figure 3.6 shows the mean and standard deviation of the population data after de-trending, smoothing, and normalisation. I de-trend the mean to remove a general upwards trend. This is done by obtaining a linear fit, then subtracting it from the data. I then smooth the result by replacing each point with a weighted average of its neighbourhood, with the neighbourhood defined by a bell-shaped, Hanning window. This is done by convolution. Finally, I run a min-max normalisation (Equation 3.3) to obtain similar values across all transcription factors.

$$\mathbf{y_{norm}} = \frac{\mathbf{y} - y_{min}}{y_{max} - y_{min}} \tag{3.3}$$

The dotted lines show the results using the GFP channel and the predicted nucleus, and the filled lines show the results using Nhp6a to define the nuclear mask. The background colour in the plot corresponds to the glucose concentration, with 2% glucose in white and 0.1% glucose in yellow. For Dot6 and Tod6 the predicted curves are almost identical to the ground truth. Most importantly, they display a transient up tick in nuclear localisation just after the switch.

Although I retrieve the downshift in nuclear Sfp1 just after switch, there are several other points during the experiment where the network predicts a de-localisation which is not visible using Nhp6a data. The performance of the network could account for this. If the precision of the network is low, then the Sfp1 signal is more difficult to quantify: the network needs to correctly predict the position of the nucleus throughout a majority of the experiment for Sfp1 whereas for Tod6 and Dot6 it is especially during the switch that the network needs to be correct, as the signal is diffused the rest of the time and small mistakes when the transcription factor is cytosolic has a limited effect on the result.

Since I trained the network using data from Dot6, I wanted to ensure that the results were not strain-dependent. I therefore re-ran an evaluation of the network, this time adding Tod6 and Sfp1 data to the testing set and separating the testing set by strain. In theory, the tagged transcription factor should not affect the network since that channel is not used at all. I evaluated the Dice coefficient for all cells in the testing data. The results are in Figure 3.7.

Surprisingly, the network's performance is indeed strain dependent. Namely, it performs better on the Dot6/Nhp6a and the Tod6/Nhp6a strains than it does on the Sfp1/Nhp6a. Thus, the errors made in the quantification of the nuclear localisation could directly be related to the performance of the network. The reason behind this difference in performance is unclear. The morphology of the cell which the network uses to predict the position of the nucleus should be consistent across cells that are identical except for a fluorescent protein. The difference between the strains would have to have an effect on the morphology of the cells that is visible in bright field images, or the tagged transcription factor would have to have an effect on the brightness or localisation of the Nhp6a-mCherry reporter. In some sense, this result is more telling about the strains than about the method of prediction. It seems likely that tagging some proteins induces a stronger burden of fitness of the cell. In this case, becoming aware of the differences between the strains is of general use as these must be taken into account in the interpretation of any results.

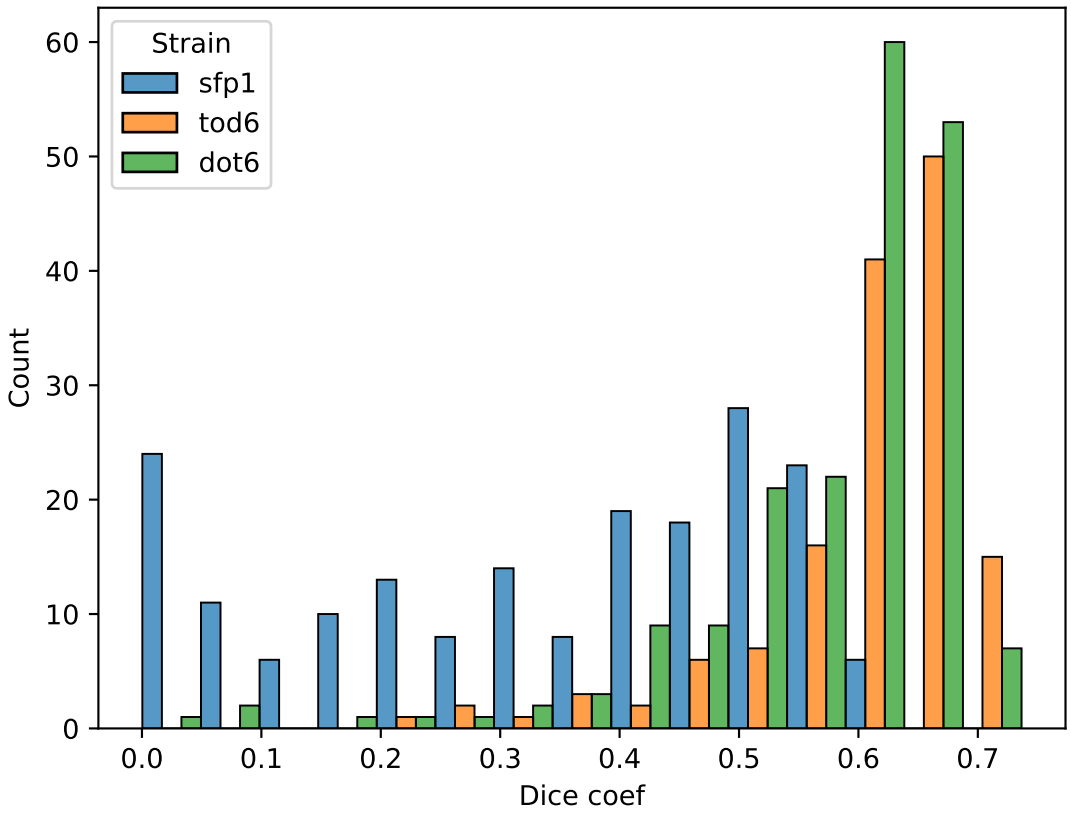


Figure 3.7: **Per-strain DICE-score for nucleus prediction.** The network was trained on data from a Dot6/Nhp6a strain. The DICE-score for Dot6/Nhp6a and Tod6/Nhp6a in the testing data are similar and peak at around 0.65. The scores for the Sfp1/Nhp6a in the testing data are more widely spread and peak closer to 0.5. This suggests that there is a physiological difference in the Sfp1/Nhp6a strains that is visible in the bright field images.

3.3.3 Timing of transcription factor localisation.

Despite the above issues, the data in Figure 3.6 can be used to determine how fast the cells respond to the glucose downshift. I obtain the time of the maximum nuclear fluorescence of Dot6, Tod6, and Msn2, and the minimum for Sfp1 in both available strains: I define this as the response time. The results are shown in Table 3.2

Overall, we see that the cells can begin their response to glucose stress within minutes of a change in environment. In the next section I use the vacuole segmentation network to obtain the timing of a phenomenon that occurs towards the end of the stress response.

Strain	Signal	Response time (min)
1	Dot6	13.0
1	Pred. Dot6	13.0
2	Tod6	14.0
2	Pred. Tod6	15.0
3	Sfp1	10.0
3	Pred. Sfp1	13.0
4	Msn2	12.0
4	Sfp1	9.0

Table 3.2: **Response time of transcription factors.** Response time, peak or dip in nuclear fluorescence, in minutes relative to the switch from 2% to 0.1% glucose. Strains are described in Table 3.1, and the signals correspond to labels in Figure 3.6. All of the transcription factors respond within 20 minutes of the stress being introduced.

3.4 Membrane protein trafficking

3.4.1 Background

Transporter	Affinity	Optimal condition [76]	Alpha-arrestins
Hxt1	Low	High glucose concentrations	Rod1/Art4 [85]
Hxt2	High	Medium-to-high glucose concentrations; glucose down-shifts	Csr2/Art8 [45]
Hxt3	Low	High glucose concentrations	Rod1/Art4; Rog3/Art7 [85]
Hxt4	High	Appears in glucose down-shifts	Csr2/Art8 [45]
Hxt5	Medium	Low glucose conditions, including no glucose	None
Hxt6	High	Low glucose concentrations	Csr2/Art8 [45] and Rod1/Art4 [72]
Hxt7	High	Low glucose concentrations	Csr2/Art8 [45]

Table 3.3: **Budding yeast hexose transporters.** Along with conditions in which they appear, based on a glucose hat experiment [76]. Cells were growth in sugarless SC media, then moved to 1% glucose, then shifted back to sugarless. Additionally, I show arrestins which have been shown to interact with the transporter.

Glucose intake in budding yeasts is primarily done by hexose transporters Hxt1-Hxt7. The affinities of these transporters to glucose span the full range from low ($K_m \approx 1mM$) to high ($K_m \approx 100mM$). Their expression and localisation to the membrane is regulated for optimal glucose uptake in the current condition [76]. In particular, when a transporter at the membrane is no longer optimal it may undergo endocytosis and degradation in the vacuole. Hexose transporters are tagged for endocytosis by E3-ubiquitin ligase Rsp5 in conjunction with an alpha-arrestin. These arrestins are expressed and activated according to the glucose concentrations, mostly regulated by Snf1. The hexose transporters, their affinities, and their interaction with arrestins are described in Table 3.3.

Strain	2% Glucose (phase 1)	0% glucose	2% glucose (phase 2)
Hxt1-GFP	On the membrane; some in the vacuole after long times	Endocytosis and degradation in the vacuole	New expression going to the membrane
Hxt3-GFP	On the membrane; some in the vacuole after long times	Endocytosis and degradation in the vacuole	New expression going to the membrane
Hxt6-GFP	Vacuolar or absent	New expression membranal; some endocytosis after long times	Endocytosis and degradation in the vacuole
Vph1-GFP	Vacuolar	Vacuolar	Vacuolar

Table 3.4: **Hexose transporter strains and expected dynamics.** A description of my expectations for hexose transporter fluorescence based on literature, as a comparison to the results in Figure 3.8. The Vph1-GFP strain is used as a control.

In literature interactions between the hexose transporters, arrestins, Rsp5, and regulating pathways are usually determined by snapshots in time. These data do not give an indication of the time taken for these processes to occur. We know from section 3.3 that the cell can sense a change in glucose concentration within minutes. In the following, I determine whether endocytosis occurs on the same time-scale.

3.4.2 Analysis in glucose depletion

I analyse data from an experiment where cells were grown in ALCATRAS micro-fluidics devices in three phases (experiments run by Ivan Clark). During the first phase they are grown in 2% glucose in synthetic complete medium for six hours. The cells are starved for six hours by switching to a 0% glucose environment. Though their starvation response is triggered at this point, they are still able to grow slowly on the nutrients available in synthetic complete medium. After 6 hours in starvation, glucose (2%) is re-introduced into the medium.

We know from previous work [76] that based on their affinity for glucose, different transporters are optimal in different concentrations of glucose. In 2% glucose, the low affinity transporters Hxt1 and Hxt3 should dominate, but they need to be replaced by higher affinity transporters, here Hxt6, when moved to low glucose conditions. It is expected that as Hxt6 is expressed and moved to the membrane, Hxt1 and Hxt3 should be moved to the vacuole for degradation. The strains used and their expected dynamics are summarised in Table 3.4

For this analysis, I compare the fraction of total fluorescence available on the membrane against the fraction of fluorescence in the vacuole. I obtain membrane fluorescence using the method

described in chapter 2, subsection 2.4.1. I use the vacuole segmentation network as a mask to obtain the fluorescence in the vacuole. As previously (subsection 3.3.2), the quantification of fluorescence in the vacuole is obtained by multiplying the maximum projection of fluorescence by the vacuolar mask to get only the vacuolar pixels, then taking the sum. I obtain the total fluorescence in the same way, using the cell mask instead of the vacuole mask.

Because of background membrane fluorescence, the predicted vacuolar fluorescence is never zero. Furthermore, its global dynamics will resemble those of the membrane/whole cell. This is why I use the fraction of total fluorescence, rather than the absolute value of fluorescence in the vacuole, to determine whether the protein is predominately vacuolar or membranal.

Unlike in subsection 3.3.2, there are no strains with both a hexose transporter and the vacuolar marker, Vph1, tagged. As such, I can not compare the predicted time series to a ground truth. Instead, I use the Vph1-GFP strain as a control: it should always be vacuolar. This is the same data used to verify the performance of the vacuole segmentation network in subsection 3.2.3.

The results of the analysis are summarised in the Figure 3.8. For each protein, the top panel shows the fraction of total fluorescence in the membrane (blue) and in the vacuole (red), while the bottom panel shows the total fluorescence. The background colour of the plot represents the media in which cells are growing, with 2% glucose SC in white and 0% glucose SC in grey.

I first verify that Vph1 is indeed vacuolar throughout the experiment, and its total fluorescence is relatively stable. It undergoes a small decrease during the 0% glucose phase which could be due to the cell having limited resources at that time. The noise levels indicate that the predictions that the network makes are prone to error in each single cell.

The two low affinity transporters, Hxt1 and Hxt3, have the expected dynamics. In the first phase of high glucose, they are predominantly found in the membrane and are kept at a stable concentration. After the switch to 0% glucose there is a lag, after which the transporters are moved to the vacuole, where we can see from the total fluorescence plot that they are being degraded.

The high affinity transporter Hxt6 does not have the expected endocytosis dynamics, however. Throughout the experiment, it is consistently found on the membrane, although its total fluorescence does exhibit the expected increase during the 0% glucose phase and subsequent decrease when glucose is added back into the media. Interestingly, the decrease of Hxt6 does not have the same trend as the decrease of both Hxt1 and Hxt3. Hxt6 decreases linearly in

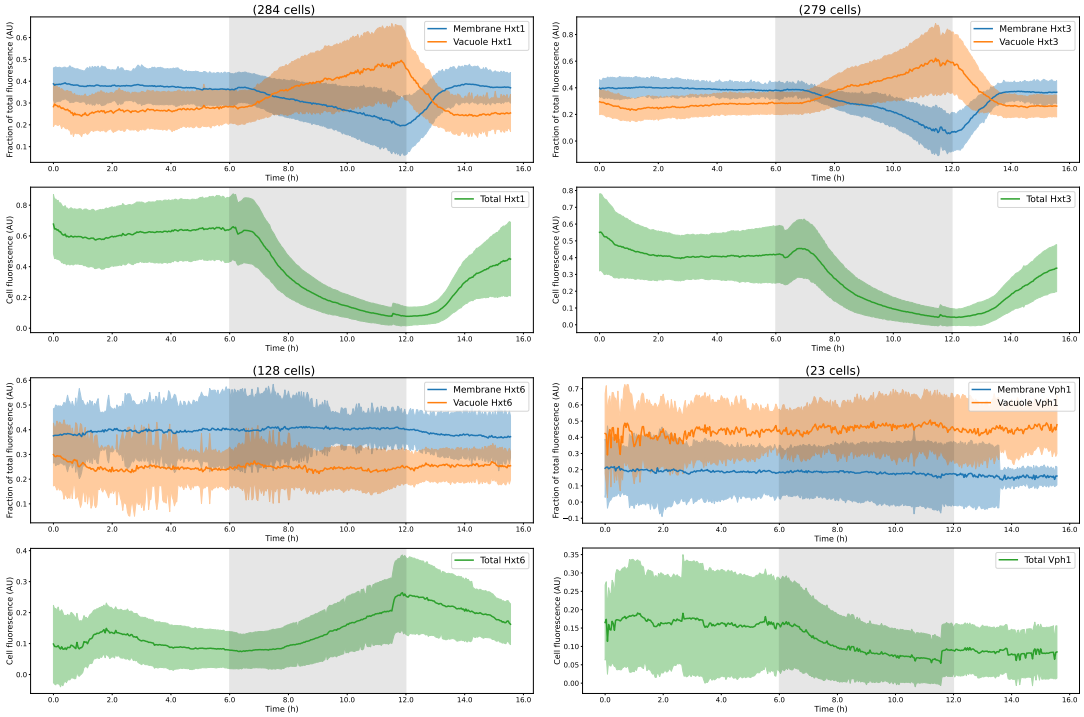


Figure 3.8: **Hexose transporter endocytosis results.** Cells are grown in 2% glucose in SC for 6 hours, then 0% glucose SC for another 6h, then moved back to 2% glucose. Using the predicted vacuole location, as well as the membrane outline described in chapter 2, I measure the fraction of total fluorescence on the membrane (blue) and in the vacuole (red). Additionally, I plot the total fluorescence in the cell (green). (a) Hxt1 fluorescence is stable and membranal in the first 2% glucose phase. After the switch to 0% glucose, it becomes vacuolar and is gradually degraded after a lag. When glucose is returned to the medium, there is once again an increase in fluorescence on the membrane after another short lag. (b) Hxt3 fluorescence has similar dynamics to Hxt1, it is membranal in 2% glucose, vacuolar and degraded in 0% and returns to membranal after glucose is returned to the media. (c) Hxt6 unexpectedly stays primarily membranal throughout the experiment. Its absolute fluorescence does, however increase after glucose is removed from the medium. It also slightly decreases after glucose is returned again. (d) I use Vph1 as a control: it remains vacuolar throughout the entire experiment, as expected.

time, while Hxt1/3 decrease exponentially: this is consistent with the hypothesis that Hxt1/3 are actively degraded in the vacuole, while Hxt6 is not. These findings for Hxt6 contrast with previous studies [45].

3.4.3 Delay in endocytosis

There is a noticeable, stable delay between the switch into glucose starvation, and the low affinity transporter endocytosis. Based on the population data in Figure 3.8, this delay is consistent across the population of cells. I verified this by measuring it in a per-cell manner. For each cell, I defined the response time as the delay between the shift into 0% glucose, and

the first point at which the fluorescence becomes mostly vacuolar:

$$\frac{vacuolar}{membranal} > 1.1 \tag{3.4}$$

This can be seen in population data as the point where the membrane and vacuole fractions cross. The results can be seen in Figure 3.9.

The endocytosis response time takes, on average, 1.54h for Hxt1, and 1.93h for Hxt3. Interestingly, this delay is much longer than the transcription factor response time defined in subsection 3.3.2. There is, therefore, an additional delay between the initiation of the stress response and the endocytosis of hexose transporters. This standard delay suggests that there is an active process occurring during that time.

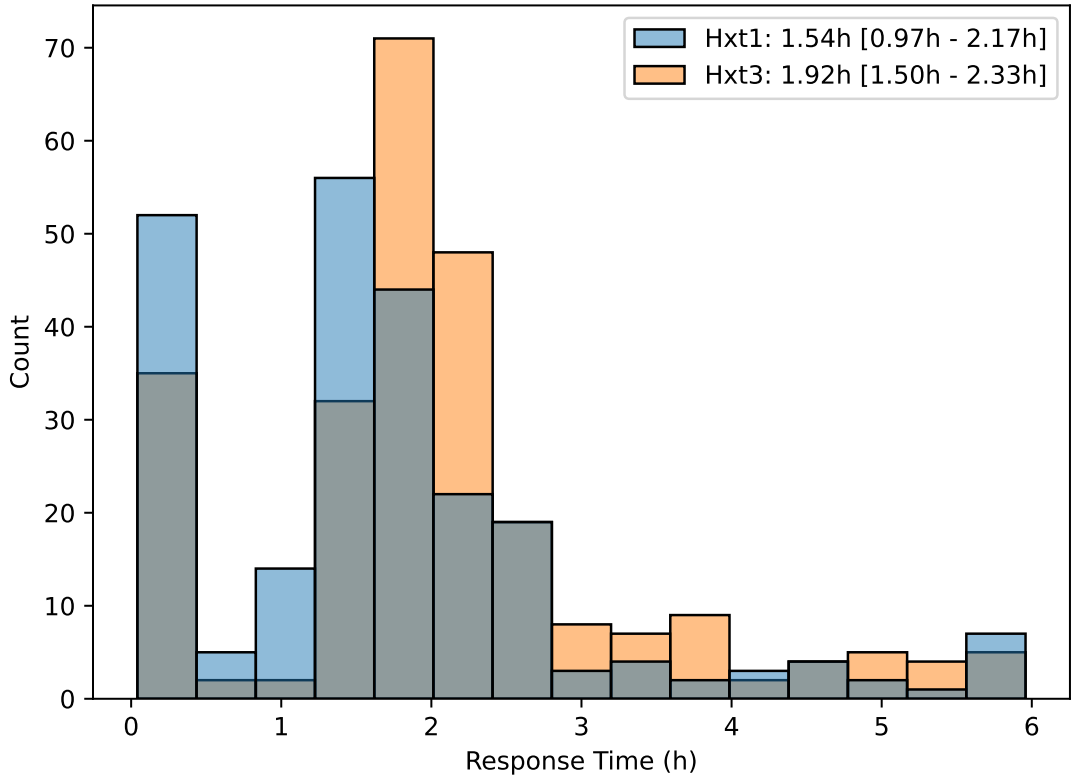


Figure 3.9: **Lag before endocytosis of low affinity transporters.** The response time of low affinity transporter endocytosis after switch from 2% glucose in SC to 0% glucose in SC. Response time is defined as the amount of time taken for the fluorescence to become primarily vacuolar after the switch to 0% glucose. Response times for Hxt1 (respectively Hxt3) are shown in blue (respectively orange), with a median time of around 90min (respectively 2h). Interquartile ranges of response times are show in brackets as [25%, 75%].

