Bacterial Deepfakes: Generating Synthetic Microscopy Data to Improve Adaptability of Deep Learning-Based Segmentation Models

Daniela Koch

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2025

Reading Committee:
Paul A. Wiggins, Chair
Beth Traxler
Andrew Laszlo

Program Authorized to Offer Degree:
Physics

University of Washington

Abstract

Bacterial Deepfakes:
Generating Synthetic Microscopy Data to Improve Adaptability of Deep Learning-Based Segmentation Models

Daniela Koch

Chair of the Supervisory Committee:
Professor Paul A. Wiggins
Department of Physics
Department of Bioengineering

Quantitative analysis of biological images typically involves image segmentation; the identification of pixels belonging to individual cells or structures present in each image. Given a sufficiently large and diverse set of training images, deep-learning based models are able to reliably automate segmentation with adequate precision for further analysis. However, the significant time and expertise required to annotate training images often makes the use of such models impractical when studying new systems or phenomena. To lower the burden of training data curation, we present approach for generating and training with synthetic images. We introduce OmniStyle, a custom conditional-generative-adversarial network (CGAN) for generating annotated synthetic images to artificially increase the size and/or diversity of training datasets. We demonstrate that training with images generated by OmniStyle improves segmentation performance on diverse datasets without additional hand-annotations. To evaluate the utility of our method, we train an identical segmentation model on datasets of equal size consisting of exclusively real images and compare performances with those trained with real and synthetic images.

TABLE OF CONTENTS

LIST OF FIGURES

artificial neural networks Computer systems modeled after the brain's networks that process data by passing signals through connected layers. 26

batch size The number of training samples processed before the model's weights are updated. 67

computer vision A field of AI that enables machines to interpret and understand visual information from images. 18

convolutional neural network a type of deep learning model specifically designed for processing grid-like data such as images.. 27

deep neural networks A type of machine learning that uses layered neural networks to learn patterns from data. 26

E. coli A common bacterium found in the intestines of humans and animals, often used as a model organism in biology. 13

epochs Complete passes through the entire training dataset during model training. xiii, 90

feature space A multi-dimensional space where each dimension represents a feature used to describe data points. 95

field-of-view The visible area captured by an imaging device or seen through a microscope at one time. 13

fluorescent tagging A method of attaching fluorescent markers to molecules or cells to visualize them under specific lighting in microscopy. 6

foundation models Deep-learning models trained on vast quantities of *unlabeled* data that are designed to be used for a variety of downstream tasks. 95

ftsN A bacterial protein essential for cell division, helping form the division septum in cells like *E. coli*. 9

generative adversarial network Strategy for training generative models to produce "new" samples.. 66

generative machine learning A type of machine learning that creates new data similar to what it was trained on, like images or text. 24

image segmentation partitioning of an image into distinct regions or objects. 2

imaging modality A specific technique or method used to capture images, such as MRI, CT, or microscopy. 4

inference The process of using a trained model to make predictions on new data. 3

instance segmentation A computer vision task that labels each pixel by both category and object, distinguishing between individual objects of the same type. iii, 4

learning rate A parameter that controls how much a model's weights are updated during training. 76

loss function a mathematical function that quantifies the difference between ground-truth target and model prediction.. 25

minibatch A small subset of data used to train a machine learning model in one iteration, improving training efficiency. 51

mode collapse common failure mode of generative adversarial networks. 81

model checkpoints Saved versions of a machine learning model during training, allowing recovery or evaluation at specific points. vi, 30

optimizer An algorithm that adjusts a model's weights to reduce errors during training. 76

phase contrast A microscopy technique that enhances contrast in transparent samples without staining. iii, 3

semantic segmentation A computer vision task that labels each pixel in an image with a category. iii, 4

septation The process of forming a dividing wall (septum) during cell division in bacteria. 9

style transfer A technique that applies the style of one image (like a painting) to the content of another image. 49

supervised machine learning A type of machine learning that learns from labeled data to make predictions or classifications. 24

training dataset A collection of labeled data used to teach a machine learning model to make predictions. 16

U-net A neural network architecture designed for image segmentation, featuring a U-shaped structure with downsampling and upsampling paths. 68

validation dataset Data used during training to tune a model's settings and prevent overfitting. vi, 28

ANN Artificial Neural Network. 26

cGAN Conditional Generative Adversarial Network. 52

CNN Convolutional Neural Network. 27, 63

DL Deep Learning. 1

DNN Deep Neural Network. 26

FOV Field of View. 13

GAN Generative Adversarial Network. 49

GPT Generative Pretrained Transformer. 94

GPU Graphical Processing Unit. 27

IoU Intersection over Union. 57

LLM Large Language Model. 94

ML Machine Learning. 1

ood out-of-distribution. 4

ACKNOWLEDGMENTS

Writing a thesis is in many ways a profoundly solitary endeavor, and yet what stands out most in the end are the people in my life who actually made this all possible. I'd like to take this chance to acknowledge some of those people for their very real contribution to this work.

First and foremost, to my thesis advisor, *Paul Wiggins*: for your guidance and open-minded support throughout this entire process, for cultivating a creative, collaborative, and curious learning environment, for throwing me off the deep end into a field I knew nothing about, and for giving me the autonomy and agency I needed to figure it out for myself...and for fishing me out of the occasional rabbit-whole along the way. You helped me learn how to learn, and for I am sincerely, deeply grateful.

To every member of the *Wiggins Lab*, past and present: for the hours spent in the basement, for our daily zoom coffee sessions during the pandemic, for playing inside baseball, for listening to every version of every big talk I've given throughout the years, for everything you've all taught me. Your collaboration and comradery made me a better, more confident scientist. Thank you. And special thanks to Kevin Cutler, who is arguably the real protagonist of this dissertation.

To my fellow physicist-turned-machine-learning-researcher, *Olivia Zahn*: for struggling through a brand-new field alongside me (guh), for keeping me sane during the pandemic, for listening to the chaos of thoughts and ideas on my mind and somehow managing to extract something meaningful, for every Swansons trip, for convincing me to give Reputation another

chance, and for telling me to "just get a kitten, it'll be fine." I literally can't imagine what this PhD would have looked like without you.

To my brain's second half, *Alex Kineret*: for keeping my spirits high, but always getting on board with a good old-fashioned doom-spiral when the universe calls for it. Between the two of us, there is (almost) always enough executive function to accomplish big serious things in life, like this degree, without ever taking ourselves too seriously. Love you, Grumbles.

To my too-many animals at home, *Yzma, Niko,* and *Mipha*: for being blissfully ignorant of the stressors in my life, and for being the best thing to happen to me during graduate school.

And finally, to the one person who may be the one person more excited about the completion of this PhD than I am, my partner, and my favorite person, *Michael Martinez*: for every late night, every cup of coffee, every snack run, every wärmflasche, every vet visit, and for every other way you've kept me and zoo afloat over the years, no amount of morning coffees will ever be thanks enough. I love you.

DEDICATION

I dedicate this work to my parents,
for the roots and wings, and everything in between
from the bottom of my heart - *thank you*.

Chapter 1

INTRODUCTION

However you feel about the current craze surrounding machine-learning and artificial intelligence, their impact on scientific study — and even our daily lives — is undeniable. Machine-learning (ML) is everywhere, especially in research. From detecting particle-collision anomalies at CERN [1] to predicting protein-folding structures from amino-acid sequences [33], machine-learning models have become powerful research tools across a diverse range of disciplines. In our work, we are interested in developing such tools for quantitative bacterial microscopy applications. This thesis tells the story of the challenges we've encountered in this pursuit, the solutions we've developed to navigate them, and some of the bigger-picture lessons we've learned along the way.

1.1 Machine Learning for Quantitative Imaging

Quantitative biological imaging has benefited immensely from recent breakthroughs in machine-learning. Extracting quantitative data from images is usually a complex and tedious, yet unavoidable task for image-based analysis, so large-scale analyses of imaging datasets are only practical if this step is automated. Machine learning makes this significantly more achievable; Deep learning (DL) models in particular have demonstrated an impressive aptitude for handling complex biological imaging-data for a variety of computer-vision tasks. Through the automation of time-intensive image-processing steps, machine-learning has played a critical role in enabling large-scale quantitative analyses of otherwise qualitative observations.

One of greatest promises of machine-learning in research is it's potential to unlock access to otherwise impractical areas of scientific exploration. Machine-learning models are remarkably successful at efficiently handling vast quantities of highly-complex data, a task

which often creates bottlenecks for researchers. When implemented thoughtfully, ML models can reduce or eliminate this bottleneck to allow scientists to investigate systems from fundamentally new perspectives.

With novel approaches, however, come new challenges. One of the major hurdles of applying machine learning to biological datasets is training data acquisition and annotation [64]. Machine-learning models need data to learn. A *lot* of data - generally speaking, the more data a model is trained with, the better. Dataset curation can limit the use of machine-learning in practice because it is a time-consuming process, and usually requires significant domain expertise in scientific contexts. This challenge is further magnified in research settings if important data features change unexpectedly in response to experimental parameter variation, because such changes can can render previously trained models obsolete. Even the most sophisticated cutting-edge machine-learning model is of no use to researchers if that model requires constant retraining, and therefore new training data, for every change in experimental configuration.

Limitations in training data availability pose a fundamental challenge the application of machine-learning based tools because it makes those tools less generally useful, but more importantly, it limits what we can study with those machine-learning based tools. Reducing the burden of training data curation is an important component in improving the accessibility of high-quality machine-learning algorithms in settings where they can make the most impact, such as quantitative biological imaging. High-quality, reliable, and *adaptable* image-processing, which can be the rate-limiting step of large-scale statistical analyses of information-rich imaging datasets of new phenomena.