3.5 Conclusion

3.5.1 Label-free segmentation of organelles from bright field.

I trained U-Net to segment sub-cellular compartments from bright field images of budding yeast cells. Using these networks, we can now quantify protein localisation dynamics without the need for localisation markers. The predictions are accurate enough to use to quantify the transient changes in localisation of various proteins in response to a change in external glucose concentration, namely the endocytosis of some glucose transporters in response to starvation, and the transient nuclear localisation of transcription factors as a response to a downshift in glucose.

The trained networks can be used to augment microscopy experiments, but they the same network cannot be easily applied to any experiment. In particular for the nucleus prediction network, the results are very dependent upon the training data. I was not able to create a network that generalises across experiments without sacrificing performance. As such, in order to be usable the network needs to have been trained on data from the same experiment. When combined with a multi-chamber ALCATRAS micro-fluidics device, however, this can still yield a powerful tool. By putting strains with the desired markers in some chambers, and the strains with the proteins of interest in other chambers of the device, we generate the training data and the data that we wish to analyse at the same time. Since the networks can be trained on little data, and very quickly, re-training to get optimal performance comes a little cost. Using transfer learning could improve the set-up even further.

Future work on this project should focus on improving generalisability and including additional compartments. In this work I used data from existing experiments or experiments that were run to observe specific phenomena. Creating a curated dataset of a variety of morphological markers could improve the generalisability of the network, allowing us to avoid re-training for each new experiment. Additionally it would be interesting to compare the performance on various markers for the same organelle, including with different colours. Although I have used a pre-processed fluorescence image as the ground truth in all of the evaluations, there are additional considerations that can make some markers better for training than others. Vph1, for example, is localised on the membrane of the vacuole rather than inside the vacuole. A comparison of Vph1 against another marker would help define how much of the prediction error might be due to pre-processing introducing artefacts.

3.5.2 The relative timing of stress response processes.

Using the trained networks, I sketched the relative timing of events in the cell's stress response. I found that although the transcription factor localisation is a very fast process [36], there is a significant lag before we see the change in membrane composition (hexose transporters) that accompanies it. This lag is not a result of a large population variation, but rather that individual cells have a waiting time of around 2h on average before any significant shift from plasma membrane to vacuole is detected. By comparison, the transcription factor localisation is maximal less than 20 minutes after the stress has been administered.

The reasons behind such a long delay remains an open question. One hypothesis is that the set-up for endocytosis could take up to this time. The expression of the necessary alpha-arrestin, followed by its activation, its binding with Rsp5 and to the transporter as described in subsection 3.4.1 could be a lengthy process. Additionally, the time between ubiquitylation of the transporter and endocytosis needs to be taken into account. We know that ubiquitylation involves the protein Rsp5 binding the hexose transporters at the membrane. A co-localisation experiment with Rsp5 and Hxt1/3 both tagged would determine an intermediate time point: where the arrestin is expressed and activated, by endocytosis has not yet occurred.

The second possible explanation is a bet-hedging one. Especially in the situation where cells are moved from a very high glucose media to a media without any glucose, it could be disadvantageous for them to invest resources in adapting to a new, sub-optimal condition. In this case, the cells would wait a short time to ensure that the new media is permanent before making any adaptations. If during that waiting period the cells were returned to a more favourable condition, they would immediately be able to grow quickly. This would suggest a memory of past conditions in the membrane reorganisation strategy, as exists for transitions between fermentation and respiration [15]. This hypothesis could be verified by trying to modify this memory using evolution experiments. When growing cells in periodically changing environments, with different lengths of starvation periods, I would expect the delay to increase if the starvation periods are short, and decrease if they are very long. This could first be tested in simulations, with dynamics initially fit to the data in Figure 3.8.

Overall, by training neural networks to recognise organelles in bright field images, I determined single-cell timings, and showed that there is significant delay between the encoding of a stress and the response in population data. I also showed that both are relatively well synchronised

across the cells, suggesting that there are a fixed set of intermediate steps that are a cause of the delay.

Chapter 4

Label-free cell cycle predictions

4.1 Introduction

4.1.1 Stress and the budding yeast cell cycle

In the conditions provided by the experiments in this thesis, budding yeast replication occurs through the vegetative cell cycle in four phases: G1, S, G2, and Mitosis. Initial growth occurs in G1, DNA replication in S phase, secondary growth and repair in G2, and the split of both nuclei (anaphase) and cells (cytokinesis) happens during Mitosis. At various moments during the cell cycle, the cell passes through regulatory checkpoints. At these checkpoints, the cell checks whether the conditions are favourable to enter the next step of the cell cycle. If they are not it extends the current cell cycle phase until the conditions have changed. The START checkpoint, in particular, is when the cell verifies that the external conditions are good enough to start a new cell cycle. The START point is irreversible [18]. After START, cells must complete the cell cycle even if the environment changes; this may affect their ability to survive a sudden switch into an unfavourable environment, such as glucose starvation.

When the cell undergoes glucose starvation, its position relative to START can affect its reaction to the stress. More specifically, it affects how the cell goes about its replicative cycle. If the cell is in G1 when the stress is sensed, then it does not pass the checkpoint. In this case, the cell will enter a quiescent state called G0 until the external conditions have changed for the better [37]. Signals used by the cell to determine whether or not to grow include growth

rate [2, 29], and glucose signalling [12]. If the cell has already passed START, then it must finish the cell cycle. The length of the cell cycle may change as the cell's growth slows down, however. In particular, the duration of mitosis may change to ensure that the bud reaches a certain size before becoming independent [66].

In order to determine how cell cycle phase interacts with other aspects of glucose deprivation stress response, I endeavoured to obtain cell cycle information in a label-free manner. I began by trying to find cytokinesis: this in combination with the BABY estimation of START (using the appearance of a bud) allows separation between pre-START and post-START cells. I show that the growth rate estimations obtained from BABY can be used to accurately and robustly determine the point of cytokinesis.

I then trained a classifier to split the cell cycle into four phase: G1, S, G2/M, and Anaphase using a ground truth obtained from the Htb2 marker [31]. When this classifier returned a poor accuracy, I attempted the more difficult task of image-to-image translation from bright field to Htb2, this time including brightness information. I found that two variations of the U-Net were able to translate from bright field to the GFP channel. I also show that the fluorescence values obtained from the generated images are periodic, following the period of the cell cycle.

4.1.2 Contributions

- Results in section 4.2 were obtained by Diane-Yayra Adjavonfrom data obtained by Ivan B.N. Clark and Alán Muñoz.
- Methods in section 4.3 were developed by Karen Trippler in the context of her master's thesis, supervised by, Diane-Yayra Adjavon, Julian M.J. Pietsch, and Peter S. Swain. The results were independently obtained by Diane-Yayra Adjavon.
- Results in section 4.4 were obtained by Diane-Yayra Adjavonfrom data obtained by Ivan B.N. Clark.

4.2 Label-free prediction of cytokinesis

I began by trying to obtain the point of cytokinesis from growth rate information. The instantaneous growth rate of the cell varies across the cell cycle. In particular, the growth is mostly within the mother cell during G1, and then predominantly in the bud during S/G2. At the

point of cytokinesis, the growth is again predominantly in the mother, in preparation for the next cycle.

By observing the dynamics of growth of the mother and bud cells separately, we are able to determine the points at which these switches from mother-growth to bud-growth occur. Thanks to the BABY algorithm (see section 2.3), we can obtain this information label-free, directly from the segmentation of bright field images. I compared these growth rate dynamics to the changes in fluorescence of a cell cycle marker: a schematic representation of their relative dynamics is show in Figure 4.1a.

Since we are interested in cytokinesis, I use images of cells fluorescently tagged with Myo1-GFP. Myo1 forms a part of the actin-myosin contractile ring at the bud neck of budding yeast cells. This ring forms the connection between the mother cell and the bud; it is required for cell separation during cytokinesis [67, 70]. It is localised at the bud neck. Myo1-GFP fluorescence appears during S phase as the cell begins to bud, then abruptly decreases back to baseline state at cytokinesis. In the following, I use the downshift of Myo1 fluorescence as the point of cytokinesis against which I compare the algorithm.

I first verify that the growth rate is indeed periodic with the cell cycle in our set up. In Figure 4.1b, I then align the growth rate by cytokinesis obtained from Myo1 fluorescence. We can observe the expected two phases of growth. The bud dominates growth during S/G2/M, with its growth rate peaking midway through that period. Simultaneously the mother's growth rate is decreasing. Bud growth rate decreases when approaching cytokinesis: at this point the mother and bud growth rate are at their closest, crossing in many instances. After cytokinesis, the mother growth rate increases again, peaking during G1.

I used these observations to predict cytokinesis from growth rate: assigning the point where mother and bud growth rate cross as cytokinesis. The algorithm makes use of the uncertainty measures on growth rate. The crossing point is reached when the difference between the two growth rates is below a given threshold. I set a range of thresholds, moving from smallest to largest, to account for noise in the growth rate measurements.

1. Measure the distance between mother and daughter growth rate.
2. If there is a time point where this distance is less than the threshold value, and
3. If the growth rate of the daughter is larger than the threshold value at that time, then

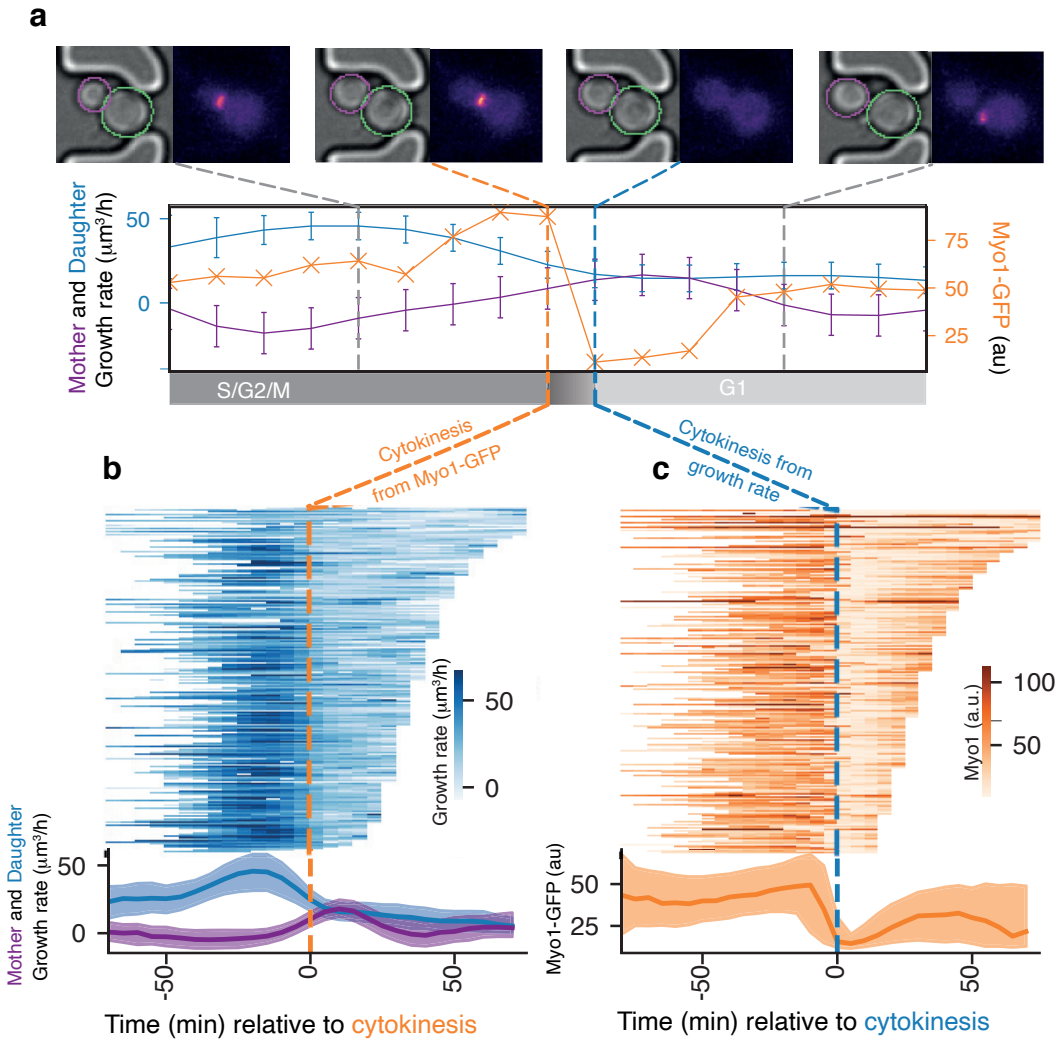


Figure 4.1: **Growth rate predicts Myo1 downshift.** (a) A representative cell cycle. The point of cytokinesis estimated from Myo1 fluorescence (respectively Mother/Daughter growth rate) is shown as an orange (respectively blue) dashed line. I show the bright field and Myo1-GFP images before, during, and after cytokinesis (top). (b) The growth rates for the daughters of 268 cells, split by cell cycle, shown as horizontal lines with the shading indicating magnitude. Cell cycles are aligned by the time of the drop in Myo1 fluorescence. At the bottom: the population median and interquartile range of mother (purple) and daughter (blue) growth rates. (c) Myo1 fluorescence of cells in (b), aligned by cytokinesis predicted from the mother and daughter growth rates (top), with population median and Inter-Quartile Range (bottom)

4. Choose this point as the point of cytokinesis.

5. Otherwise, move to the next threshold value.

Additionally, we require a minimum growth rate of the daughter at cytokinesis. This avoids errors during transient periods of high stress when the growth rates of daughter and mother both dip significantly, possibly causing spurious crossing points that are not related to the cell cycle. If the conditions above are not met for any of the threshold values in the set range, a

cell cycle is marked as unknown and not used in further analysis.

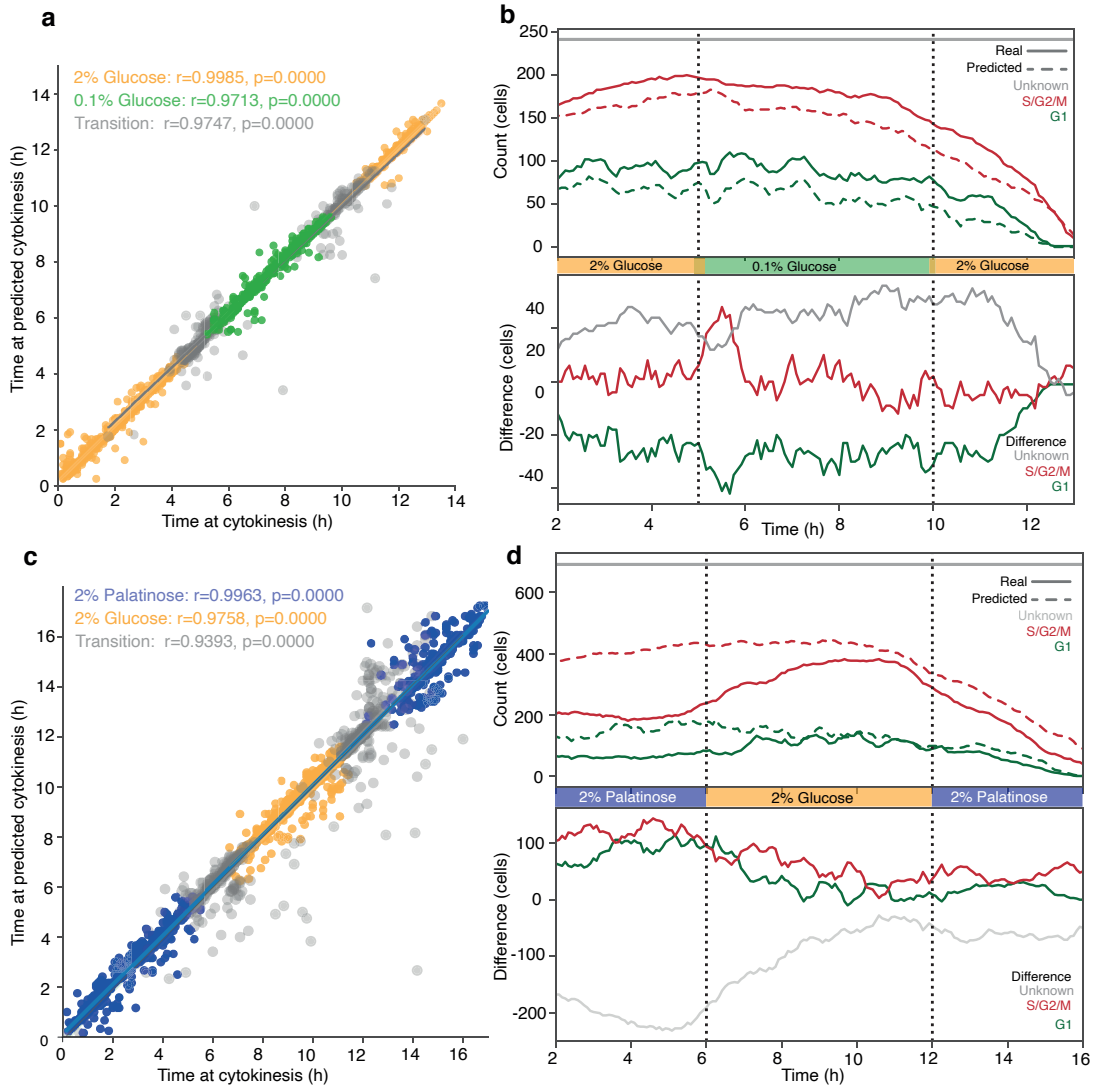


Figure 4.2: **Prediction of cytokinesis is highly accurate.** (a) Cytokinesis vs predicted cytokinesis for cells under going a transition from Glucose 2% in synthetic complete (SC) media, to 0.1% glucose in SC, and back. Transition points correspond to cell cycles that begin during one condition and end in the next. I show correlation coefficient and p-value for all three conditions. (b) Proportion of cells in G1 or S/G2/M across the experiment, based on cytokinesis from Myo1 data (Real) or predicted from growth rate (Predicted). Shown as a stacked graph. At the bottom, the difference between real and predicted number of cells, across the experiment. (c) Same as (a), but cells are grown in 2% Palatinose in SC, then 2% Glucose in SC, and back. (d) Same as (c), but with Palatinose and Glucose.

Cytokinesis predicted from this crossing point is highly accurate across conditions, as depicted in Figure 4.2. I compare the cytokinesis point predicted from growth rate with cytokinesis based on Myo1 fluorescence in two experiments containing changes in media. The first is a glucose downshift experiment: beginning at 2% glucose, shifting down to 0.1% glucose after six hours, then shifting back to 2% glucose. The second follows the same format, but shifting between 2%

Palatinose in SC and 2% glucose in SC media. The Pearson correlation coefficient is between 0.939 and 0.998 across all conditions, including for cell cycles that occur during transitions. As further confirmation, I aligned the Myo1 fluorescence of the cell cycles by predicted cytokinesis in Figure 4.1c, and observe that the cytokinesis point coincides with the downshift of Myo1 as expected.

The errors that are made are well distributed across conditions and cell cycle phases. In Figure 4.2b and Figure 4.2d, I show the proportion of the population in each cell cycle over time for both experiments, based on predictions of cytokinesis or on cytokinesis obtained from Myo1 fluorescence. I separate the cells into G1 (after cytokinesis, before budding) and S/G2/M (after budding, before cytokinesis). We see that the population dynamics are similar in both cases. For instance, after a switch from palatinose to glucose, the population shifts to mostly S/G2/M as the cells grow faster during G1 (Figure 4.2d). Additionally, we can see that there is a slight lag during which no new cells are in G1, after which the population seems to synchronise cell cycles as seen in the oscillations of G1 phase. This is likely due to the cell that are in G1 during the switch adapting to the new environment before crossing START and beginning a new cell cycle. As mentioned in chapter 3, the adaptation time of cells after such a switch is consistent across the population.

Note that there is still quite a high proportion of unknown states in both the predictions based on Myo1 and on growth rate. In particular, the number of unknowns increases significantly towards the end of the experiment. This is mostly due to the way that I separate cell cycles. Extracting full cell cycles from the end of the experiment would require obtaining data on budding that occur after the experiment has ended. Thus the increase in unknown cells is mostly an increase in cells for which we haven't observed the next cell cycle. Interestingly, the prediction of cytokinesis from growth rate does not produce many more unknowns during the rest of the experiment than the prediction from Myo1 fluorescence. We could increase the number of predicted cells by increasing the range of distance thresholds, but this would come at a cost both in terms of accuracy and in terms of computation time.

By combining these results with the beginning of S phase obtained from BABY, we can therefore separate cell cycles in to pre- and post-START in a label-free manner.