Creating *usable* research tools from novel machine-learning methods is the premise of this thesis. The context of our work is quantitative bacterial microscopy imaging, where we leverage machine-learning to automate the processing of imaging data. Specifically, we use deep-learning models for a task called *image segmentation*, in which we partition an image into distinct regions or objects (see Figure 1.3). Image segmentation is in itself an rich, ongoing topic in computer-vision research, but unique challenges arise in biological research

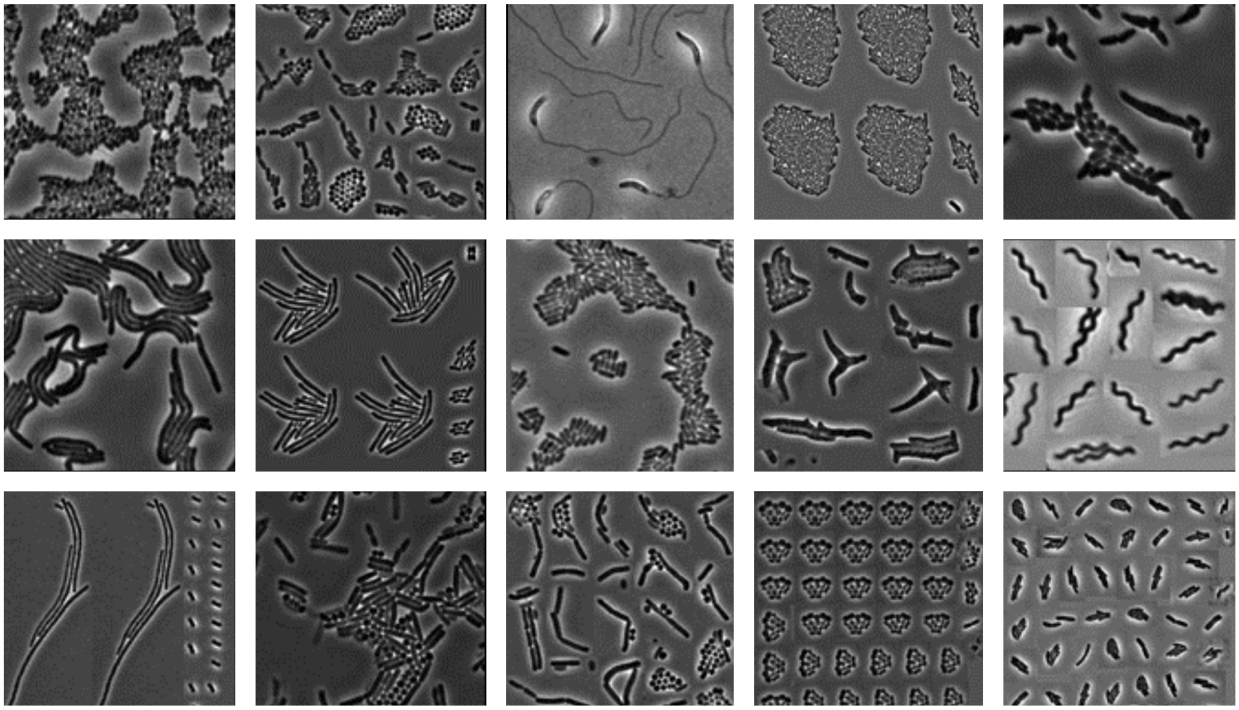


Figure 1.1: **Diversity in bacterial microscopy images in phase contrast.** Bacterial microscopy images are diverse, even when limited to a single imaging modality. These images are all in phase contrast, yet vary significantly in morphology due to differences in species or morphology-altering mutations. This illustrates the diversity of data possible within even a single imaging modality.

applications; Segmentation algorithms must be extremely precise for reliable downstream analysis, yet robust to the high levels of noise and diversity inherent to many biological systems of interest. Machine-learning approaches, particularly deep-learning, have gained popularity because they have been shown to achieve this critical balance through an impressive potential for sufficiently complex and subtle inference. As a result, there has been a focus on developing machine-learning based image-segmentation models specifically for high-resolution microscopy applications [5, 4, 66, 55].

This approach has been successful in our previous experiments [67], but we also have

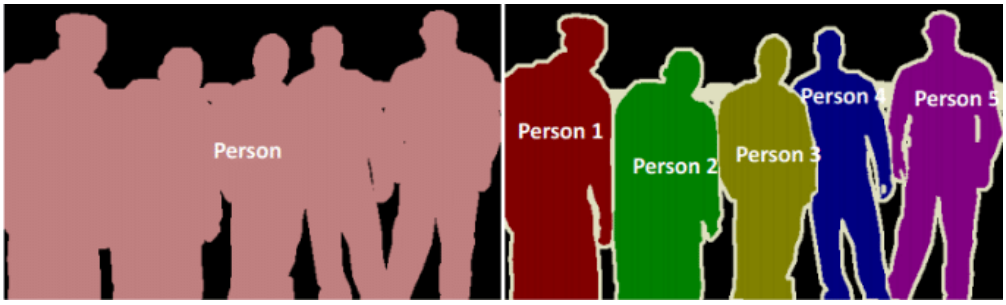


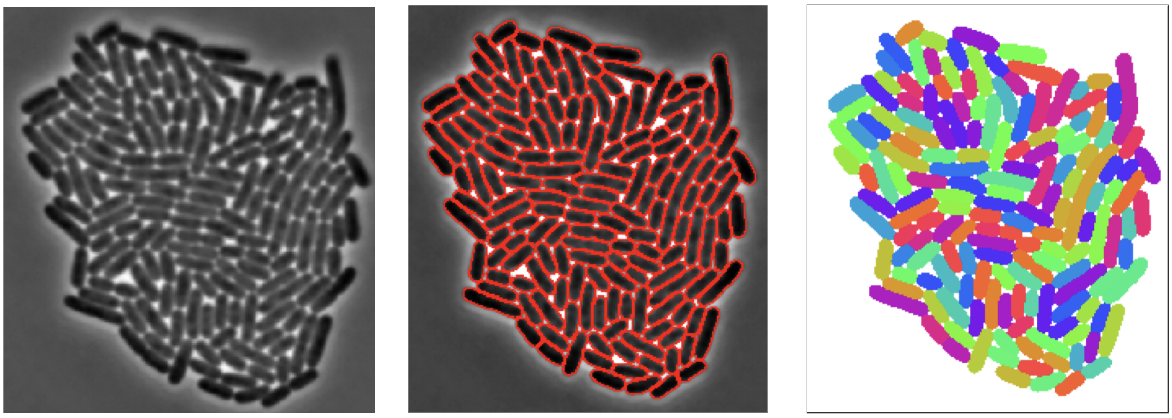
Figure 1.2: **Image Segmentation:** Semantic vs. Instance [71]. In semantic segmentation (left), each pixel is assigned a class according to the type of object it belongs to. Any and all pixels that make up a any person present in the image are assigned to the same "person" class. Instance segmentation (right) further specifies *which* individual person each pixel belongs to, which allows us to differentiate between multiple objects of the same class.

encountered the data limitations when new experimental approaches lead to changes in cell appearance. This is a general problem when working with bacterial imaging data, which can be highly diverse even within a single imaging modality, as illustrated in Figure 1.1. Through the course of our work, we have found continual retraining, and therefore also manual dataset-curation, unavoidable for maintaining performance on new the new images. Even "generalist" models require examples of the new images before they can be used reliably. This retraining requirement significantly reduces the power of these approaches and limits the scope of applications to those that lead only to modest changes in cell morphology. Our work centers on navigating this challenge using synthetic imaging data.

This thesis explains in detail our approach for generating synthetic images and how they can reduce the burden of dataset creation. Specifically, our objective is to reduce the number of training images required to adapt existing machine-learning models to new, "out-of-distribution" (ood) samples, and we examine the extent to which synthetic data can be used for training high-precision models for high-resolution microscopy applications.

1.2 Bacterial Microscopy

Quantitative microscopy is an important tool in the study of bacteria. High-resolution microscopy images allow us to observe and characterize bacteria the individual *and* population level. With enough samples, we can use imaging data quantitatively study bacterial population dynamics and measure physical characteristics such as the size, shape, or intensity values of individual cells [28]. We can even capture and quantify changes in metrics such as cell morphology, population density, and microcolony formation over time using time-lapse data. These kinds of observations offer unique insights into bacterial processes and dynamics, but large-scale analysis is only possible if we can efficiently extract quantitative information from imaging data. This is particularly true for large-scale single-cell analyses, in which metrics of interest are collected or calculated for individual cells, rather than averaging over populations.



(a) Phase contrast image (b) Cell boundaries (c) Single-cell mask regions

Figure 1.3: **Single-Cell Segmentation.** This is an example of single-cell segmentation of *Escherichia coli* microcolony in phase contrast. Segmentation may be represented by identifying cell-boundaries (middle) or masks defining single-cell regions (right).

Image segmentation is a standard image-processing approach for obtaining single-cell

data for quantitative microscopy analysis. The term "image-segmentation" refers to the partitioning of images into distinct regions or, in our case, cells. This involves determining which pixels belong to each cell for every cell in an image (see Figure 1.3). By segmenting images, we can identify all individual cells present to make any desired measurement (fluorescence intensity, cell size, growth-rate, etc) from the microscopy image using only the relevant pixels for each individual. Quantitative single-cell imaging datasets created this way can be used to study a wide range of phenomena, but the quality of these datasets (and any subsequent analysis) are contingent on accurate and precise segmentation. As we will discuss in greater detail throughout this work, development of tools that enable robust segmentation remains an ongoing challenge [27].

1.2.1 Segmentation for quantifying overabundance of essential genes

Essential genes are, as their name implies, genes that a cell cannot survive without. We can learn a lot about the function of essential genes by observing populations in which those genes have been removed from the genome. Recent work in the Wiggins lab has focused on the phenomenon of protein overabundance in bacteria, in which cells produce more copies of a gene than is necessary for survival. At first glance, this is surprising because genes require energy to produce - so why might a cell want to produce more than absolutely necessary? A short explanation is that cells stockpile essential genes to ensure they are robust to stochastic fluctuations in protein expression levels (for more, read the paper!) [16]. Quantitative bacterial microscopy offers unique insights to this phenomenon - one way to measure overabundance is by counting the number of generations that a cell has after the protein has been depleted.