4.3 Cell cycle classifier

Although it is the most commonly studied, START is not the only blocking checkpoint in the budding yeast cell cycle. In particular, the duration of mitosis is modulated by nutrients with the transition to Anaphase as a potential blocking checkpoint [66]. A better understanding of the effect of cell cycle phase on response to nutrient stress would therefore require a more fine-grained classification of cell cycle phases than can be obtained with the method above. To this end, we looked for a classifier that would be able to separate the cell cycle into four phases: G1, S, G2/M and Anaphase.

To obtain training data for this classifier, we used the cell cycle marker Histone 2B (Htb2). The Histone 2B is involved in chromatin assembly, and as such its amounts a proportional to the amount of DNA in the cell [9]. This means that during S phase it doubles, and it decreases during Anaphase when the nucleus is split from the mother to the daughter. It is possible to use this to split the cell cycle into the four phases described earlier [31]. We therefore create our training data from images of this strain. This data pre-processing was done by Karen Trippler.

To label the cell cycle phase, we once again split the cell cycles as occurring between two BABY-predicted birth points. Then, following [31], we fit a piecewise-linear model to each cell cycle. The first phase, G1, is essentially flat. S phase has a positive slope. G2/M is once again flat, and Anaphase/Cytokinesis is fit with a strongly negative slope. As the marker is located in the nucleus, it is not possible to distinguish between anaphase and cytokinesis.

From this fit we create labels that we assign to the corresponding bright field z-stack snapshots. In order to reduce errors, we use several different fitting methods on the data and choose the result with the highest likelihood. When two methods are equally confident, we assign the cell with two different cell cycle phases. The bright field images, along with a mask of the cell in question, are the input to the network. An example labelling of a cell is shown in Figure 4.3a. The full data set is described in Figure 4.3b, including the labels where there is uncertainty. As expected, there are more examples of some labels than others, proportionally to the lengths of each phase of the cell cycle.

Based on the labels obtained, we additionally measure the length of the different cell cycle phases across the experiment. I show these in Figure 4.3c, with the length of the phase in minutes. These phase lengths generally correspond to those found in previous uses of this method [31].

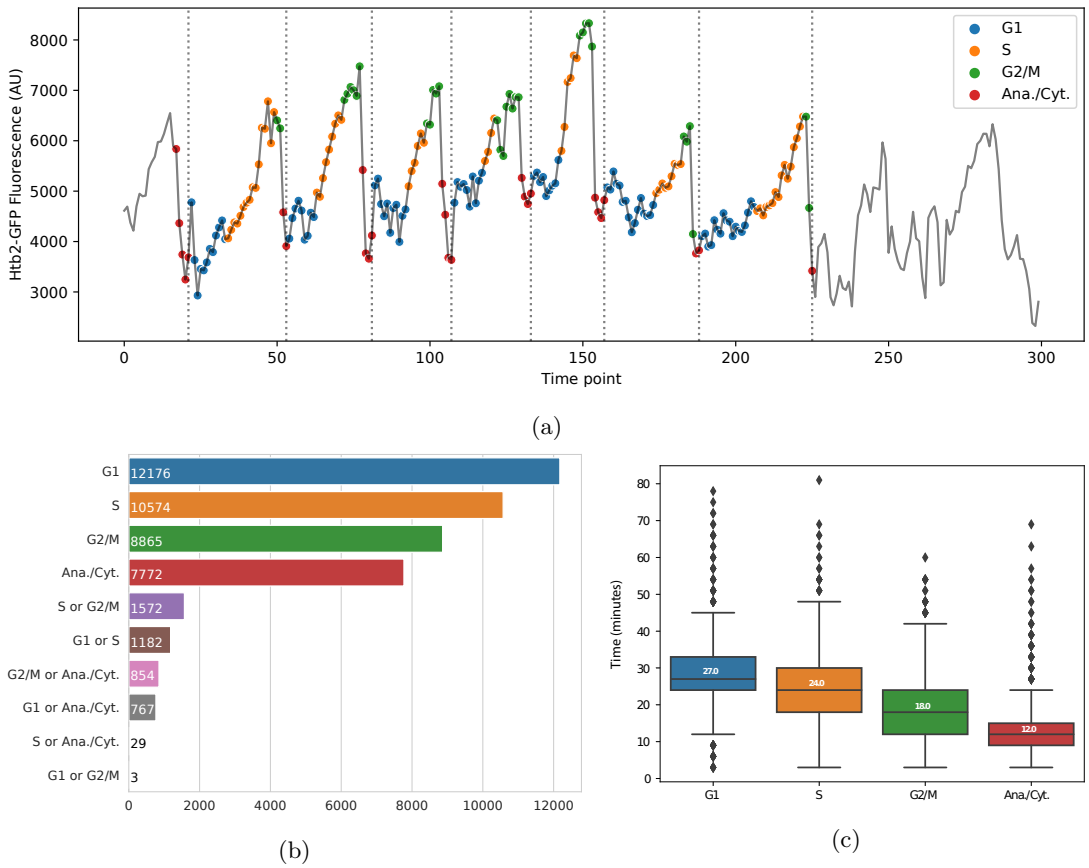


Figure 4.3: **Cell cycle label generation.** (a) Htb2 fluorescence (blue) and corresponding piecewise-linear fit, with colours representing the attributed label. (b) Number of samples per label in the training and testing set. We do not balance the data sets, but rather apply weights to each label during training to counter the class imbalance. (c) Length of cell cycle phases based on the labels.

We first trained a VGG16 classifier [111] in Tensorflow. We use binary cross-entropy as a loss function, and run gradient descent with the Adam optimiser with the default parameters provided by Tensorflow. This network is not able to learn the classification: its final accuracy is equivalent to that of a random guess: approximately 25%. Interestingly, it systematically returns a single cell cycle phase. We postulated that this might be related to the imbalance in the number of examples in each phase in our training data set, as shown in Figure 4.3b. We added weights to the different phases, such that the network gives a higher importance to the cell cycle phases that had fewer examples (e.g. Ana./Cyt.). This did not improve the accuracy of the model. I show the final results can be seen in Figure 4.4, as a confusion matrix. The labels of at the bottom of the matrix represent the labels predicted by the network, and the vertical labels correspond to the true label of the cell. The values in the matrix show the number of cells in the testing set with each pair of labels. For example, the value in the top right corner says that 196 of cells had label G1 but were predicted by the network as label

Ana./Cyt.

Importantly, the network does not favour any given phase of the cell cycle. In other words, when the network is re-trained, the phase it returns changes. In Figure 4.4, I show the result of only one training run for each network.

I hypothesized that the VGG16 network was too simple to capture the complexity of the data, thus preventing it to learn the task. I therefore trained a more complex classifier network, ResNet50 [42] on the same data, with class weights to counter-act the imbalance of cell cycle phase examples in the training data. The results were somewhat improved, as can be seen in Figure 4.4. This network reaches an accuracy of about 40% on the testing set. Although this is much higher than the VGG16 network, this accuracy is still too low to use in practice.

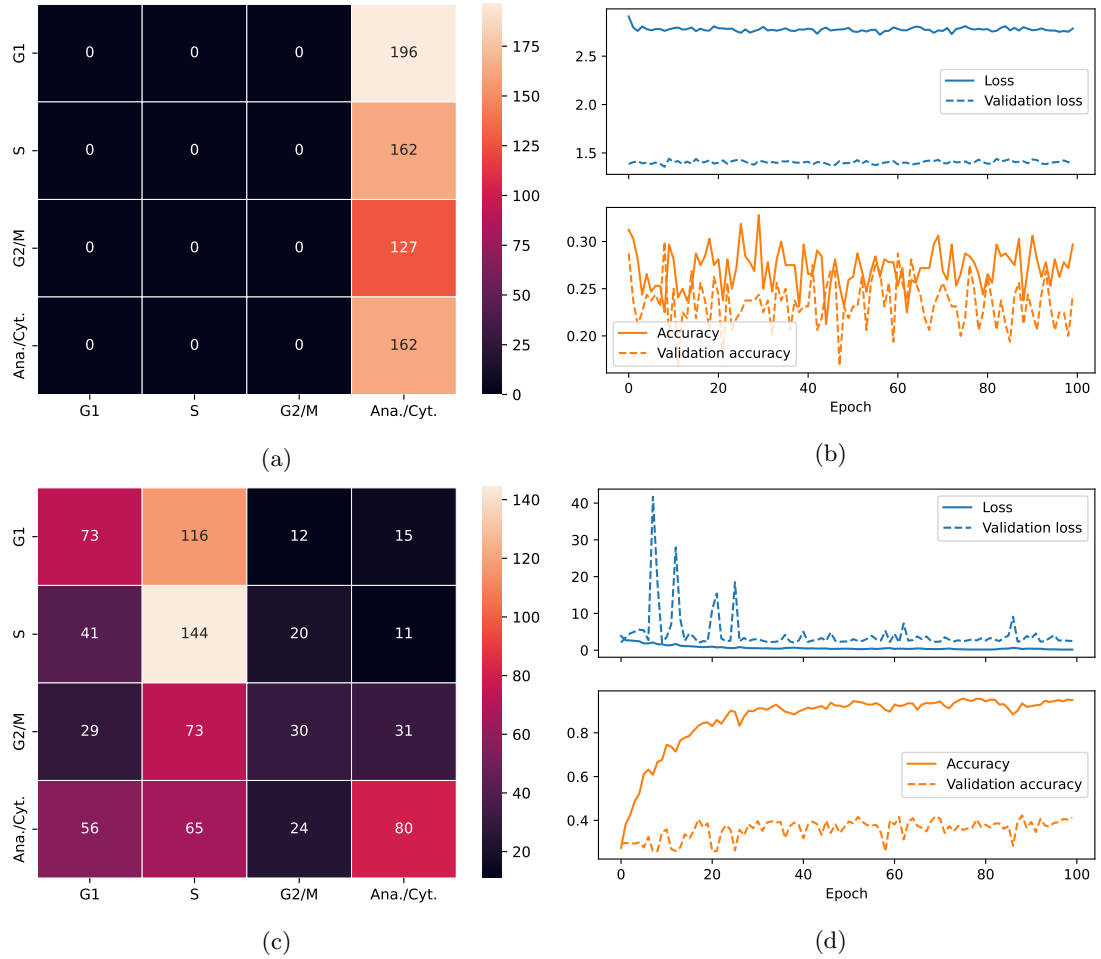


Figure 4.4: **Performance of cell cycle phase classification.** (a) Confusion matrix of VGG16 classifier after 100 epochs. (b) Cross-entropy loss of VGG16 classifier during training. (c) Confusion matrix of ResNet classifier after 100 epochs. (d) Cross-entropy loss of ResNet classifier during training.

The mistakes made by the ResNet give an indication as to why the accuracy is so low. The

majority of mis-classifications correspond to assigning one of the adjacent phases. For example, when the target label is S, a majority of mis-classifications are in G1 (116 mis-classifications). This suggests that the uncertainties in labelling our training data may be preventing the network from properly learning. In other words, errors may have been introduced into the training data during the fitting process used to obtain the labels. I therefore re-defined the problem to work directly with the raw fluorescence data.

4.4 Continuous cell cycle label from bright field

4.4.1 Training data

For an even more fine-grained labelling of the cell cycle, I tried image-to-image translation to obtain the fluorescent Htb2 marker directly from bright field. We know that a U-Net is able to predict the location of Htb2 based on our nucleus-segmentation pipeline. Other works suggest that U-Net should also be able to predict the fluorescence values directly, rather than just a mask [21, 88].

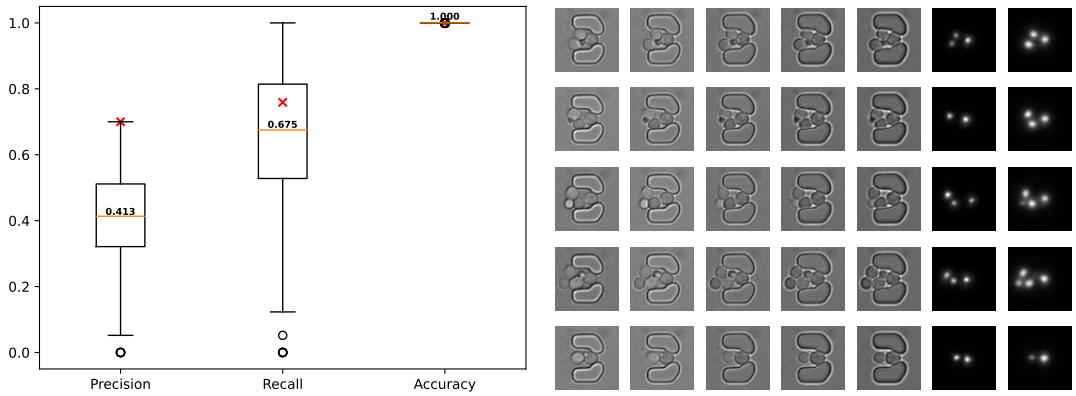
The training data is once again generated from an experiment with Htb2-GFP fluorescence. In this case, I use the bright field z-stack as an input, and fluorescence data as an output. To create the ground truth, I first take a maximum projection of the fluorescence data across the z-stack to get a single 2D image as an output. I then normalise the pixel values across the time-lapse, rather than for each ground truth image, in order to preserve the cell cycle information in the normalised image. I use a Min-Max normalisation around the median (Equation 4.1), where $\mathbf{Y}$ corresponds to the full three-dimensional time-lapse.

$$\mathbf{Y_{norm}} = \frac{\mathbf{Y} - Y_{median}}{Y_{min} - Y_{max}} \tag{4.1}$$

I then clip the values between -1 and 1: all values below -1 are set to -1, and all values above 1 are set to 1. As as result, the ground truth images in cells with a low fluorescence will have mostly values close to -1, while those at peak Htb2 fluorescence (at the end of S phase) will have many values close to 1.

4.4.2 Training the U-Net

I begin by training the U-Net architecture from chapter 3 on Htb2-GFP data. Unlike previously, I do not modify the fluorescence data to turn it into a binary mask. Instead, I kept the brightness information in the target data, normalising the fluorescence data as described in subsection 4.4.1. I used Mean Average Error or L1-loss, and an Adam optimiser with default parameters. I trained for a maximum of 100 epochs, with early stopping if the validation loss does not improve over 15 consecutive epochs.



(a) (b)

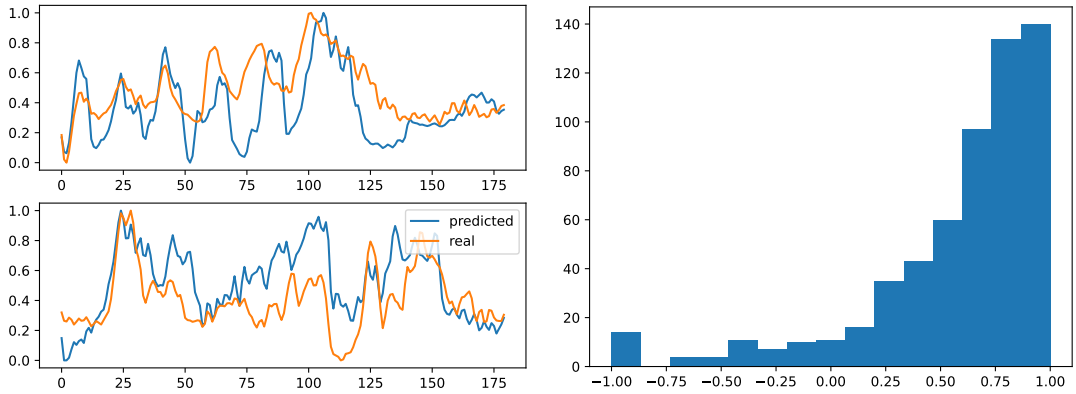
Figure 4.5: **Metrics and examples of U-Net trained on Htb2-GFP data.**

We see in Figure 4.5 that the U-Net architecture struggles to recover the target fluorescence image, although its loss decreases steadily throughout training. More specifically, it seems to be retrieving the position of the cell, rather than that of the nucleus, and attributing the same absolute value of fluorescence to all of the cells in the image. I verify both of these things quantitatively on testing data.

I first verify how well the network is able to locate nuclei in the cells, by transforming the both the target and the output of the network into a mask, and computing precision and recall values. The results are shown in Figure 4.5a. As we can see, although the average recall is high, the precision has significantly decreased compared to when the U-Net was trained on masks in chapter 3. The network is therefore doing badly at fine-tuning to fit to the location of the nucleus. Furthermore, the noise is much high for both precision and recall in this case. In other words, the network is not consistent.

Next, I evaluate the network's ability to recover cell cycle information in the brightness of fluorescence. Using cell outlines obtained by BABY, I mask the fluorescence images to only

look at the pixels within each cell. On a cell-by-cell basis, I obtain the 97.5^{th} percentile of brightness across the cell as a measure of the fluorescence of Htb2. This is done both on the real fluorescence data and the network's prediction. I plot an example in Figure 4.6a. Unexpectedly, the network's fluorescence predictions do match the actual fluorescence, although with a reduced dynamic range. To measure how similar values are across all of the cells in the experiment, I obtain the correlation coefficient between the real Htb2-fluorescence time series, and the predicted time series. The correlation coefficients are shown in Figure 4.6b The correlation is generally high. This suggests that despite failing to find the nucleus within the cell with this marker, the network is still able to extract some cell cycle information from the bright field snapshots. These predictions could not be used to directly replace Htb2-GFP fluorescence, but they could be an alternative when only cell cycle times are needed.

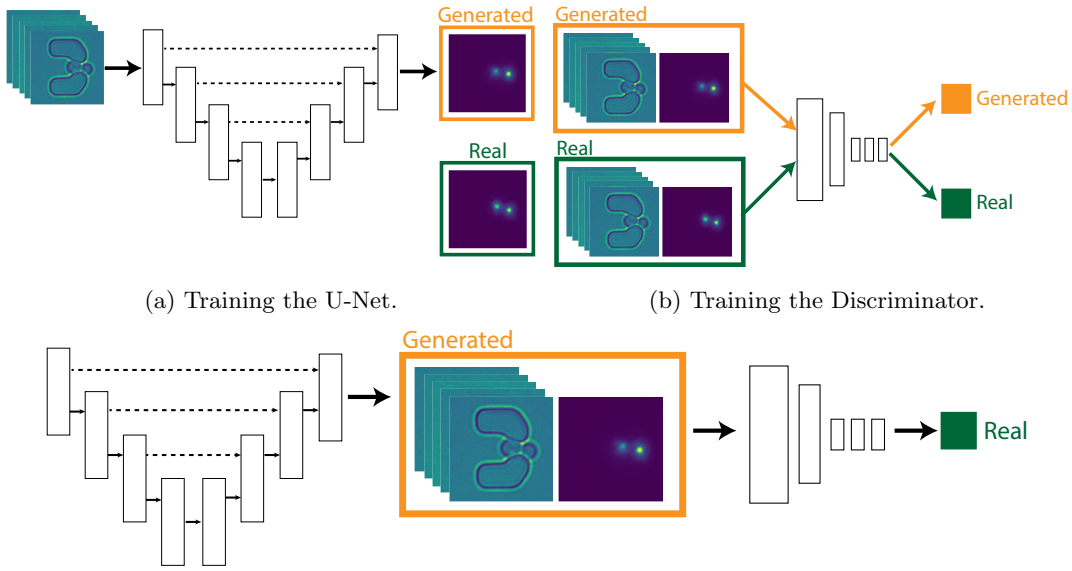


(a) (b)

Figure 4.6: **Evaluation of the U-Net trained on Htb2-GFP data.** (a) The predicted and real fluorescence oscillations during the cell cycle. Although the predictions do not directly match the real fluorescence time series, they follow the same oscillating trend. These data are smoothed representations of the top 97.5th percentile of fluorescence. (b) Correlation coefficients between the real and predicted time series across all of the cells in the testing data set. The correlations are generally close to one, meaning that there is a close match between the dynamics of the predicted fluorescence and the real one, despite the lower dynamic range in our predictions.

4.4.3 Training Pix2Pix

As we want to model the entire range of brightness, I turned to a modification of the U-Net that was designed for Image-to-Image translation. The Pix2Pix network [49] combines the U-Net structure with an adversarial network. The adversarial training aims to force the network to create realistic outputs; I tried to use it here to improve the dynamic range of fluorescence output by the network, as well as its localisation of the fluorescence to the nucleus.



(a) Training the U-Net.

(b) Training the Discriminator.

(c) Adversarial training.

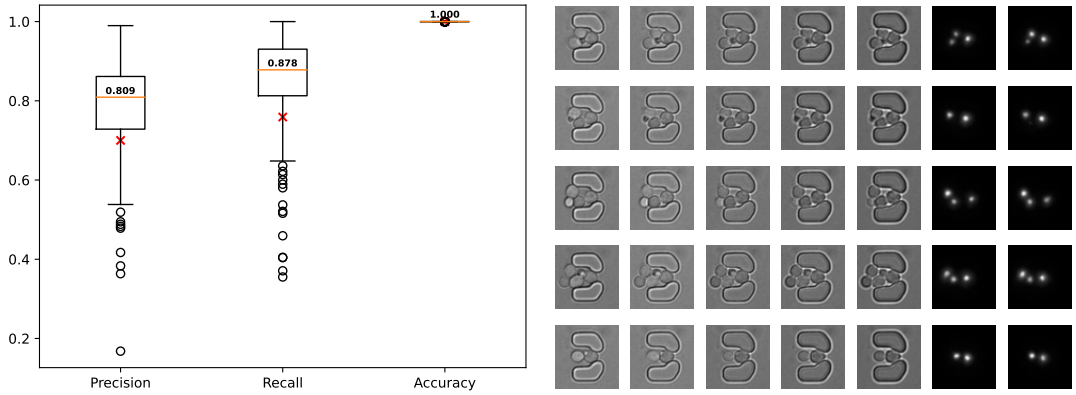
Figure 4.7: **Pix2Pix Network.** (a) Training the U-Net to reconstruct the fluorescence image (real) from the bright field input. The generated image is compared to the real one at each training step, and the parameters of the network are updated. (b) Training the discriminator to recognise the difference between real and generated samples. The discriminator is given both the Bright Field and the real or generated fluorescence as an input. It returns a classification into Real/Generator as an output. (c) Training the U-Net to trick the discriminator. The U-Net's parameters are updated to make its output as believable as possible, such that the discriminator will confuse its output with a real fluorescent image.

Training occurs in two phases: the first phase is the standard U-Net as seen in subsection 4.4.2. The second phase of training is adversarial: the U-Net output is fed into a Discriminator network that has to determine whether the image comes from the training data (label 1) or was generated by the U-Net (label 0). During this phase, the Discriminator is trained to recognise generated data, and the U-Net is trained to try to trick the Discriminator. The goal of this phase of training is to ensure that the U-Net outputs images that fall within the same distribution as the training set. This helps retain the low-to-high fluorescence values that come with the cell cycle.

The network was trained using an Adam optimiser with learning rate 0.0002 and beta value of 0.5 for each phase of the network. I alternated training the generative and the adversarial side of the network at each batch, to avoid giving either task precedence. The network is trained until convergence using an Early Stopping checkpoint with a patience of 10: this stops training once our metric does not improve for more than 10 epochs. I chose the mean average error on the validation set (the loss function on the U-Net's generative training phase) as a metric. I verified that the adversarial side of the network was not collapsing, meaning that

neither the Discriminator nor the U-Net were "winning" during the training phase. To verify this, I kept track of the binary cross entropy loss of the discriminator: it converged to 0.69 or approximately the natural log of 2, while the U-Net's adversarial loss converged to 1.74. [] At this point, the discriminator has reached an accuracy of about 50%, which means that it cannot tell the difference between the real and generated images and is guessing which is which randomly.

The results of the Pix2Pix network are evaluated in the same was as we did for U-Net in subsection 4.4.2. I evaluate the network's ability to find the nucleus by measuring precision, recall and accuracy. The performance of this network is significantly better than that of the U-Net alone. In fact, this network is even better at locating the nucleus in the image than a U-Net that was trained for this task alone, in chapter 3. This suggests that the adversarial side of the network did not hinder learning.

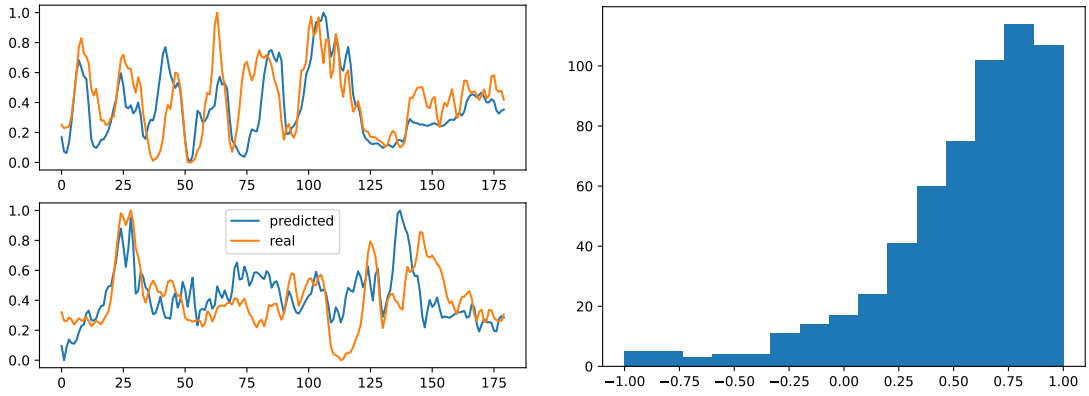


(a) (b)

Figure 4.8: **Performance of Pix2Pix nucleus segmentation.**(a) Metrics evaluating the performance of the Pix2Pix network for the localisation of nuclei within the cell. Both precision and recall are significantly improved compared to the U-Net alone, and are even generally higher than the specially-trained U-Net in chapter 3, although the evaluation is done on different data. (b) Some examples of inputs and outputs of the network. From left to right I show the five z-stacks of Bright field, given to the network as an input, the target image which is the maximum projection of real Htb2-GFP fluorescence, and the prediction of the network. The predictions are almost identical to the targets.

I then tested Pix2Pix's ability to recover the cell cycle information. The example traces are shown in Figure 4.9a; the predictions of Pix2Pix more closely resemble the real Htb2 traces in their dynamic range. Nevertheless, the same kinds of mistakes seem to be made. The correlations between the two time series across the testing set are shown in Figure 4.9b. Once again the correlation between real and predicted time series is high for a majority of cells in the testing set. Interestingly, however, the predictions of Pix2Pix are generally less correlated to

the real data than those of the U-Net alone. This suggests that although this network is better at finding the location of the fluorescence, it is doing worse at extracting cell cycle information. It may therefore be worth separating the two and predicting the time series of fluorescence directly rather than going through the proxy of image-to-image translation.



(a) (b)

Figure 4.9: **Pix2Pix predicts the cell cycle.** (a) An example Htb2 fluorescence trace for a single cell (blue) and the same trace from the Pix2Pix fluorescence prediction (orange). The 97.5^{th} percentile of brightness in the cell is used as a proxy for nuclear fluorescence here. (b) Pearson correlation coefficient between the fluorescence time-series of Htb2-GFP and the fluorescence time-series predicted by the Pix2Pix network, for all cells in the testing set.

The addition of the discriminator, and the adversarial phase of training do not improve the network's ability to recover the cell cycle. Additionally, because of the adversarial nature of the network, these results are difficult to reproduce. Subsequent training iterations on the same network have led to results closer to that of the U-Net above. Some differences in training could explain the results.

Overall, these results are promising as they suggest that there is indeed information on the cell cycle contained in the bright field images.

4.4.4 Measuring the length of the cell cycle

The next question is whether these predictions could be used to give interpretable and verifiable information about the cell cycle. I additionally evaluate the usefulness of the prediction by looking at its ability to predict the length of the cell cycle. I obtain the cell cycle by taking the autocorrelation of the time series with various lags. This method computes the correlation of a time series with a shifted version of itself. At a lag of 0, the auto-correlation value is 1: the time series is exactly correlated with itself. For a periodic time series, the autocorrelation

increases again after each multiple of one period. As Htb2 fluorescence is periodic with the cell cycle, the autocorrelation of this time series peaks after each cell cycle. I measure the length of the cell cycle by obtaining the peaks in autocorrelation, as shown in Figure 4.10

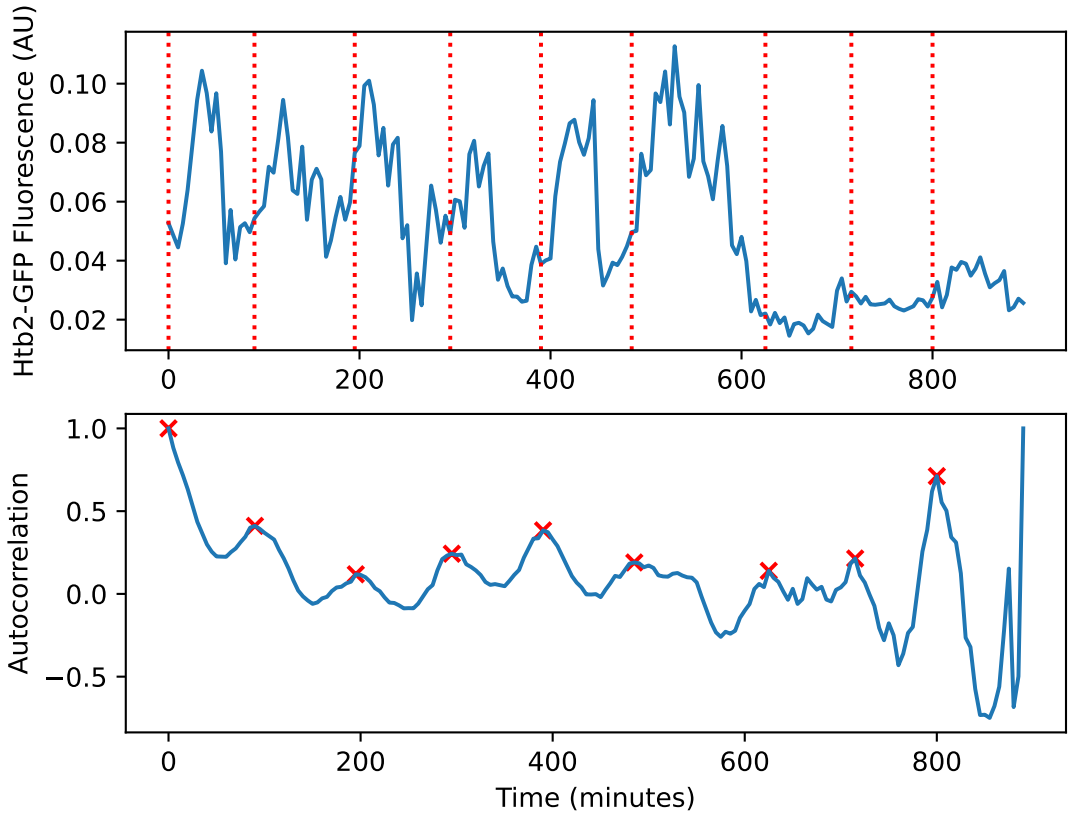
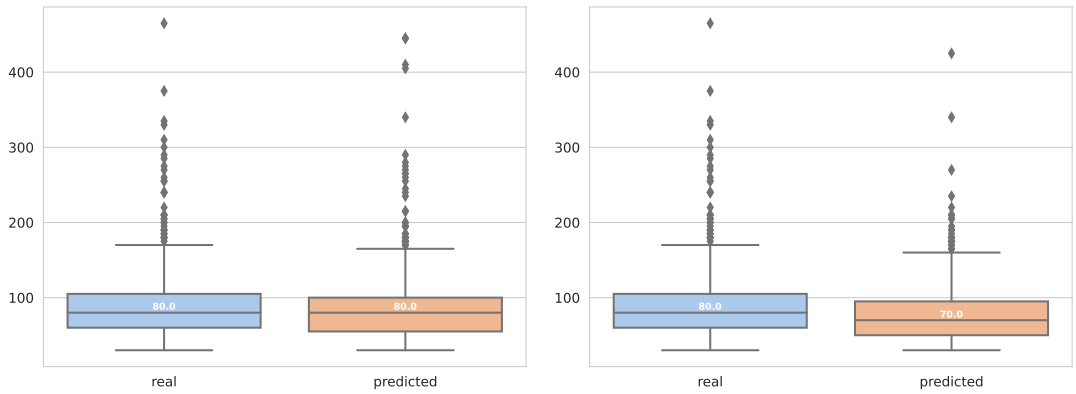


Figure 4.10: **Cell cycle from autocorrelation.** (top) An example trace of Htb2 fluorescence, with vertical lines splitting cell cycles. The cell cycles are not defined relative to any given phase, but relative to which ever phase the cell was in at the beginning of the experiment. In this case, the phase is late G1. (bottom) The autocorrelation of the above trace, with the peaks used to define a new cell cycle shown as red crosses.

The results for both network are in Figure 4.11. The predictions made by U-Net are more adept at finding cell cycle lengths than those made by Pix2Pix, despite the lesser ability to locate nuclei and the reduced dynamic range see in Figure 4.6a and Figure 4.9a. This is in line with the fact that the U-Net predictions are more correlated with the real data. The average cell cycle length of 80 minutes corresponds well to the cell cycle length obtained in section 4.3.



(a) U-Net (b) Pix2Pix

Figure 4.11: **Cell cycle lengths from fluorescence predictions.** The cell cycle lengths for all of the cells in the testing data set (never previously seen by the network, even during validation) are shown. These were computed from peaks in the auto-correlation of the Htb2-GFP fluorescence, or in the predicted fluorescence. The predicted distribution of cell cycle lengths matches the real one, especially for predictions made with U-Net. Cell cycle lengths are systematically underestimated by the Pix2Pix prediction.

4.5 Discussion

At first glance, it is not obvious that there is cell cycle information accessible in Bright field images. Aside from budding, there is little clear indication of phase of the cell cycle, even to expert eyes, in these snapshots. With the right transformations, I showed that is possible to use information extracted from bright field to predict the cell cycle.

That growth rate contains cell cycle information is consistent with existing literature [29, 66]. Cytokinesis prediction, when combined with birth annotations from BABY, allows us to split cell cycles into G1 and S/G2/M phase, or pre-and post START. This is already enough to verify the effect of cell cycle stage on stress response. In stress experiments, this method can be used to compare the fitness of cells based on whether stress occurs before of after they have passed START.

The classifier results suggest that splitting the cell cycle into G1, S, G2/M, and Anaphase/Cytokinesis, based only on snapshots, is not possible. Since we are able to predict at least part of this from growth rate time series, a possible improvement would be to give the classifier timing information. This could be done simply, by making predictions on a window of time-points at once rather than on snapshots: the network would be able to take into account the evolution of the bud size, for example, in its predictions. In order to include time implicitly in the network, we could look into networks specifically designed for this, such as the Long Short-Term Memory

Network used in similar tasks [5].

I was, however, able to predict a cell cycle marker from snapshots of bright field. This labelling is particularly promising because it can not be done manually. Although some points in the cell cycle have obvious visual cues in bright field for manual annotation (for instance, the appearance of a bud as START), the majority of the cycle information can only be obtained through fluorescent markers. With this network we can introduce information into experiments that we would otherwise not be able to obtain at all.

The fact that the networks were able to predict Htb2 fluorescence, which is enough to determine cell cycle stage (section 4.4), but not directly classify cell cycle stage (section 4.3) is still unexplained. One possible explanation is that there was a difference in the quality of our ground-truth annotations for the two tasks. 7eep learning is often only as good as its training data [34]. For the image translation, there is a direct correspondence between the fluorescence and the bright field image.For the classification, however, our annotations were automated and not verified for quality. It is therefore possible that the classifier was training on incorrect ground-truth labels. Even if the bright field snapshots contain enough information to be able to classify the cell cycle phase, which our image-translation results suggest that they do, the classifier would not be able to learn them. This could be verified by trying to train the same classifier with the Htb2 fluorescence, rather than the Bright field images, as an input.

Once these issues have been resolved, an interesting future direction would be to try to find out what cues within the bright field z-stack the network is using to produce these fluorescent images. An advantage that the network has over a human observer is that it can use the data in all five z-stacks simultaneously, so it is possible that there are depth-based features that are distinctive across the cell cycle. Several attribution methods could be use to make the network's predictions more interpretable [116, 136, 139]. These are methods that help determine which features in the input were most used by the network to produce its output. A review including a discussion of interpretability of deep learning in bio-medical sciences can be found in [19]. One could apply these interpretability methods to our network to figure out what the most salient features of a cell in each phase of the cell cycle are. This could, in turn, be verified against known morphological changes during the cell cycle, or generate new testable hypotheses.

I have shown that obtaining cell cycle information from bright field images is feasible. When combined with stress response markers, these methods will allow us to determine what effect the cell cycle has on the timing of the cell's stress response, as well as on the cell's fitness.

Chapter 5

Metabolic adaptation as a local optimisation

Encoding and navigation in a compressed space

5.1 Introduction

Evolution in variable environments has outfitted microorganisms with a variety of ways to obtain the energy and components necessary to grow and divide. Due to limited resources, they have simultaneously evolved decision-making strategies, which allow them to choose the best metabolic pathways for a given task and environmental condition.

Constraint-based models of metabolism and their accompanying analyses have been instrumental in predicting metabolic decisions at a large scale, and without the need for specific information about gene regulation or the kinetic constants of enzymes. Curated genome-scale constraint-based models of metabolism now exist for many different organisms, are regularly updated by an active community, and software is readily available to perform most common analyses. Most analyses on constraint-based models are steady-state, and cannot be used to describe the transition of the cell's metabolism from one state to another. Moreover, these

methods often find a single global optimum point for any given input medium, and do not reflect the cell-to-cell variability found in experimental settings.

In this work, I first give a brief overview of constraint-based models of metabolism, and describe the different ways in which they are commonly analysed. I describe previous attempts to introduce dynamics and variability into these analyses. I then introduce elementary flux modes, a basis for the metabolic space defined by the constraint-based model. Then, I discuss the information cost of running a global optimisation on such a space.

Next I build upon the elegant simplicity of constraint-based models to develop a dynamic model of metabolic adaptation. I begin by working on a toy model, describing a navigation of the metabolic space based on similarities between flux modes, and on matching exchange reactions to the environment. Then I describe how to encode a full genome-scale into a tractable space, and show that the space naturally describes different strategies along different directions. I then demonstrate how it is possible to navigate this space, using information about the environment, despite lossy encoding. The model does not require full knowledge of the metabolic space, nor a proxy description of growth rate. Additionally, it is dependent on initial-conditions, and naturally describes cell-to-cell variability, as contrasts with the conventional methods.

5.2 Constraint-based models of metabolism

5.2.1 Biological description and Mathematical formulation

Constraint-based models describe the metabolic biochemical network of a given organism (or set of organisms in a culture) in terms of the reactions that can occur and their stoichiometry. As such, they do not require knowledge of kinetic parameters of the enzymes in the network, unlike kinetic models. The network is represented in the form of a directed graph (see Figure 5.1), where each node represents a metabolite and each edge represents a reaction connecting two metabolites. Each reaction is parametrised by a rate or flux. A boundary is drawn, categorising metabolites as either internal to the cell or part of the external environment. Reactions among internal metabolites are considered metabolic reactions, and reaction involving an external metabolite are considered exchange reactions.