For example, when studying the relative importance of essential genes, we can use fluorescent tagging to mark proteins of interest in our samples and observe how their numbers vary over time. By creating a mutation in which the protein of interest is no longer produced, we can easily observe the depletion of that protein in the population over time, and measure the number of cell cycles that are able to occur after the mutation was performed. This

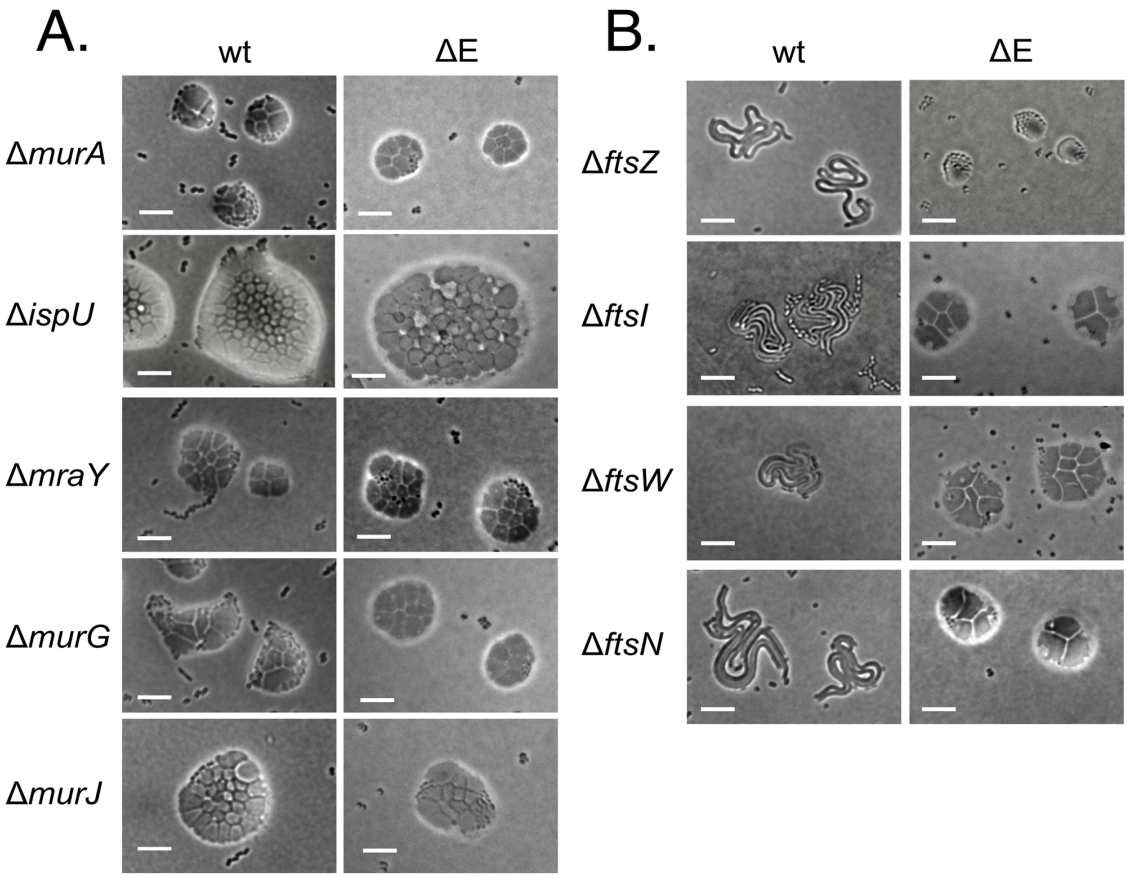


Figure 1.4: **Blocking peptidoglycan precursor synthesis or polymerization leads to giant cells. (A)** Precursor synthesis mutants. Microcolonies formed after deletion of genes required for different steps of peptidoglycan precursor synthesis are shown. Deletions of meso-diaminopimelic acid synthesis genes dapA and dapB also resulted in giant cells. Colonies were incubated 20–24 h on protective agar. Scale bar, 10 μm. **(B)** Microcolonies of cell division mutants. Microcolonies formed after deletion of genes required for cell division are shown. Colonies were incubated 20–24 h on protective agar. wt, wild-type (MAY101) genetic background; ΔE, peptidoglycan elongation-minus triple mutant (ΔpbpA ΔrodA ΔponA) genetic background (MAY106). Scale bar, 10 μm. Figure and caption from [8]