All internal metabolites are assumed to be at steady state, which forces the input fluxes and output fluxes of each reaction to balance out to fit stoichiometric constraints. This balance limits

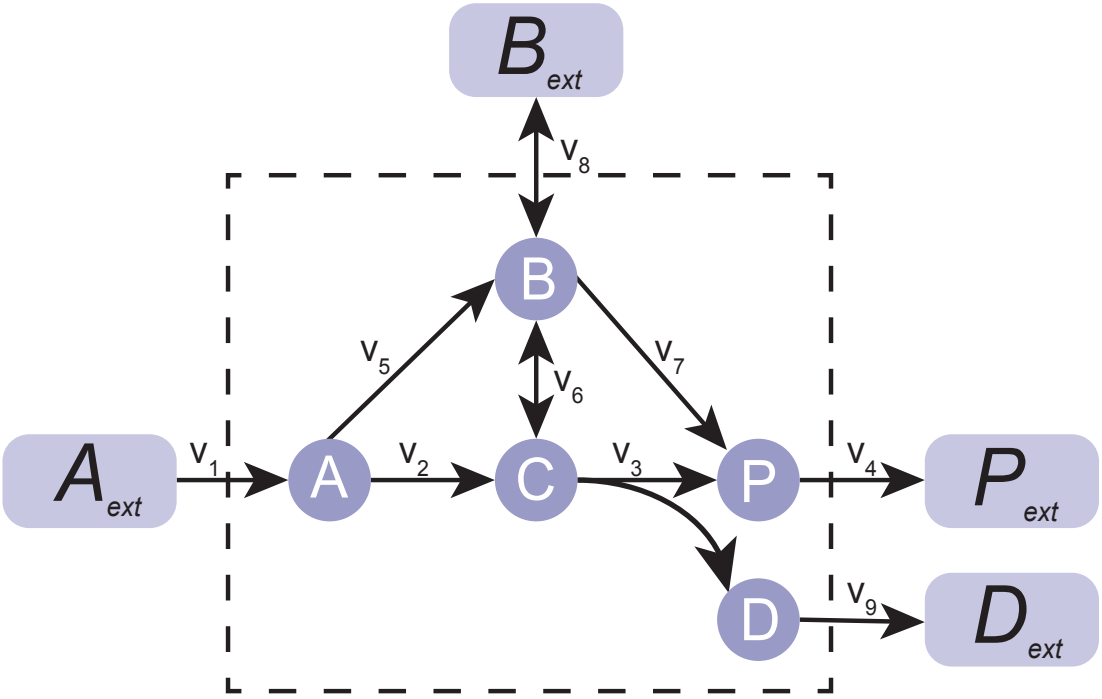


Figure 5.1: **An example constraint-based model.** The circular nodes correspond to metabolites, with the reactions as edges linking them. Some metabolites also undergo exchange reactions: these are seen as reactions entering or leaving the cell's interior (dotted line), and the extra-cellular version of these metabolites are portrayed in squares.

the space of possible fluxes. Mathematically, this can be summarised by the simple constraint in Equation 5.1, where $\mathbf{S}$ is the stoichiometric matrix and $\mathbf{v_{int}}$ is the vector of reaction fluxes that only include internal metabolites.

$$\mathbf{S v_{int}} = \mathbf{0} \tag{5.1}$$

To restrain the model to biologically plausible flux rates, upper ($\mathbf{v_{max}}$) and lower ($\mathbf{v_{min}}$) bounds are additionally put on the fluxes ($\mathbf{v}$). Irreversible reactions have a lower bound of 0, whereas reversible ones may have a negative lower bound. For internal reactions, these bounds are usually set generously (e.g. from 0 to 1000). For exchange reactions, that is reactions that represent a movement of boundary metabolites to or from the environment, the bounds are linked to the concentrations of the metabolites in the medium, and whether the exchange is an uptake or a secretion. The general convention is to consider uptakes as negative fluxes, and secretions as positive fluxes.

$$\mathbf{v_{min}} \leq \mathbf{v} \leq \mathbf{v_{max}} \tag{5.2}$$

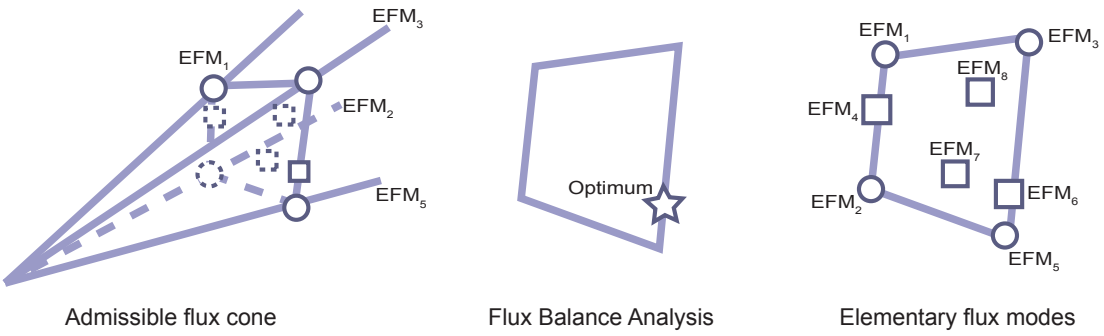


Figure 5.2: **Types of analyses on constraint-based models of metabolism.** The model and its stoichiometric constraints defines an admissible flux cone (a): an area of the metabolic space that does not violate the constraints. Flux balance analysis (b) finds the state of the network that is optimal for a given objective function (usually, a proxy for growth rate). Elementary flux modes (c) are a base of the entire admissible flux cone, and optimisation usually chooses one of these flux modes, or a combination of few equivalent flux modes.

5.2.2 Analyses on constraint-based models

The metabolic space resulting from the constraints in Equation 5.1 and Equation 5.2 is called the admissible flux cone (Figure 5.2a). In most conventional analyses of constraint-based model, this cone is searched for a configuration of reaction rates that can sustain steady state and, optionally, maximises an objective criterion, such as biomass production rate. The obtained reaction rates imply a distribution of corresponding metabolite concentrations (up to a multiplicative factor).

The most common analysis method is Flux Balance Analysis (FBA) (Figure 5.2b). It consists of choosing among the feasible flux vectors $\mathbf{v}$ — the solutions of Equation 5.1 — by maximising an objective function. Usually, this objective function is a linear combination of fluxes manually curated as a proxy for growth; it is often called the biomass production rate. Because the system of equations is linear, even large metabolic networks can be tractably solved using linear programming.

The choice of this objective function stems from the assumption that cells regulate their metabolism in order to optimise growth. Classical flux balance analysis tends to over-estimate the growth rate of populations of microbes, and various alterations have been introduced to better explain experimental data. Some include alternative objectives, including multi-objective Pareto optimisation, for a better fit with fluxomics data [39,107]. Flux balance analysis has also been extended with the addition of other constraints such as thermodynamic limitation [80], and intracellular crowding [10]. Finally, several models have been developed to include optimal allocation of proteomic resources into metabolic models. Usually, this consists of minimising the

protein cost of catalysing the metabolic reactions, for example through an additional objective function [107, 127] or through additional constraints [77] and bounds on fluxes [105]. In some cases, the optimisation of resources has been explicitly added to genome-scale models [33, 134], including the production of proteins from metabolic intermediates and the reliance of metabolic reactions on enzymes into the stoichiometric matrix and boundary conditions.

5.2.3 Elementary flux modes

The stoichiometric matrix, boundary conditions, and steady-state assumption yield a set of flux vectors that sustain steady-state, named feasible flux vectors. The space in which these vectors live can be represented geometrically as a cone in a high-dimensional space, illustrated in Figure 5.2. In flux balance analysis and its extensions a particular vector in this cone is chosen by optimising for a predefined objective. However, by considering only the optimal vector, flux balance analysis ignores the large number of options that a cell has to choose from.

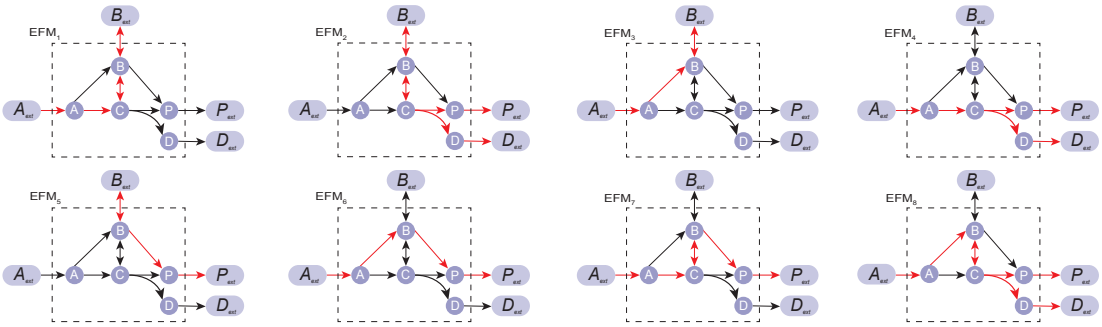


Figure 5.3: **Elementary flux modes.** The elementary flux modes of the model in Figure 5.1. The reactions that are active in these flux mods are shown in red.

Elementary flux modes are a spanning set of the flux cone that instead describe all of the feasible flux vectors [59, 122, 123]. They are unique, and none can be written as a combination of the others without requiring a non-zero flux in both directions of a reversible reaction, but are sufficient to describe all of the feasible flux vectors of the network. Indeed, with $e_1, e_2, \ldots, e_N$ denoting the elementary flux modes, there exists a set of N non-negative coefficients w_i for every feasible flux vector $\mathbf{v}$ such that:

$$\mathbf{v} = \sum_{i=1}^{N} w_i e_i \tag{5.3}$$

Each elementary mode is minimal in its support: the set of non-zero fluxes in the vector.

Biologically, this means that an elementary mode represents a minimal subnetwork that still maintains a steady-state, or a set of balanced pathways. In other words, the removal of any reaction in an elementary flux mode leads to an imbalance, and the system being out of steady-state. Furthermore, the values of the fluxes in an elementary flux mode are fixed up to a multiplicative constant, so the ratios between these fluxes are fixed: to maintain a steady-state, these ratios must be maintained.

The cell's metabolism can choose among different strategies, such as respiring or fermenting a carbon source, through gene-regulation, effectively choosing between elementary modes and combinations thereof [131]. Additionally, actual cell states can usually be described as a combination of a small number of elementary flux modes [38]. Like flux balance analysis, elementary mode analysis can be extended to include resource allocation, while still conserving similar attributes [39]

The elementary flux modes of the example network (Figure 5.1) are shown in Figure 5.3.

5.3 A trajectory to optimality in a toy model

5.3.1 Dynamic FBA

Constraint-based models have been used to describe metabolic optimisation in a dynamic environment in two different ways. The static optimisation approach breaks dynamic FBA into a series of static FBA. The algorithm consists of iterating over time and at each step solving the FBA problem, then updating the external medium according to the exchange rates in the solution. This is the approach used to describe consumption patterns of E.coli in mixed media [10, 134]. This method makes the assumption that there is no lag in adaptation, and that the cell is purely acting on the current state of the environment with no memory and no predictions of the future.

The dynamic optimisation approach turns the individual steps of the static method into a single problem, and simultaneously optimises all time steps. This approach is much more computationally intensive, and hence does not easily scale. Nevertheless, it was applied by [125] to model diauxic shift, and by [97] to model circadian rhythms in cyanobacteria. Here the cells have full knowledge of the future, as it is the final biomass of the cell that is optimised. Although this introduces the possibility of lag, the trajectory of the cell is tied to the arbitrary end time

and time step introduced in the model.

5.3.2 The information problem

The canonical way of working with constraint-based models is through a global optimisation method. In this framework, the cell is able to know which point in the metabolic space to move to for any given external condition, and the shortest path to get there. This raises the question of how a cell could implement such an optimisation. The natural way of implementing this would be a global lookup table that would assign an EFM to each media condition. This would be more information than the cell could store, however, even if it used its entire genome to do so.

Consider, for example, the E.coli core constraint-based model. This model has about 46 million elementary flux modes [120]. The model has 102 reactions. Let's assume the cell had to store only a flux level — low, medium or high — rather than an actual flux value. In this case, it would need to store 2 bits of information for each reaction for each of the 46 million EFMs, in addition to a (negligible) description of their corresponding media. This makes over 4 billion values to store, for a model that only describes part of the E.coli metabolism. In comparison, the full E.coli genome only has 4.6 million base pairs [99]. A more sensible assumption is that part of the genome encodes an optimisation method, possibly a noisy one, that allows the cells to move through their metabolic state space in order to reach an optimal state, using only local information to make the changes. Here I describe one example of such a method.

5.3.3 Algorithm requirements

The method is based on a set of assumptions. First that the cell prefers to be in steady states, and that therefore it is travelling between elementary flux modes. I therefore define a cell's metabolic state as being its current flux mode. This assumption is mostly to simplify the description I will use of moving around a graph, however the algorithm could easily be modified to work in a continuous space. Although I do not explicitly include linear combinations of flux modes, there are areas in the graph in which all states are equally likely. In this condition, the algorithm will cycle through the states. In this case I consider that the cell is in a linear combination of states.

Second, I assume that moving from one state to another is costly, and that cells want to minimise this cost as much as possible. This minimisation is local, that is it will take the least costly option at each step, even if this is longer and possibly costlier in the long run. This is an assumption that can allow us to choose between two states that are equally beneficial in terms of the objective. The toy model is too simple to properly implement this constraint based only on stoichiometry, so I arbitrarily introduced costs to each reaction. The cost of moving to a new flux mode is then proportional to the change in cost of all reactions that are active in that flux mode. This can be thought of as the energy or enzyme cost per reaction, reminiscent of previous work [33, 134].

Finally, I assume that the cell has a specific objective in moving through the state space. I will consider that optimal use of the media without breaking steady state as the objective. Optimal use of the media is defined as having uptake fluxes as close as possible to their maximum rate. This breaks with the standard practice of considering a growth objective, but is justified when considering what cells usually are able to sense. The first sensing point of the external conditions, for the cell, is transport [11], whereas there is not much evidence that cells are able to sense their growth and use it as an objective.

5.3.4 Nearest-neighbour representation of metabolic states

I organise the elementary flux modes into a network. I create a sparse network by only connecting flux modes if they are similar, meaning that going from one to the other would not be a big change (and therefore, not a costly step), by creating a k-nearest neighbour representation. I choose k as small as possible such that the network is still a single connected component. This means that there is a path from any state to any other state, ensuring that the cell can always adapt to a new situation even if it is very different from its current state. The sparsity of the network is an asset in terms of information storage. The more edges there are in the network, the more space is needed to store it.

The algorithm for traversing the network is the following. A cell in a given state will be able to compare its state only to the neighbouring states. It will transition to one of its neighbours based on a combination of whether this improves the objective function, and the efficiency of the transition. If its own fitness metric is higher than that of any of its neighbours, the cell will simply transition to its current state.

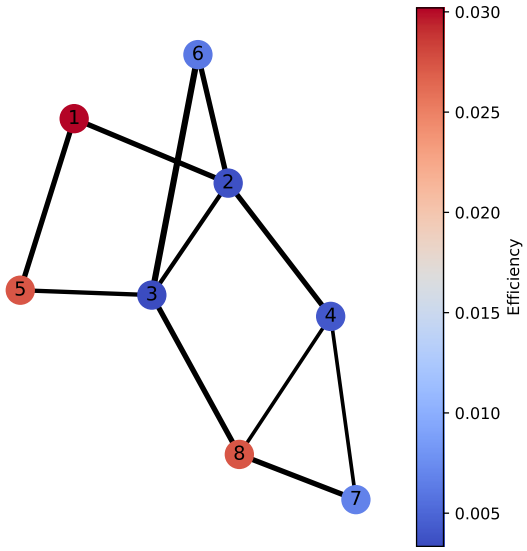


Figure 5.4: **Nearest-neighbour representation**. A nearest-neighbour model of the flux modes in Figure 5.3. Each was given an (arbitrary) efficiency, based on costs assigned to each reaction in the model. The thickness of the edges in between two nodes corresponds to their similarity

First, I apply this algorithm to the toy model in Figure 5.4. Starting in a random state, I apply a random media to the cell. I show the initial states, trajectory, and optimal state for a set of media with a known optimum. The algorithm always converges to the (global) optimum. Additionally, though this was not a requirement of the algorithm, it often goes through the shortest path to get there.

5.4 Encoding a larger network into a small space

The nearest-neighbour algorithm is possible in a toy model, because it is possible to create the full network. In actual constraint-based models, however, even enumerating flux modes is a prohibitively expensive task, and calculating the nearest neighbour graph would be computationally infeasible. We need a way to search the local space without knowing the neighbouring EFMs in advance. The next goal is therefore to find a representation of the flux modes that can be navigated with ease, but that does not require the pre-emptive computation all of the elementary flux modes.

The requirements of this representation are as follows. First the representation needs to be small enough to be stored into the genome of the corresponding organism. Ideally, this means that the representation is of a smaller dimension that the full model. Second, it must be possible

to translate from this smaller representation back into a corresponding elementary flux mode. Additionally, this translation should also not be too costly, so as to also be able to fit within the genome. Third, the information within that representation needs to be sufficient determine whether the objective is improving. In other words, by navigating through the space defined by this representation, one should be able to know whether the algorithm is getting closer or further away from the desired media.

I train an auto-encoder to create this representation(Figure 5.6). An auto-encoder is an unsupervised learning method with an encoder-decoder structure. This neural network is made up of two sub-parts, an encoder and a decoder, which have opposing roles. The encoder translates from a full 102-dimension flux mode to a 3-dimensional latent space representation. The decoder translates from a latent space representation back to a full flux mode. This sets a constraint on the latent space representation: although it is much smaller, it still needs to contain enough information to faithfully represent a full vector. I additionally train the encoding in an adversarial manner, in order to force the latent space representation to resemble a 3-dimensional Gaussian. The full network is therefore an adversarial auto-encoder [74, 133].

5.4.1 Training

I create a training set by generating the flux modes for a genome scale model of E.coli using `efmtool` [120]. This set contains too many flux modes to be kept in memory and worked with at once, so they are stored on disk and randomly sampled from the disk throughout the training period. I train for 5000 epochs with batch sizes of 16: I therefore sample only 80000 fluxes through out the training period. In contrast, there are several million flux modes in total. I therefore do not expect any flux modes to appear twice during training, so we do not apply any augmentation. I furthermore use flux modes from the same data set for testing. Although I do not explicitly check that the flux modes sampled for testing were not used during training, I nevertheless expect them to be distinct based on the sheer number of available modes and on the randomness of sampling.

The auto-encoder part of the network is trained to re-create its input, by minimising the mean-squared error between its input and its output. The training loss is shown in Figure 5.5a. The adversarial side of the network is trained to force the latent space into a 3-dimensional Gaussian. To do so: I train the discriminator to recognise the difference between the encoder's output and a sample from a 3-dimensional Gaussian with binary cross-entropy (Figure 5.5c).

Simultaneously, I train the auto-encoder to trick the discriminator, also using binary cross-entropy as loss (Figure 5.5b). The two are trained against each other until the discriminator is right about half the time, meaning the encoder's output is indistinguishable from a three dimensional Gaussian. At this point, both accuracies converge towards 0.5 (Figure 5.5d). I do not, at this point, enforce any requirements on the relationship between the latent space embedding and the media.

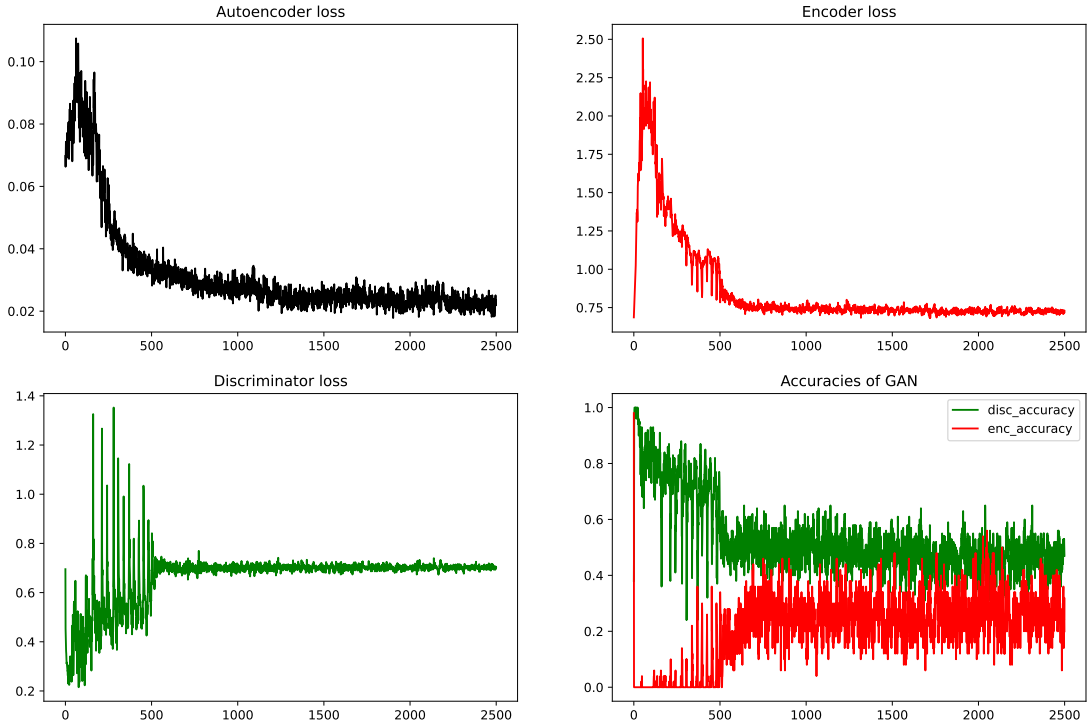


Figure 5.5: **Losses of the adversarial auto-encoder.** The auto-encoder loss (a) shows the auto-encoder's ability to recreate flux modes after encoding and decoding. It is a mean squared error. The encoder loss (b) shows how well the encoder is able to stay within the bounds of the Gaussian space that I impose. This is determined by how well it is able to trick the discriminator; it is a binary cross-entropy. The discriminator loss (c) is the other side of this. It is also a cross entropy, and I expect it to increase rather than decrease (the encoder gets better). (d) shows similar data to (b-c), but as accuracies, which have values between 0 and 1. The accuracy of the discriminator is the fraction of samples that it can accurately classify as real or fake. The accuracy of the encoder is the fraction of samples where it successfully tricks the discriminator: i.e. where the discriminator classifies a fake sample as being real. Ideally, they should converge to 0.5, meaning that the discriminator is deciding randomly whether a sample is real or generated.

5.4.2 Latent space embedding

Once the training has converged, that is that the loss is no longer decreasing, I check whether the latent space embedding satisfies my previous constraints. First, the latent space used in this case is very small: it is a three dimensional Gaussian and could in theory be represented

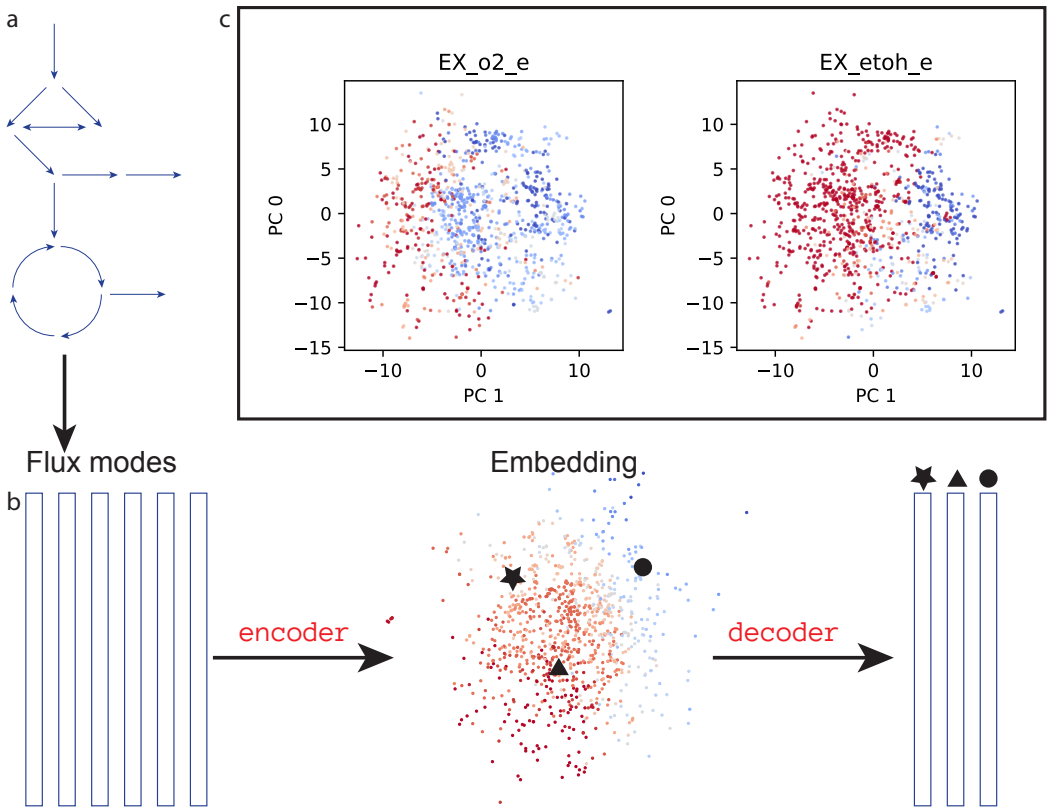


Figure 5.6: **Encoding with autoencoders.** (a) The elementary flux modes are obtained from the carbon-core model of E.coli using `efmtool`. These are the inputs to the auto-encoder. (b) The encoder translates a full elementary flux mode to a small vector in the latent space. Here I project a set of latent vectors along two of the three dimensions and colour according to oxygen use. The decoder samples from that space and translates back to a full-scale elementary flux mode. () The latent space naturally segregates according to key reactions in the model, in particular exchange reactions. Here took a sample of flux modes and encoded them into the latent space. I project these encodings onto the two main principal components of the sample, and colour according to oxygen uptake (EX_o2_e) and ethanol secretion (EX_etoh_e).

by the concentrations of only three molecules. For example, the nuclear concentration of three orthogonal transcription factors would in theory be enough to fully describe the latent space. The result could get even better by increasing the number of dimensions and this would still be feasible for the cell, as there are hundreds of transcription factors available.

We can also see that the latent space was spontaneously organised into different media types by plotting the flux value for different exchange reactions. In particular, we observe that respiratory and fermentative states are separated along one axis, while different sources of glucose are separated along another (Figure 5.6c). The general task of condensing flux modes into a smaller space reduces to the description of key exchanges. This suggests that the media-matching objective is feasible, and that once the network is optimised, its output contains enough information to recreate a full set of fluxes for all reactions in the cell. Furthermore, the

network converges to a low mean squared error (Figure 5.5a), meaning that the decoder is able to retrieve the initial flux mode from just its latent space representation. Finally, the network is easily small enough to fit into the genome.

5.5 Traversal of the latent space

I created an algorithm to traverse the latent space based on local information, using the decoder of the neural network as a model, and only information about exchanges. First, I define a target flux mode $\mathbf{o}$, sampled from the existing elementary flux modes. The encoded version of this flux mode is denoted as a red cross in Figure 5.7a. I then define the medium based on $\mathbf{o}$, by making constraining the exchanges fluxes such that the uptakes in $\mathbf{o}$ are maximal. I also constrain the secretion rates to match, as closely as possible, those in $\mathbf{o}$. The goal of the traversal is to retrieve the target vector $\mathbf{o}$ by matching its uptakes and secretions.

I then choose a starting $\mathbf{s}$ point from the sample that is different from $\mathbf{o}$. This starting point is embedded into the latent space as $\mathbf{s}_{latent}$ (coloured circles in Figure 5.7a). This is the latent variable. I then repeat the following steps:

- Decode the latent variable into a full (102 dimensional) flux vector $\mathbf{v}$
- Compute the distance between the exchange fluxes of $\mathbf{v}$, and those of $\mathbf{o}$: this is the gradient in the genome-scale space.
- Obtain the gradient in the latent space $\mathbf{g}_{latent}$by back-propagation through the network: the gradient at each layer is obtained based on the gradients at its outputs, recursively until it reaches the latent space.
- Move the latent variable in the direction of the gradient, scaled by the learning rate α: $\mathbf{s}_{latent} \leftarrow \mathbf{s}_{latent} + \alpha\mathbf{g}_{latent}$
- Continue until convergence: the gradient in the latent space converges to 0.

5.5.1 The algorithm converges to near-optimality

Figure 5.7a shows the trajectories for ten different starting points given the same optimal flux mode. Although they all begin in different areas of the latent space, they all converge to the

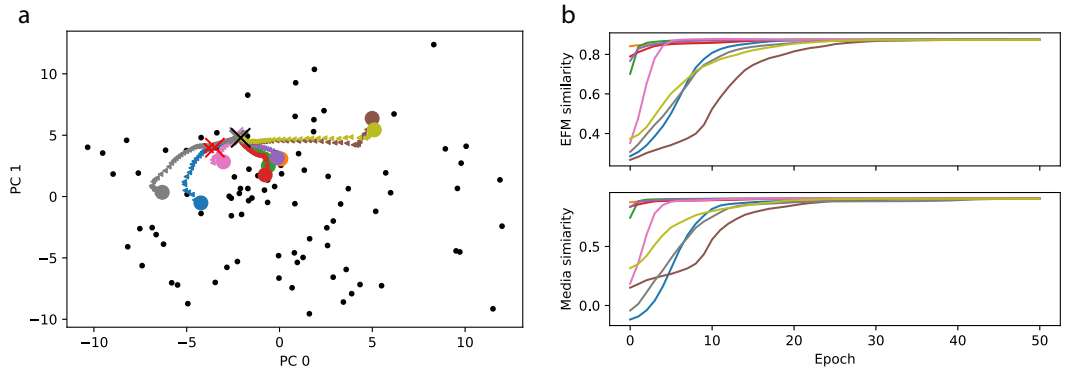


Figure 5.7: **Latent space navigation.** (a) Trajectories of latent space navigation projected onto the first two principal components. Ten different trajectories are shown. The starting points are shown as circles of colour; the intermediate steps of the trajectory are triangles in the same colour. All of the trajectories converge onto the same point, which is shown as a black cross. For comparison, the latent space representation of the target flux mode, from which I described the media, is shown as a large red cross. The smaller red cross beside it is the same vector after having gone through decoding and then another encoding. (b) Cosine similarity between the target flux mode and the current flux mode across the optimisation trajectories. The top plot shows the cosine similarities for the full flux mode, the bottom for only the exchange reactions with which the gradient is computed for navigation.

same point. I plot their cosine similarity to the optimal vector over time in Figure 5.7b, both using the full flux mode (top) and looking only at the media which is used in the loss (bottom). The cosine similarity is a normalised dot product between two vectors (Equation 5.4).

$$c(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}.\mathbf{y}}{\|x\| \, \|y\|} \tag{5.4}$$

I use the cosine similarity rather than the mean squared error because it is scale-invariant. Flux modes are only defined up to a multiplying constant, so this better captures the equivalency. In the media similarity (Figure 5.7b, bottom), we can see that the navigation eventually leads to a flux mode whose exchange reactions are close (up to a multiplying factor) to those of the target vector $\mathbf{o}$. In the EFM similarity (Figure 5.7b, top), we see that at the same time the navigation is getting close to the target flux mode across internal fluxes as well. This means that despite not using the internal fluxes as an input to the navigation, they are indirectly optimised to fit the target. All of the similarities increase over the course of the navigation, but at different speeds and with different dynamics. All of the trajectories in this example end at the same place in the latent space; none of them end up exactly at the target position$\mathbf{o}_{latent}$. Instead, they converge to a different point, even for those that were initially closer to the target and have to travel further off.

A first hypothesis to explain this is that this convergence point corresponds to the network's

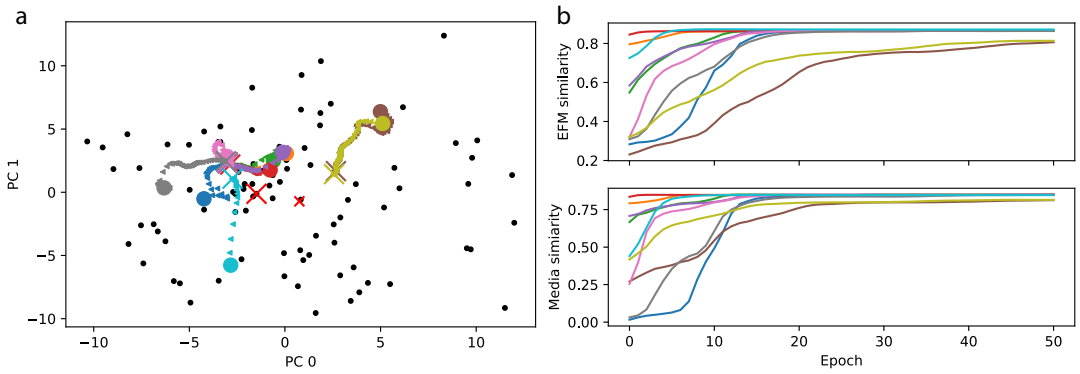


Figure 5.8: **Latent space navigation with multiple optima.** (a) The trajectories across latent space. Starting points are coloured circles, trajectories are scattered triangles, the end point of each trajectory is a cross in its corresponding colour. A large red cross shows the position of the target flux vector in latent space, whereas the small red cross shows the position of the same vector after decoding and encoding again. (b) Cosine similarity between the target flux mode and the actual flux mode across the optimisation trajectories. The top plot shows the cosine similarities for the full flux mode, the bottom for only the exchange reactions.

perception of the target $\mathbf{o}$. In other words, that because the neural network is imperfect, errors are introduced when a vector is encoded and then decoded again: $\mathbf{o} \leftarrow \mathbf{o} + \epsilon$. I plot the latent space representation of this new vector,$\mathbf{o} + \epsilon$, as a smaller red cross in Figure 5.7b. It does not correspond to the convergence point of the navigation algorithm. In fact, it is actually in the opposite direction as compared to $\mathbf{o}_{latent}$. As the same vector is repeatedly passed through the auto-encoder its latent space representation continues to diverge in the latent space, but it does not go towards the convergence point.

The convergence point's location is therefore inherently linked to the navigation process. Since the algorithms were run to convergence, until the gradients are near zero. This means that it has arrived at a locally optimal point in the latent space with regards to the loss that we defined. This suggests that the end point is not the target because the loss is imperfect: it only uses part of the information available. Nevertheless, the similarity metrics in Figure 5.7 show that even with incomplete knowledge it is possible to get close to the target. Biologically, this means that the cell does not need a sensor for every metabolic reaction in order to make a reasonable decision about how to adapt. In this case, sensing the exchange reactions is enough to at least reach local optimality.

5.5.2 Locally optimal end-point depends on the initial state

Running the algorithm on a different target vector, shows further proof that it only reaches locally optimal points. In Figure 5.8, I plot an example of navigation where different inputs reach different convergences. In this case, the starting point is essential in determining the end state, but the end states are almost equally good in terms of minimising loss, namely matching the exchange reactions to that of the target. The target $\mathbf{o}$ is even further away from its shifted version $\mathbf{o}+\epsilon$ in the latent space in this case. Still, neither coincides with any of the convergence points. We also see that the convergence points no longer overlap even when they are close. I hypothesised that the algorithm then leads to a locally optimal subspace of the latent space, rather than a single locally optimal point.

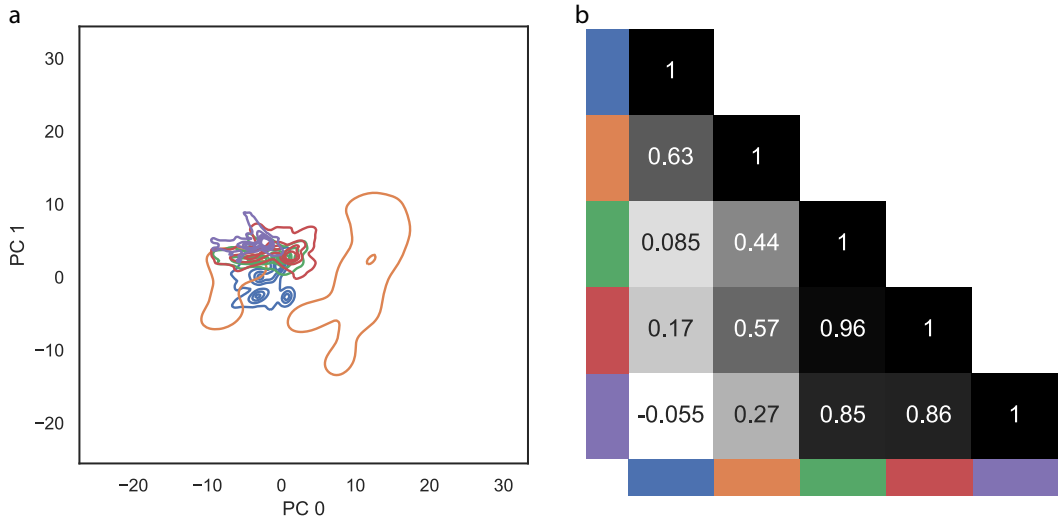


Figure 5.9: **Navigation moves to locally optimal areas of the latent space.** (a) I performed navigation through the latent space for 1000 normally distributed starting points, for five different media (sampled from the flux modes of the model). I plot the distribution of the convergence points when projects onto the principal components obtained from the embeddings of the flux mode samples. The trajectories end in spaces rather than specific points, and for some media these spaces can be large and disjoint. (b) Pairwise cosine similarity between the media in (a). The higher/darker the value, the closer the media are, and the closer their target points are expected to be.

Next I determine the characteristics that these locally optimal spaces can have. To do this, I apply the navigation to a set of randomly sampled points in the latent space, given five different target flux modes. Unlike previously, the starting points are not sampled from the flux modes and then embedded into the latent space. Instead, I sample directly from a three dimensional Gaussian into which I constrained the latent space through adversarial training. I sample 10,000 starting points from this distribution.

I set the targets used for the loss by sampling five flux modes. Once again, I only use their exchange reaction values in the navigation algorithm; all internal fluxes are left unset. I run the navigation algorithm for each starting point and target combination, and plot the results in Figure 5.9. The distribution of convergence points varies across targets: some are localised to a small region as expected (green, purple, red), while others are far more widespread and even multi-modal (orange). When two targets have similar exchange fluxes (Figure 5.9b), their convergence regions have a higher overlap. This suggests a navigation from region to region rather than point to point across the space.

5.6 Discussion

Constraint-based models are an elegant representation of the metabolic range of a cell. They describe the many possible metabolic states that a cell can be in, without requiring specific biological constants beyond stoichiometric coefficients. Classic analysis of these models often requires strong additional assumptions, and describes global optimisation of the cell's metabolism to a single state that is only dependent on the cell's current environment. Instead, I described the adaptation of a cell to a new environment as a navigation through a small, easily described space, using only local information.

In this proof-of-concept, I set a target flux mode rather than a target medium. In doing so, I make the implicit assumption that this target is optimal in the medium described by its exchange fluxes. This is not necessarily true. For any given set of exchange reactions, there can be several different fluxes modes. Depending on the chosen objective — growth rate, ATP production, or some combination — one or several of these flux modes will be optimal. However, as I have shown that the algorithm can navigate towards any flux mode, navigating towards a specific optimum is straight-forward. Once an objective function is chosen, one can simply set its optima as targets when navigating through the latent space. With little extra effort, the gradients in the space could also be computed directly from the objective.

The different parts of the proposed navigation system have biological analogies. The cell senses through its exchanges, and measures the direction of any imbalance when it is moved to a new medium. This corresponds to how we navigate the space. The sensing begins at the membrane and eventually finds its way to the gene regulatory apparatus of the cell. This is the back-propagation step, where the gradients are converted at the media-level to gradients at the

latent space level.

Although the regulation apparatus is far more complex than the three dimensional Gaussian used here, it is clear that it is is not able to directly describe the infinite space of possible media conditions the cell can be in. There needs to be significant compression for the cell from the media to a small set of actionable responses. I emulate this compression by making the latent space small. As we have seen, it is still able to encode, at least in large strokes, variations and gradients in the media conditions.

Once a strategy is chosen, the regulation needs to be able to make a correspondence between the new regulatory state and the full metabolic state of the cell. This is analogous to a decompression. I modelled this with the decoder, which transforms data from the 3-dimensional latent space into a 102-dimensional flux mode. The new state will then have an effect on the sensing, and the process starts anew.

Local algorithms give rise to cell-to-cell variability. There are two sources of variability in the proposed algorithm. The main one is the initial position of the cell in the latent space. Unlike classic flux-balance based methods, the initial variability in the population of cells has an effect on the variability of the population after adaptation to the new environment. In the navigation, as in experiments, the starting point matters: two cells can end up in different local optima despite being put into the same environment because their starting states were different. Rather than a global optimisation with full knowledge of the system, I created a navigation by steps with only partial information used for navigation. The effect of this is condition-dependent, with some conditions having several equally good although generally different strategies (e.g.Figure 5.8; Figure 5.9a, orange), and others simply clustering around nearby points (e.g.Figure 5.7; Figure 5.9a, blue).

The second source of variability is linked to noise in the encoding and decoding. A flux vector that is passed through the auto-encoder twice does not end up with the same position in the latent space. Although this could be mitigated by fine-tuning the training parameters of the neural network, this can also be seen as a model of the intrinsic noise in both sensing and gene expression. In my simulations as in experimental data, even the strategies that have a single optimum manifest as a distribution of points, albeit a narrow one (e.g.Figure 5.9a, purple). None of the simulated cells end up in the optimal state. The sub-optimality of metabolism is often ascribed an evolutionary advantage. Often, it is explained as bet-hedging [6, 15, 124], where the cell gives up some amount of fitness in order to improve adaptability to change or to

fit a niche within a population [24]. Other times, it is seen as a trade-off between metabolism and other components of cellular function [77, 82, 97, 110, 130] I have shown that it can also simply be linked to a local, noisy, optimisation of state.

The auto-encoder that I trained is by no means the only possible description of this navigation. It was mostly chosen for the simplicity of the training and the possibility of constraining the latent space to a distribution that would be easy to sample from. This network describes the type of mechanism, and shows that this kind of navigation is possible. For better interpretability, it would be sensible to choose an explainable model: one where the weights of the network have a direct biological correspondence. This could mean using the metabolic network itself as an underlying architecture using Graph Neural Networks [137], or using the discriminator to force a direct correspondence between the latent space and known transcription factors for instance. This type of mechanistic modelling based on back-propagation has been used previously to model the evolution of phenotypic variability in development [128] and to predict the effect of drug therapies on melanoma cancer cells [135].

Altogether, I have given a new conceptual framework for describing metabolic adaptation. I have shown that the complexity of the genome-scale network can be reduced to a simple yet navigable underlying latent space through generalisable methods which can be applied to other biological data with minimal conceptual changes.

Chapter 6

Conclusion

Budding yeast has been a model of eukaryotic cells for over a century, and the study of these cells has been fundamental to advances across the field of biology [56, 71]. By observing these cells in single-cell time-lapse microscopy, we generate rich data on their response to sudden environmental changes, many of which have analogies in other organisms. The main goal of this thesis was to create image analysis methods to improve the description of the various transient processes occurring in yeast cells as a response to sudden glucose starvation. The methods developed improve accessibility to data, and expand the range of quantitative data that can be extracted from bright field images. These techniques are not specific to yeasts, and could be used on other types of data.

In chapter 2, we developed a framework for transforming the raw data obtained by the microscope into useable, single-cell, image information. A texture-based segmentation of traps in micro-fluidics images was proposed, along with mitigations to common problems. I then contributed to a cell segmentation, tracking, and lineage assignment pipeline to obtain full time-lapses of single-cells from the images. This was used for automated extraction of both fluorescence and morphology data at a single-cell level. The pipeline is fully automated, modular and easily extensible. It is also suited to real-time analysis, and could be used to generate data on-the-fly in control-loops to define key points in the experiment. I showed how we store the results and what considerations are to be taken into account to ensure that they are easily accessible and shareable. The speed-up of the pipeline compared to previous work is significant.