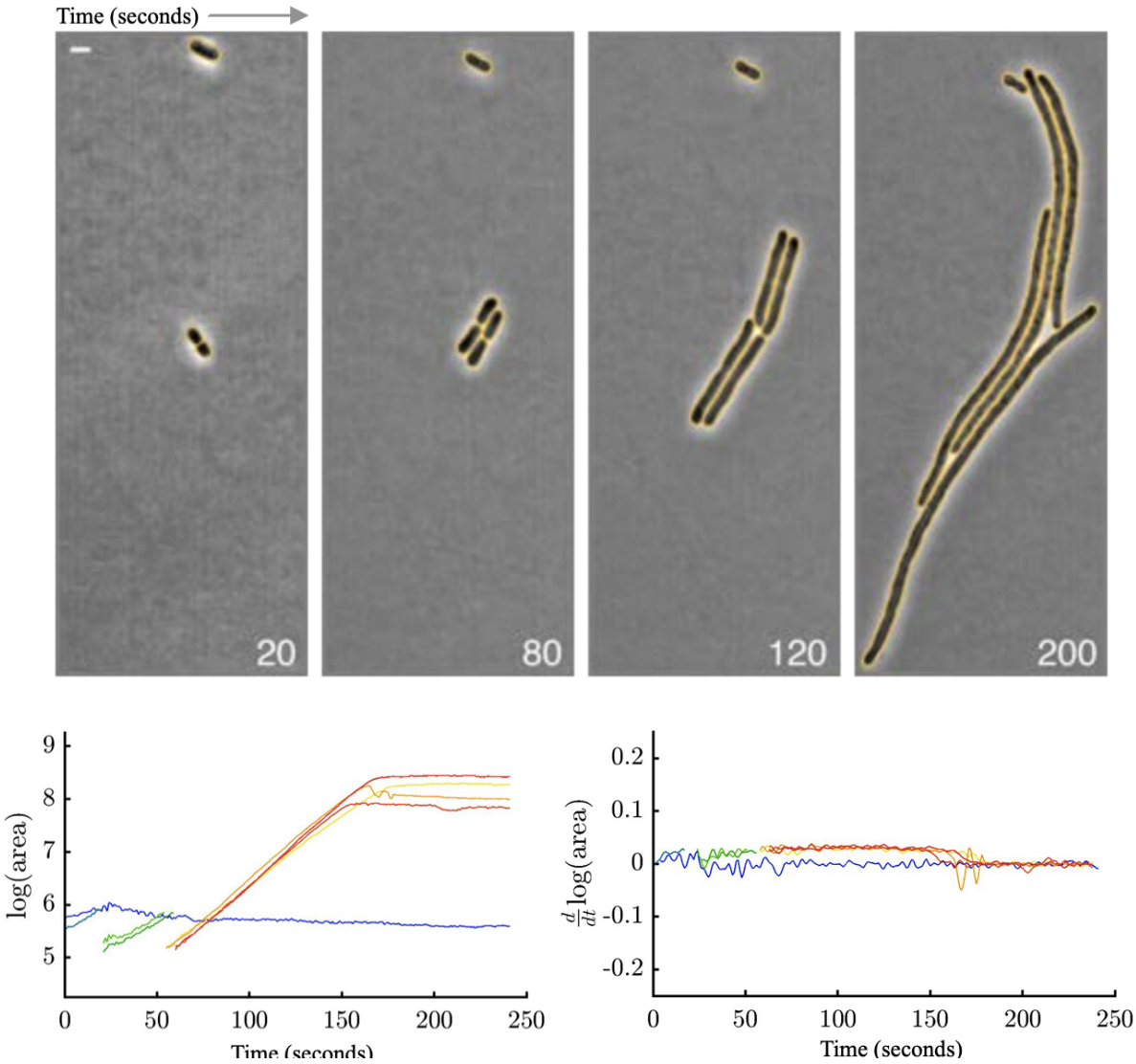


Figure 1.5: **Quantitative imaging measurements of protein overabundance.**

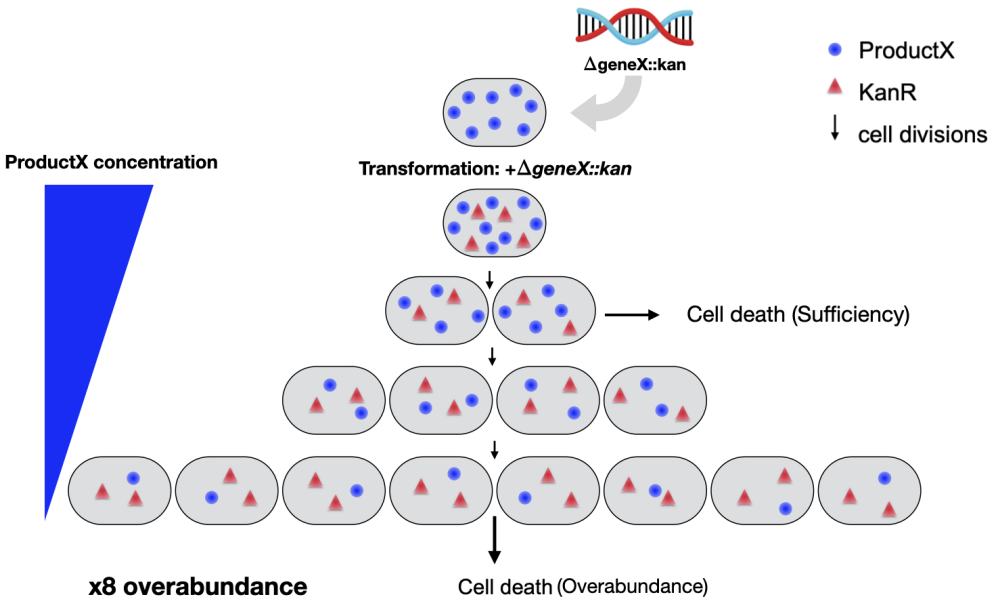


Figure 1.6: **Diagram of protein overabundance.** Figure adapted from [16].