The amounts of data generated by high-throughput microscopy, and expected in deep learning

methods, are unmanageably large. This kind of data pre-processing is an often overlooked yet primordial part of analysis, necessary before one can gain any quantitative biological insight. By speeding it up and making it more transparent, this work improves the return on investment of running time-lapse experiments in the first place. It is important to note that the methods used in this pipeline were built specifically for ALCATRAS micro-fluidics devices or similar. In particular, the trap and cell segmentation and tracking rely on a relatively sparse array of cells, kept fixed throughout the experiment in traps. The abstract description remains relevant, however, to other types of data. We make heavy use of this pipeline in the background of the rest of this thesis to create training data for training multiple neural networks. In future work, these trained neural networks should be included as a part of the pipeline.

In chapter 3, I looked at two spatial aspects of the cellular stress response: the nuclear localisation of transcription factors and the endocytosis of hexose transporters. To do this in a quantitative way, I first trained two U-Net neural networks to recognise the nucleus and the vacuole from bright field images. I showed that this could be done with little training and without manual annotation, using images of fluorescent markers to generate training data. With the predictions of the neural networks, I quantified the localisation dynamics of several different transcription factors in response to a downshift in glucose, replicating existing results [36]. This transient response occurred in minutes, giving an indication of how quickly the cell can sense its new external environment. I then quantified the endocytosis of sub-optimal hexose transporters, and found that low affinity hexose transporters undergo endocytosis approximately two hours after the stress is induced. This delay between transcription factor response and membrane re-organisation has not yet been described to the best of our knowledge.

The reasons behind this delay remain unexplained, and our current hypotheses need to be verified experimentally. This would likely require the imaging of a transcription factor and a hexose transporter in the same cell. I have not yet shown that the neural networks are able to accurately predict both the nucleus and the vacuole within the same cell. The next step for this project is therefore to validate the predictions on a strain tagged with both a vacuolar marker and a nuclear marker. Following this, the method may be potentially applied to any sub-cellular compartment for which there is a marker, with theoretical limits on the performance linked to the signal-to-noise ratio afforded by the fluorescent marker [88]. With specially curated data, it is not infeasible that the pipeline in chapter 2 will include sub-cellular segmentation of most major organelles from bright field images in the future.

One hypothesis as to how the cell determines the timing of the various aspects of its stress response is related to the cell's cycle. In chapter 4, I describe and evaluate several methods of predicting key cell cycle events and phases in label-free manner. I first showed that it is possible to accurately determine the point of cytokinesis from the relative growth rate of mother and bud during a cell cycle. This method is robust across different carbon sources and during transition phases. I then described the evaluation of two classifiers to obtain cell cycle information from snapshots of bright field images. We were unsuccessful in reliably training these classifiers, suggesting either that timing information is necessary, or that our labelling was not precise enough. By directly predicting the fluorescence of a cell cycle marker from bright field, I showed that there is some cell cycle information included in a single snapshot of cells, without added timing information. I compared two networks for this task, and showed that the network's ability to accurately predict the location of the fluorescent marker did not translate to its ability to recapitulate the cell cycle information. Nevertheless, I obtained cell cycle duration estimates that correspond well to those obtained with the cell cycle marker, and showed on examples that our predicted fluorescence time series was highly correlated with the real fluorescence.

The results raise an important question of how the network knows cell cycle information from bright field images. This annotation cannot be done manually; aside from a few important event such as budding and cytokinesis, there is little known information in bright field that corresponds to cell cycle stage. Future work on this project should attempt to uncover what features of the images the network is using to obtain its result, and whether these features are interpretable as known morphological features of the cells.

The applications of label-free prediction of cell cycle are far-ranging. In the context of stress, one could now verify whether the delay between transcription factor localisation and endocytosis described in chapter 3 can be explained by the cell waiting to reach a specific cell cycle phase; for example whether it is going to G1 so that it can arrest in G0.

Overall, the results of chapter 4 are a promising indication of how much unexpected information lies in bright field images that can be extracted using deep learning.

In chapter 5 I model the change in metabolic state of a cell mirroring a change in environment. I base the model on elementary flux modes, basis vectors of constraint-based models of metabolism. I showed how a large complex description of metabolism can be reduced to a small, easily tractable space. I further showed that navigation through such a space is possible,

and that each point in this space can be expanded back to a full metabolic description. This model is applicable to other kinds of complex biological networks.

Some improvements to the model are still required to move beyond a proof-of-concept. I do not, in this thesis, establish a method for trajectories to move out of local optima; which could be included as additional noise in the navigation. I constrain the small space to fit a certain shape, but do not guide this shape to be maximally informative for the navigation task, although we do show that some reactions appear in a gradient within this space, aiding navigation. Nevertheless, the model represents a good conceptual framework to describe how cells are able to reliably adapt when the metabolic decision space is so complex.

Put together, the methods described in this thesis could turn into a powerful tool. I envisage a framework that accurately detects cells, identifies sub-cellular compartments, and determines cell cycle stages all from bright field. This would leave the spectral space of fluorescent markers open to quantitatively determine how cells response to stress at several different sensing, decision, and actuation points. The framework could then map these responses to a low-dimensional space, sufficiently complete to accurately recapitulate all of the information in fluorescence, but also sufficiently simple to make clear the strategies that cells use to make decisions.

The physiology of the cell remains a complex system of subsystems, each evolving over time and space. As high-throughput experimental methods become available to query these, computational methods must follow that are able to contend with the size and complexity of the data produced. In this thesis, I have shown that deep learning is one such method. By adopting these tools, we can transform the complexity of the data, which is difficult for us to interpret, into quantitative descriptions and conceptual analogies of biological phenomena, upon which we can build knowledge.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] Martí Aldea, Kirsten Jenkins, and Attila Csikász-Nagy. Growth Rate as a Direct Regulator of the Start Network to Set Cell Size. Frontiers in Cell and Developmental Biology, 5:57, May 2017.

[3] Chris Allan, Jean-Marie Burel, Josh Moore, Colin Blackburn, Melissa Linkert, Scott Loynton, Donald MacDonald, William J. Moore, Carlos Neves, Andrew Patterson, Michael Porter, Aleksandra Tarkowska, Brian Loranger, Jerome Avondo, Ingvar Lagerstedt, Luca Lianas, Simone Leo, Katherine Hands, Ron T. Hay, Ardan Patwardhan, Christoph Best, Gerard J. Kleywegt, Gianluigi Zanetti, and Jason R. Swedlow. OMERO: Flexible, model-driven data management for experimental biology. Nature Methods, 9(3):245–253, March 2012.

[4] Orlando Argüello-Miranda, Yanjie Liu, N. Ezgi Wood, Piya Kositangool, and Andreas Doncic. Integration of Multiple Metabolic Signals Determines Cell Fate Prior to Commitment. Molecular Cell, 71(5):733–744.e11, September 2018.

[5] Theo Aspert, Didier Hentsch, and Gilles Charvin. DetecDiv, a deep-learning platform for automated cell division tracking and replicative lifespan analysis, October 2021.

[6] Laura E. Bagamery, Quincey A. Justman, Ethan C. Garner, and Andrew W. Murray. A Putative Bet-Hedging Strategy Buffers Budding Yeast against Environmental Instability. Current Biology, 30(23):4563–4578.e4, December 2020.

[7] Elco Bakker, Peter S. Swain, and Matthew M. Crane. Morphologically constrained and data informed cell segmentation of budding yeast. Bioinformatics (Oxford, England), 34(1):88–96, January 2018.

[8] LaKisha Barrett, Marianna Orlova, Marcin Maziarz, and Sergei Kuchin. Protein Kinase A Contributes to the Negative Control of Snf1 Protein Kinase in Saccharomyces cerevisiae. Eukaryotic Cell, 11(2):119–128, February 2012.

[9] Lisa L. Baumbach, Gary S. Stein, and Janet L. Stein. Regulation of human histone gene expression: Transcriptional and posttranscriptional control in the coupling of histone messenger RNA stability with DNA replication. Biochemistry, 26(19):6178–6187, September 1987.

[10] Q. K. Beg, A. Vazquez, J. Ernst, M. A. de Menezes, Z. Bar-Joseph, A.-L. Barabási, and Z. N. Oltvai. Intracellular crowding defines the mode and sequence of substrate uptake by Escherichia coli and constrains its metabolic activity. Proceedings of the National Academy of Sciences, 104(31):12663–12668, July 2007.

[11] James R Broach. Nutritional Control of Growth and Development in Yeast. Genetics, 192(1):73–105, September 2012.

[12] Stefano Busti, Paola Coccetti, Lilia Alberghina, and Marco Vanoni. Glucose Signaling-Mediated Coordination of Cell Growth and Cell Cycle in Saccharomyces Cerevisiae. Sensors, 10(6):6195–6240, June 2010.

[13] Fabrice Caudron and Yves Barral. Septins and the Lateral Compartmentalization of Eukaryotic Membranes. Developmental Cell, 16(4):493–506, April 2009.

[14] Helen C. Causton, Bing Ren, Sang Seok Koh, Christopher T. Harbison, Elenita Kanin, Ezra G. Jennings, Tong Ihn Lee, Heather L. True, Eric S. Lander, and Richard A. Young. Remodeling of Yeast Genome Expression in Response to Environmental Changes. Molecular Biology of the Cell, 12(2):323–337, February 2001.

[15] Bram Cerulus, Abbas Jariani, Gemma Perez-Samper, Lieselotte Vermeersch, Julian Mj Pietsch, Matthew M. Crane, Aaron M. New, Brigida Gallone, Miguel Roncoroni, Maria C. Dzialo, Sander K. Govers, Jhana O. Hendrickx, Eva Galle, Maarten Coomans, Pieter Berden, Sara Verbandt, Peter S. Swain, and Kevin J. Verstrepen. Transition between fermentation and respiration determines history-dependent behavior in fluctuating carbon sources. eLife, 7:e39234, October 2018.

[16] Lucas von Chamier, Romain F. Laine, Johanna Jukkala, Christoph Spahn, Daniel Krentzel, Elias Nehme, Martina Lerche, Sara Hernández-Pérez, Pieta K. Mattila, Eleni Karinou, Séamus Holden, Ahmet Can Solak, Alexander Krull, Tim-Oliver Buchholz, Martin L. Jones, Loïc A. Royer, Christophe Leterrier, Yoav Shechtman, Florian Jug, Mike Heilemann, Guillaume Jacquemet, and Ricardo Henriques. Democratising deep learning for microscopy with ZeroCostDL4Mic. Nature Communications, 12(1):2276, April 2021.

[17] Gilles Charvin, Frederick R. Cross, and Eric D. Siggia. A Microfluidic Device for Temporally Controlled Gene Expression and Long-Term Fluorescent Imaging in Unperturbed Dividing Yeast Cells. PLoS ONE, 3(1):e1468, January 2008.

[18] Gilles Charvin, Catherine Oikonomou, Eric D. Siggia, and Frederick R. Cross. Origin of Irreversibility of Cell Cycle Start in Budding Yeast. PLoS Biology, 8(1):e1000284, January 2010.

[19] Travers Ching, Daniel S. Himmelstein, Brett K. Beaulieu-Jones, Alexandr A. Kalinin, Brian T. Do, Gregory P. Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M. Hoffman, Wei Xie, Gail L. Rosen, Benjamin J. Lengerich, Johnny Israeli, Jack Lanchantin, Stephen Woloszynek, Anne E. Carpenter, Avanti Shrikumar, Jinbo Xu, Evan M. Cofer, Christopher A. Lavender, Srinivas C. Turaga, Amr M. Alexandari, Zhiyong Lu, David J. Harris, Dave DeCaprio, Yanjun Qi, Anshul Kundaje, Yifan Peng, Laura K. Wiley, Marwin H. S. Segler, Simina M. Boca, S. Joshua Swamidass, Austin Huang, Anthony Gitter, and Casey S. Greene. Opportunities and obstacles for deep learning in biology and medicine. Journal of The Royal Society Interface, 15(141):20170387, April 2018.

[20] François Chollet et al. Keras, 2015.

[21] Eric M. Christiansen, Samuel J. Yang, D. Michael Ando, Ashkan Javaherian, Gaia Skibinski, Scott Lipnick, Elliot Mount, Alison O'Neil, Kevan Shah, Alicia K. Lee, Piyush Goyal, William Fedus, Ryan Poplin, Andre Esteva, Marc Berndl, Lee L. Rubin, Philip Nelson, and Steven Finkbeiner. In Silico Labeling: Predicting Fluorescent Labels in Unlabeled Images. Cell, 173(3):792–803.e19, April 2018.

[22] Michaela Conrad, Joep Schothorst, Harish Nag Kankipati, Griet Van Zeebroeck, Marta Rubio-Texeira, and Johan M. Thevelein. Nutrient sensing and signaling in the yeast Saccharomyces cerevisiae. FEMS Microbiology Reviews, 38(2):254–299, March 2014.

[23] Matthew M. Crane, Ivan B. N. Clark, Elco Bakker, Stewart Smith, and Peter S. Swain. A Microfluidic System for Studying Ageing and Dynamic Single-Cell Responses in Budding Yeast. PLOS ONE, 9(6):e100042, June 2014.

[24] Daniele De Martino, Anna MC Andersson, Tobias Bergmiller, Călin C. Guet, and Gašper Tkačik. Statistical mechanics for metabolic networks during steady state growth. Nature Communications, 9(1):2988, July 2018.

[25] Veerle De Wever, Wolfgang Reiter, Annalisa Ballarini, Gustav Ammerer, and Cécile Brocard. A dual role for PP1 in shaping the Msn2-dependent transcriptional response to glucose starvation. The EMBO Journal, 24(23):4115–4123, December 2005.

[26] Nicola Dietler, Matthias Minder, Vojislav Gligorovski, Augoustina Maria Economou, Denis Alain Henri Lucien Joly, Ahmad Sadeghi, Chun Hei Michael Chan, Mateusz Koziński, Martin Weigert, Anne-Florence Bitbol, and Sahand Jamal Rahi. A convolutional neural network segments yeast microscopy images with high accuracy. Nature Communications, 11(1):5723, November 2020.

[27] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. arXiv:1603.07285 [cs, stat], January 2018.

[28] Thorsten Falk, Dominic Mai, Robert Bensch, Özgün Çiçek, Ahmed Abdulkadir, Yassine Marrakchi, Anton Böhm, Jan Deubner, Zoe Jäckel, Katharina Seiwald, Alexander Dovzhenko, Olaf Tietz, Cristina Dal Bosco, Sean Walsh, Deniz Saltukoglu, Tuan Leng Tay, Marco Prinz, Klaus Palme, Matias Simons, Ilka Diester, Thomas Brox, and Olaf Ronneberger. U-Net: Deep learning for cell counting, detection, and morphometry. Nature Methods, 16(1):67–70, January 2019.

[29] Francisco Ferrezuelo, Neus Colomina, Alida Palmisano, Eloi Garí, Carme Gallego, Attila Csikász-Nagy, and Martí Aldea. The critical size is set at a single-cell level by growth rate to attain homeostasis and adaptation. Nature Communications, 3(1):1012, August 2012.

[30] Karin M. Flick, Nathalie Spielewoy, Tatyana I. Kalashnikova, Marisela Guaderrama, Qianzheng Zhu, Hui-Chu Chang, and Curt Wittenberg. Grr1-dependent Inactivation of Mth1 Mediates Glucose-induced Dissociation of Rgt1 from HXT Gene Promoters. Molecular Biology of the Cell, 14(8):3230–3241, August 2003.

[31] Cecilia Garmendia-Torres, Olivier Tassy, Audrey Matifas, Nacho Molina, and Gilles Charvin. Multiple inputs ensure yeast cell size homeostasis during cell cycle progression. eLife, 7:e34025, July 2018.

[32] Mehran Ghafari, Justin Clark, Hao-Bo Guo, Ruofan Yu, Yu Sun, Weiwei Dang, and Hong Qin. Complementary performances of convolutional and capsule neural networks on classifying microfluidic images of dividing yeast cells. PLOS ONE, 16(3):e0246988, March 2021.

[33] Anne Goelzer, Vincent Fromion, and Gérard Scorletti. Cell design in bacteria as a convex optimization problem. Automatica, 47(6):1210–1218, June 2011.

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, November 2016.

[35] Andrew Gordon, Alejandro Colman-Lerner, Tina E. Chin, Kirsten R. Benjamin, Richard C. Yu, and Roger Brent. Single-cell quantification of molecules and rates using open-source microscope-based cytometry. Nature Methods, 4(2):175–181, February 2007.

[36] Alejandro A. Granados, Julian M. J. Pietsch, Sarah A. Cepeda-Humerez, Iseabail L. Farquhar, Gašper Tkačik, and Peter S. Swain. Distributed and dynamic intracellular organization of extracellular information. Proceedings of the National Academy of Sciences, 115(23):6088–6093, June 2018.

[37] Joseph V. Gray, Gregory A. Petsko, Gerald C. Johnston, Dagmar Ringe, Richard A. Singer, and Margaret Werner-Washburne. "Sleeping Beauty": Quiescence in Saccharomyces cerevisiae. Microbiology and Molecular Biology Reviews, 68(2):187–206, June 2004.

[38] Daan H. de Groot, Coco van Boxtel, Robert Planqué, Frank J. Bruggeman, and Bas Teusink. The number of active metabolic pathways is bounded by the number of cellular constraints at maximal metabolic rates. PLOS Computational Biology, 15(3):e1006858, March 2019.

[39] Daan H. de Groot, Julia Lischke, Riccardo Muolo, Robert Planqué, Frank J. Bruggeman, and Bas Teusink. The common message of constraint-based optimization approaches: Overflow metabolism is caused by two growth-limiting constraints. Cellular and Molecular Life Sciences, 77(3):441–453, February 2020.

[40] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. Nature, 585(7825):357–362, September 2020.

[41] Leland H. Hartwell, Joseph Culotti, John R. Pringle, and Brian J. Reid. Genetic Control of the Cell Division Cycle in Yeast. Science, 183(4120):46–51, 1974.

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, June 2016.

[43] Larissa Heinrich, Davis Bennett, David Ackerman, Woohyun Park, John Bogovic, Nils Eckstein, Alyson Petruncio, Jody Clements, Song Pang, C. Shan Xu, Jan Funke, Wyatt Korff, Harald F. Hess, Jennifer Lippincott-Schwartz, Stephan Saalfeld, and Aubrey V. Weigel. Whole-cell organelle segmentation in volume electron microscopy. Nature, pages 1–6, October 2021.

[44] Larissa Heinrich, Jan Funke, Constantin Pape, Juan Nunez-Iglesias, and Stephan Saalfeld. Synaptic Cleft Segmentation in Non-isotropic Volume Electron Microscopy of the Complete Drosophila Brain. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger, editors, Medical Image Computing and Computer Assisted Intervention – MICCAI 2018, Lecture Notes in Computer Science, pages 317–325, Cham, 2018. Springer International Publishing.

[45] Junie Hovsepian, Quentin Defenouillère, Véronique Albanèse, Libuše Váchová, Camille Garcia, Zdena Palková, and Sébastien Léon. Multilevel regulation of an α-arrestin by glucose depletion controls hexose transporter endocytosis. The Journal of Cell Biology, 216(6):1811–1831, June 2017.

[46] Alexandre Huber, Bernd Bodenmiller, Aino Uotila, Michael Stahl, Stefanie Wanka, Bertran Gerrits, Ruedi Aebersold, and Robbie Loewith. Characterization of the rapamycin-sensitive phosphoproteome reveals that Sch9 is a central coordinator of protein synthesis. Genes & Development, 23(16):1929–1943, August 2009.

[47] Alexandre Huber, Sarah L French, Hille Tekotte, Seda Yerlikaya, Michael Stahl, Mariya P Perepelkina, Mike Tyers, Jacques Rougemont, Ann L Beyer, and Robbie Loewith. Sch9

regulates ribosome biogenesis via Stb3, Dot6 and Tod6 and the histone deacetylase complex RPD3L. The EMBO Journal, 30(15):3052–3064, July 2011.

[48] James E Hughes Hallett, Xiangxia Luo, and Andrew P Capaldi. State Transitions in the TORC1 Signaling Pathway and Information Processing in Saccharomyces cerevisiae. Genetics, 198(2):773–786, October 2014.

[49] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5967–5976, July 2017.

[50] Shruti Jadon. A survey of loss functions for semantic segmentation. In 2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), pages 1–7, Via del Mar, Chile, October 2020. IEEE.

[51] Amy Johnson and Jan M Skotheim. Start and the restriction point. Current Opinion in Cell Biology, 25(6):717–723, December 2013.