approach can in some cases lead to significantly different results than genetic sequencing approaches, where the total protein count is measured at the end and that is used to determine the number of generations of daughter cells that were able to proliferate after the mutation. However, that is not always a good measure of the metric in question, as replication may happen for more cycles than there are daughter cells. For example, when the essential gene *ftsN* is removed from *Acinetobacter baylyi* (*A. baylyi*), the cell wall will no longer form a septum during septation. This means that, while the replication cycle continues, the cell will no longer be able to split into two daughter cells and will instead continue to grow into long "spaghetti" cells, which are not viable and can be considered dead. In contrast, the sequencing approach would count the number of replication cycles and therefore overestimate the number of additional possible cell cycles post-mutation.

1.3 Automated Image Segmentation

For large scale statistical analyses, we need to be able to efficiently segment many images with reliably high precision. Existing approaches range in complexity, from relatively straightforward (and interpretable) thresholding methods to cutting-edge machine-learning algorithms. The. Efficient high-precision segmentation of bacterial images is particularly challenging for two major reasons. Firstly, bacteria are very small, so every pixel counts. Missing even a few pixels can cause significant uncertainty in downstream analysis. This is particularly challenging because we are working at microscopic scales; the small size of the cell causes it to experience the effects diffraction as it nears the microscope's depth of field, and make images difficult to interpret [28], even for humans. Secondly, bacterial microscopy images are diverse. Factors such as cell morphology, population density, and imaging modality make it difficult to develop algorithms that generalize across these factors with sufficient precision.

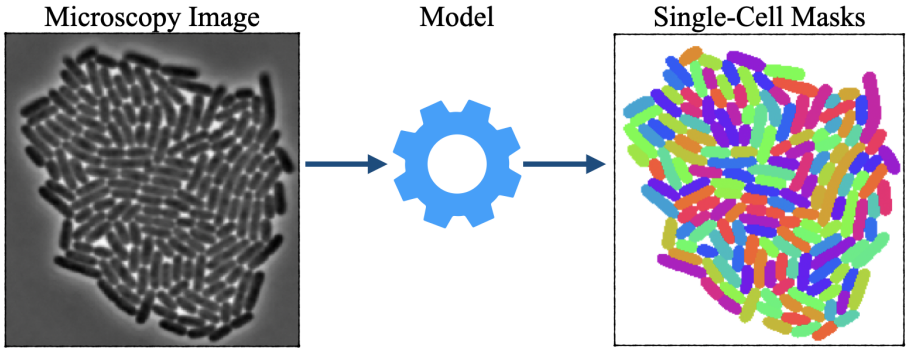


Figure 1.7: **Automated Image Segmentation.** Large-scale quantitative analysis of bacterial images is only practical if segmentation is automated; that is, we require a model/algorithm that automatically computes single-cell masks regions from the original microscopy image.

Machine learning based image segmentation models have shown great promise in handling

the noise and complexity of bacterial microscopy images, consistently outperforming previous traditional "rules-based" computational approaches. In this section, we examine some of these factors and how they pertain to image segmentation in greater detail.

1.3.1 A closer look at image segmentation

Generally speaking, there are two types of image segmentation - semantic and instance segmentation(see Figure 1.8). In semantic segmentation, each pixel is assigned a class according to the type of object it belongs to. Any and all pixels that make up a any cell present in the image are assigned to the same "cell" class. Instance segmentation further specifies *which* individual cell each pixel belongs to, which allows us to differentiate between multiple objects of the same class. As it's name suggests, quantitative *single-cell* microscopy requires the ability to distinguish between individuals in a given image, and we therefore use *instance* segmentation.

High-precision, reliable image segmentation is critical, particularly when considering its impact on downstream analysis. Any errors that occur during segmentation will propagate through any subsequent analysis; note that segmentation errors may vary in severity and we should prioritize our efforts accordingly - some errors may be negligible while others downright catastrophic to subsequent analysis, and must be avoided. This often depends on the downstream task at hand, but missing half the pixels of a given cell, for example, is generally more problematic than missing just a few pixels, but perhaps not as catastrophic as under-segmenting an image by labeling multiple adjacent cells as a single cell. Preventing these errors is essential for maintaining the integrity of quantitative biological insights.

Identifying common errors and understanding when and where they may arise is an critical component in the development of segmentation algorithms. For example, the more cells that are present in a given frame, the higher the probability of "catastrophic" segmentation error, which can in turn increase error in downstream analyses. Such segmentation errors include temporal "2-to-1" problems, where segmentation fails to reflect changes over time accurately. These include situations where two distinct cells are incorrectly merged into one,