[52] P. Jorgensen. A dynamic transcriptional network communicates growth potential to ribosome synthesis and critical cell size. Genes & Development, 18(20):2491–2505, October 2004.

[53] Paul Jorgensen and Mike Tyers. How Cells Coordinate Growth and Division. Current Biology, 14(23):R1014–R1027, December 2004.

[54] David Jouandot, Adhiraj Roy, and Jeong-Ho Kim. Functional dissection of the glucose signaling pathways that regulate the yeast glucose transporter gene (HXT) repressor Rgt1. Journal of Cellular Biochemistry, 112(11):3268–3275, November 2011.

[55] Aneta Kaniak, Zhixiong Xue, Daniel Macool, Jeong-Ho Kim, and Mark Johnston. Regulatory Network Connecting Two Glucose Signal Transduction Pathways in Saccharomyces cerevisiae. Eukaryotic Cell, 3(1):221–231, February 2004.

[56] Hiren Karathia, Ester Vilaprinyo, Albert Sorribas, and Rui Alves. Saccharomyces cerevisiae as a Model Organism: A Comparative Study. PLoS ONE, 6(2):e16015, February 2011.

[57] Ömur Kayikci and Jens Nielsen. Glucose repression in Saccharomyces cerevisiae. FEMS Yeast Research, 15(6):fov068, September 2015.

[58] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. ICLR, 2015.

[59] Steffen Klamt, Georg Regensburger, Matthias P. Gerstl, Christian Jungreuthmayer, Stefan Schuster, Radhakrishnan Mahadevan, Jürgen Zanghellini, and Stefan Müller. From elementary flux modes to elementary flux vectors: Metabolic pathway analysis with arbitrary linear flux constraints. PLOS Computational Biology, 13(4):e1005409, April 2017.

[60] Hirofumi Kobayashi, Keith C. Cheveralls, Manuel D. Leonetti, and Loic A. Royer. Self-Supervised Deep-Learning Encodes High-Resolution Features of Protein Subcellular Localization. bioRxiv, page 2021.03.29.437595, March 2021.

[61] Quincey Koziol and Dana Robinson. HDF5. Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA (United States), 2018.

[62] Oren Z Kraus, Ben T Grys, Jimmy Ba, Yolanda Chong, Brendan J Frey, Charles Boone, and Brenda J Andrews. Automated analysis of high-content microscopy data with deep learning. Molecular Systems Biology, 13(4):924, April 2017.

[63] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. Noise2Void - Learning Denoising From Single Noisy Images. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2129–2137, 2019.

[64] Joseph Kunkel, Xiangxia Luo, and Andrew P. Capaldi. Integrated TORC1 and PKA signaling control the temporal activation of glucose-induced gene expression in yeast. Nature Communications, 10(1):3558, December 2019.

[65] Romain F. Laine, Guillaume Jacquemet, and Alexander Krull. Imaging in focus: An introduction to denoising bioimages in the era of deep learning. The International Journal of Biochemistry & Cell Biology, 140:106077, November 2021.

[66] Ricardo M. Leitao and Douglas R. Kellogg. The duration of mitosis and daughter cell size are modulated by nutrients in budding yeast. Journal of Cell Biology, 216(11):3463–3470, September 2017.

[67] Daniel J. Lew and Steven I. Reed. Cell cycle control of morphogenesis in budding yeast. Current Opinion in Genetics & Development, 5(1):17–23, February 1995.

[68] Mei Kee Lim, Wee Leng Siew, Jin Zhao, Ywee Chieh Tay, Edwin Ang, and Norbert Lehming. Galactose induction of the GAL1 gene requires conditional degradation of the Mig2 repressor. The Biochemical Journal, 435(3):641–649, May 2011.

[69] Yihan Lin, Chang Ho Sohn, Chiraj K. Dalal, Long Cai, and Michael B. Elowitz. Combinatorial gene regulation by modulation of relative pulse timing. Nature, 527(7576):54–58, November 2015.

[70] J. Lippincott, K.B. Shannon, W. Shou, R.J. Deshaies, and R. Li. The Tem1 small GTPase controls actomyosin and septin dynamics during cytokinesis. Journal of Cell Science, 114(7):1379–1386, April 2001.

[71] Gianni Liti. The fascinating and secret wild life of the budding yeast S. cerevisiae. eLife, 4:e05835, March 2015.

[72] Vicent Llopis-Torregrosa, Alba Ferri-Blázquez, Anna Adam-Artigues, Emilie Deffontaines, G. Paul H. van Heusden, and Lynne Yenush. Regulation of the Yeast Hxt6 Hexose Transporter by the Rod1 α-Arrestin, the Snf1 Protein Kinase, and the Bmh2 14-3-3 Protein. The Journal of Biological Chemistry, 291(29):14973–14985, July 2016.

[73] Feixiao Long. Microscopy cell nuclei segmentation with enhanced U-Net. BMC Bioinformatics, 21(1):8, January 2020.

[74] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial Autoencoders. arXiv:1511.05644 [cs], May 2016.

[75] Erick Moen, Dylan Bannon, Takamasa Kudo, William Graf, Markus Covert, and David Van Valen. Deep learning for cellular image analysis. Nature Methods, pages 1–14, May 2019.

[76] Luis Fernando Montaño-Gutierrez, Marc Sturrock, Iseabail Farquhar, Kevin Correia, Vahid Shahrezaei, and Peter S. Swain. A push-pull system of repressors matches levels of glucose transporters to extracellular glucose in budding yeast, April 2021.

[77] Matteo Mori, Terence Hwa, Olivier C. Martin, Andrea De Martino, and Enzo Marinari. Constrained Allocation Flux Balance Analysis. PLOS Computational Biology, 12(6):e1004913, June 2016.

[78] Douglas B. Murphy and Michael W. Davidson. Fundamentals of Light Microscopy and Electronic Imaging. John Wiley & Sons, August 2012.

[79] Raffaele Nicastro, Farida Tripodi, Marco Gaggini, Andrea Castoldi, Veronica Reghellin, Simona Nonnis, Gabriella Tedeschi, and Paola Coccetti. Snf1 Phosphorylates Adenylate Cyclase and Negatively Regulates Protein Kinase A-dependent Transcription in Saccharomyces cerevisiae. Journal of Biological Chemistry, 290(41):24715–24726, October 2015.

[80] Bastian Niebel, Simeon Leupold, and Matthias Heinemann. An upper limit on Gibbs energy dissipation governs cellular metabolism. Nature Metabolism, 1(1):125, January 2019.

[81] Elina Nikko and Hugh R. B. Pelham. Arrestin-mediated endocytosis of yeast plasma membrane transporters. Traffic (Copenhagen, Denmark), 10(12):1856–1867, December 2009.

[82] Avlant Nilsson and Jens Nielsen. Metabolic Trade-offs in Yeast are Caused by F1F0-ATP synthase. Scientific Reports, 6:22264, March 2016.

[83] Juan Nunez-Iglesias, Ryan Kennedy, Stephen Plaza, Anirban Chakraborty, and William Katz. Graph-based active learning of agglomeration (GALA): A Python library to segment 2D and 3D neuroimages. Frontiers in Neuroinformatics, 8:34, 2014.

[84] Paul Nurse. Cell cycle control genes in yeast. Trends in Genetics, 1:51–55, January 1985.

[85] Allyson F. O'Donnell, Rhonda R. McCartney, Dakshayini G. Chandrashekarappa, Bob B. Zhang, Jeremy Thorner, and Martin C. Schmidt. 2-Deoxyglucose impairs Saccharomyces cerevisiae growth by stimulating Snf1-regulated and α-arrestin-mediated trafficking of hexose transporters 1 and 3. Molecular and Cellular Biology, 35(6):939–955, March 2015.

[86] Allyson F. O'Donnell and Martin C. Schmidt. AMPK-Mediated Regulation of Alpha-Arrestins and Protein Trafficking. International Journal of Molecular Sciences, 20(3):E515, January 2019.

[87] Satoshi Okada, Marcin Leda, Julia Hanna, Natasha S. Savage, Erfei Bi, and Andrew B. Goryachev. Daughter Cell Identity Emerges from the Interplay of Cdc42, Septins, and Exocytosis. Developmental Cell, 26(2):148–161, July 2013.

[88] Chawin Ounkomol, Sharmishtaa Seshamani, Mary M. Maleckar, Forrest Collman, and Gregory R. Johnson. Label-free prediction of three-dimensional fluorescence images from transmitted-light microscopy. Nature Methods, 15(11):917, November 2018.

[89] S. Ozcan. Glucose sensing and signaling by two glucose receptors in the yeast Saccharomyces cerevisiae. The EMBO Journal, 17(9):2566–2573, May 1998.

[90] S. Ozcan, J. Dover, A. G. Rosenwald, S. Wölfl, and M. Johnston. Two glucose transporters in Saccharomyces cerevisiae are glucose sensors that generate a signal for induction of gene expression. Proceedings of the National Academy of Sciences of the United States of America, 93(22):12428–12432, October 1996.

[91] S Ozcan and M Johnston. Three different regulatory mechanisms enable yeast hexose transporter (HXT) genes to be induced by different levels of glucose. Molecular and Cellular Biology, 15(3):1564–1572, March 1995.

[92] Henry Pinkard, Nico Stuurman, Ivan E. Ivanov, Nicholas M. Anthony, Wei Ouyang, Bin Li, Bin Yang, Mark A. Tsuchida, Bryant Chhun, Grace Zhang, Ryan Mei, Michael Anderson, Douglas P. Shepherd, Ian Hunt-Isaak, Raymond L. Dunn, Wiebke Jahr, Saul Kato, Loïc A. Royer, Jay R. Thiagarajah, Kevin W. Eliceiri, Emma Lundberg, Shalin B. Mehta, and Laura Waller. Pycro-Manager: Open-source software for customized and reproducible microscope control. Nature Methods, 18(3):226–228, March 2021.

[93] Jeffrey A Polish, Jeong-Ho Kim, and Mark Johnston. How the Rgt1 Transcription Factor of Saccharomyces cerevisiae Is Regulated by Glucose. Genetics, 169(2):583–594, February 2005.

[94] Tim Prangemeier, Christian Wildner, André O. Françani, Christoph Reich, and Heinz Koeppl. Yeast cell segmentation in microstructured environments with deep learning. Biosystems, page 104557, October 2021.

[95] D. Pruyne and A. Bretscher. Polarization of cell growth in yeast. Journal of Cell Science, 113 (Pt 4):571–585, February 2000.

[96] D. Pruyne and A. Bretscher. Polarization of cell growth in yeast. I. Establishment and maintenance of polarity states. Journal of Cell Science, 113 (Pt 3):365–375, February 2000.

[97] Alexandra-M. Reimers, Henning Knoop, Alexander Bockmayr, and Ralf Steuer. Cellular trade-offs and optimal resource allocation during cyanobacterial diurnal growth. Proceedings of the National Academy of Sciences, 114(31):E6457–E6465, August 2017.

[98] Matthew Rocklin. Dask: Parallel Computation with Blocked algorithms and Task Scheduling. Proceedings of the 14th Python in Science Conference, pages 126–132, 2015.

[99] Christopher K. Rode, Lyla J. Melkerson-Watson, Amanda T. Johnson, and Craig A. Bloch. Type-Specific Contributions to Chromosome Size Differences in Escherichia coli. Infection and Immunity, 67(1):230–236, January 1999.

[100] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Lecture Notes in Computer Science, pages 234–241, Cham, 2015. Springer International Publishing.

[101] Adhiraj Roy, Yong-Bae Kim, Kyu Hong Cho, and Jeong-Ho Kim. Glucose starvation-induced turnover of the yeast glucose transporter Hxt1. Biochimica Et Biophysica Acta, 1840(9):2878–2885, September 2014.

[102] Adhiraj Roy, Yong Jae Shin, Kyu Hong Cho, and Jeong-Ho Kim. Mth1 regulates the interaction between the Rgt1 repressor and the Ssn6-Tup1 corepressor complex by modulating PKA-dependent phosphorylation of Rgt1. Molecular Biology of the Cell, 24(9):1493–1503, May 2013.

[103] D. Rudra. What better measure than ribosome synthesis? Genes & Development, 18(20):2431–2436, October 2004.

[104] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115(3):211–252, December 2015.

[105] Benjamín J. Sánchez, Cheng Zhang, Avlant Nilsson, Petri-Jaan Lahtvee, Eduard J. Kerkhoven, and Jens Nielsen. Improving the phenotype predictions of a yeast genome-scale metabolic model by incorporating enzymatic constraints. Molecular Systems Biology, 13(8):935, August 2017.

[106] Uwe Schmidt, Martin Weigert, Coleman Broaddus, and Gene Myers. Cell Detection with Star-Convex Polygons. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger, editors, Medical Image Computing and Computer Assisted Intervention – MICCAI 2018, Lecture Notes in Computer Science, pages 265–273, Cham, 2018. Springer International Publishing.

[107] Robert Schuetz, Nicola Zamboni, Mattia Zampieri, Matthias Heinemann, and Uwe Sauer. Multidimensional optimality of microbial metabolism. Science (New York, N.Y.), 336(6081):601–604, May 2012.

[108] Hans-Joachim Schüller. Transcriptional control of nonfermentative metabolism in the yeast Saccharomyces cerevisiae. Current Genetics, 43(3):139–160, June 2003.

[109] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. Hidden Technical Debt in Machine Learning Systems. In Advances in Neural Information Processing Systems, volume 28. Curran Associates, Inc., 2015.

[110] O. Shoval, H. Sheftel, G. Shinar, Y. Hart, O. Ramote, A. Mayo, E. Dekel, K. Kavanagh, and U. Alon. Evolutionary Trade-Offs, Pareto Optimality, and the Geometry of Phenotype Space. Science, 336(6085):1157–1160, June 2012.

[111] K. Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR, 2015.

[112] Kobi Simpson-Lavy, Tianchang Xu, Mark Johnston, and Martin Kupiec. The Std1 Activator of the Snf1/AMPK Kinase Controls Glucose Response in Yeast by a Regulated Protein Aggregation. Molecular Cell, 68(6):1120–1133.e3, December 2017.

[113] Chris Snowdon and George van der Merwe. Regulation of Hxt3 and Hxt7 Turnover Converges on the Vid30 Complex and Requires Inactivation of the Ras/cAMP/PKA Pathway in Saccharomyces cerevisiae. PLOS ONE, 7(12):e50458, December 2012.

[114] Nicholas Sofroniew, Talley Lambert, Kira Evans, Juan Nunez-Iglesias, Grzegorz Bokota, Philip Winston, Gonzalo Peña-Castellanos, Kevin Yamauchi, Matthias Bussonnier, Draga Doncila Pop, ACS, Ziyangczi, Pam, Alisterburt, Genevieve Buckley, Andy Sweet, Lukasz Migas, Volker Hilsenstein, Lorenzo Gaifas, Jordão Bragantini, Jaime Rodríguez-Guerra, Hector, Jeremy Freeman, Peter Boone, Alan R Lowe, Christoph Gohlke, Loic Royer, Andrea PIERRÉ, Hagai Har-Gil, and Abigail McGovern. Napari/napari: 0.4.11. Zenodo, September 2021.

[115] David J. Stillman. Nhp6: A small but powerful effector of chromatin structure in Saccharomyces cerevisiae. Biochimica Et Biophysica Acta, 1799(1-2):175–180, 2010 Jan-Feb.

[116] M. Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks. ICML, 2017.

[117] Peter S. Swain, Keiran Stevenson, Allen Leary, Luis F. Montano-Gutierrez, Ivan B. N. Clark, Jackie Vogel, and Teuta Pilizota. Inferring time derivatives including cell growth rates using Gaussian processes. Nature Communications, 7(1):13766, December 2016.

[118] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–9, June 2015.

[119] Stefano Di Talia, Jan M. Skotheim, James M. Bean, Eric D. Siggia, and Frederick R. Cross. The effects of molecular noise and size control on variability in the budding yeast cell cycle. Nature, 448(7156):947–951, August 2007.

[120] Marco Terzer and Jörg Stelling. Large-scale computation of elementary flux modes with bit pattern trees. Bioinformatics, 24(19):2229–2235, October 2008.

[121] Michelle A. Treitel, Sergei Kuchin, and Marian Carlson. Snf1 Protein Kinase Regulates Phosphorylation of the Mig1 Repressor in Saccharomyces cerevisiae. Molecular and Cellular Biology, 18(11):6273–6280, November 1998.

[122] Cong T. Trinh, Aaron Wlaschin, and Friedrich Srienc. Elementary mode analysis: A useful metabolic pathway analysis tool for characterizing cellular metabolism. Applied Microbiology and Biotechnology, 81(5):813, November 2008.

[123] R. Urbanczik and C. Wagner. An improved algorithm for stoichiometric network analysis: Theory and applications. Bioinformatics, 21(7):1203–1210, April 2005.

[124] Ophelia S. Venturelli, Ignacio Zuleta, Richard M. Murray, and Hana El-Samad. Population diversification in a yeast metabolic program promotes anticipation of environmental shifts. PLoS biology, 13(1):e1002042, January 2015.

[125] Steffen Waldherr, Diego A. Oyarzún, and Alexander Bockmayr. Dynamic optimization of metabolic networks coupled with gene expression. Journal of Theoretical Biology, 365:469–485, January 2015.

[126] Ping Wang, Lydia Robert, James Pelletier, Wei Lien Dang, Francois Taddei, Andrew Wright, and Suckjoon Jun. Robust Growth of Escherichia coli. Current Biology, 20(12):1099–1103, June 2010.

[127] Xin Wang, Kang Xia, Xiaojing Yang, and Chao Tang. Growth strategy of microbes on mixed carbon sources. Nature Communications, 10(1):1279, March 2019.

[128] Richard A. Watson, Günter P. Wagner, Mihaela Pavlicev, Daniel M. Weinreich, and Rob Mills. The Evolution of Phenotypic Correlations and "Developmental Memory". Evolution, 68(4):1124–1138, 2014.

[129] Martin Weigert, Uwe Schmidt, Tobias Boothe, Andreas Müller, Alexandr Dibrov, Akanksha Jain, Benjamin Wilhelm, Deborah Schmidt, Coleman Broaddus, Siân Culley, Mauricio Rocha-Martins, Fabián Segovia-Miranda, Caren Norden, Ricardo Henriques, Marino Zerial, Michele Solimena, Jochen Rink, Pavel Tomancak, Loic Royer, Florian Jug, and Eugene W. Myers. Content-aware image restoration: Pushing the limits of fluorescence microscopy. Nature Methods, 15(12):1090–1097, December 2018.

[130] Andrea Y. Weiße, Diego A. Oyarzún, Vincent Danos, and Peter S. Swain. Mechanistic links between cellular trade-offs, gene expression, and growth. Proceedings of the National Academy of Sciences, 112(9):E1038–E1047, March 2015.

[131] Meike T. Wortel, Han Peters, Josephus Hulshof, Bas Teusink, and Frank J. Bruggeman. Metabolic states with maximal specific rate carry flux through an elementary flux mode. The FEBS Journal, 281(6):1547–1555, 2014.

[132] Yan Wu, Jiaqi Wu, Minghua Deng, and Yihan Lin. Yeast cell fate control by temporal redundancy modulation of transcription factor paralogs. Nature Communications, 12(1):3145, May 2021.

[133] Karren Dai Yang, Anastasiya Belyaeva, Saradha Venkatachalapathy, Karthik Damodaran, Abigail Katcoff, Adityanarayanan Radhakrishnan, G. V. Shivashankar, and Caroline Uhler. Multi-domain translation between single-cell imaging and sequencing data using autoencoders. Nature Communications, 12(1):31, January 2021.

[134] Laurence Yang, Ali Ebrahim, Colton J. Lloyd, Michael A. Saunders, and Bernhard O. Palsson. DynamicME: Dynamic simulation and refinement of integrated models of metabolism and protein expression. BMC Systems Biology, 13(1):2, January 2019.

[135] Bo Yuan, Ciyue Shen, Augustin Luna, Anil Korkut, Debora S. Marks, John Ingraham, and Chris Sander. CellBox: Interpretable Machine Learning for Perturbation Biology with Application to the Design of Cancer Combination Therapy. Cell Systems, 12(2):128–140.e4, February 2021.

[136] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, Computer Vision – ECCV 2014, Lecture Notes in Computer Science, pages 818–833, Cham, 2014. Springer International Publishing.

[137] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. AI Open, 1:57–81, January 2020.

[138] Jing Zhu, Zheng-Tan Zhang, Shi-Wei Tang, Bo-Song Zhao, Hui Li, Jing-Zhen Song, Dan Li, and Zhiping Xie. A Validated Set of Fluorescent-Protein-Based Markers for Major Organelles in Yeast (Saccharomyces cerevisiae). mBio, September 2019.

[139] Luisa M. Zintgraf, Taco Cohen, T. Adel, and M. Welling. Visualizing Deep Neural Network Decisions: Prediction Difference Analysis. ICLR, 2017.