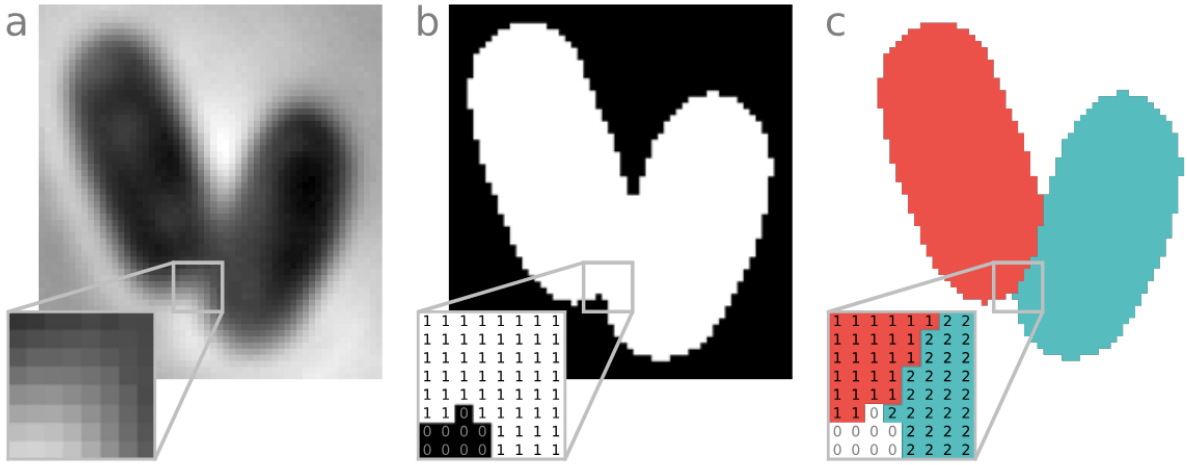


Figure 1.8: **Semantic vs. Instance Segmentation** [19]. (a) The phase-contrast microscopy image of *E. coli* to be segmented. (b) Semantic segmentation: each pixel inside *any* cell is assigned the value 1, representing "cell" class All background pixels are assigned value 0, representing the "non-cell" class. (c) Instance segmentation: Each cell is represented with a distinct integer value, and each pixel in an individual cell is assigned the corresponding integer value. Note: these values serve only as labels to distinguish between cells, but have no further meaning. All background pixels are still assigned value 0

especially when they come into contact in subsequent frames, or when recently divided cells are mistakenly fused back together. Such errors are particularly problematic because they propagate through downstream analyses, making tasks like cell tracking extremely difficult. Furthermore, these types of errors become increasingly common as the number of cells in the field of view grows, since a higher cell density raises the likelihood of encountering cells in the process of division or in close contact. Furthermore, as cells reproduce and form microcolonies, contrast between cells in a microcolony decreases so it becomes more difficult to identify boundaries between neighboring cells.

1.3.2 Classical Programming Approach

Not all bacterial segmentation requires machine-learning. For example, traditional image processing techniques can be highly effective for segmenting images containing a small number of well-spaced cells. Commonly used methods such as thresholding and watershed algorithms are often sufficient in these scenarios. Thresholding is used to identify cell regions (masks) based on pixel intensity, while the watershed algorithm is applied to separate adjacent or touching cells within those regions.

The primary advantages of traditional image processing approaches are their speed and, crucially for many research settings, their interpretability and reproducibility. Because these methods are rule-based, their behavior is transparent, and sources of error can often be identified and corrected through direct modifications to the algorithm. This level of control makes traditional approaches especially attractive in settings where traceability and manual debugging are essential.

However, these methods have significant limitations, particularly when applied to more diverse or complex datasets. As the variability of input images increases—due to differences in cell type, imaging conditions, or sample preparation—the performance of rule-based algorithms tends to degrade unless they are manually re-tuned. This manual tuning process is time-consuming and does not scale well, making it difficult, and often impractical, to define a fixed set of rules that performs consistently across a large and heterogeneous dataset. Consequently, while traditional methods are well-suited to narrowly defined tasks, they lack the flexibility and generalization capacity required for broader applications.

For example, consider a time-lapse of *E. coli* cells in phase contrast. In early frames when cells are well-distributed across the field-of-view (FOV), thresholding can be used to find semantic cell masks, followed by a watershed algorithm to distinguish between individuals if necessary. As time goes on and cells continue to proliferate, we have to work harder to distinguish between individual cell masks for a few reasons. The first is simply because there are more cells in contact for any given frame. Secondly, frames containing in which a cell

has visibly formed a septum but not yet divided become more common, and will very likely be segmented as two separate cells before the cell has actually divide. Furthermore, as cells form microcolonies, the contrast between cells in the center of the microcolony decreases - in some cases there are frames in which can be difficult for even a human to decipher an exact boundary. Thresholding approaches are likely to drastically under-segment these images due to the decreased contrast and increased cell-to-cell contact. Similarly, when cells take on diverse morphologies, common processing strategies like erosion or dilation cannot be applied to all cells the same way.

1.3.3 Machine-Learning Approach

When images become sufficiently complicated, machine-learning can offer solutions. Previous work on segmentation in the Wiggins lab aimed to address some of the challenges mentioned above, such as increased cell density and morphological variation, using machine-learning approaches. SuperSegger [67], originally published in 2016, enabled robust segmentation of high-density images of *E. coli* cells in phase contrast. This method is designed specifically for rod-like cells, as shown in Figure **??**, and facilitates scalable analysis of cells with similar morphologies including time-lapse analysis, such cell-tracking and linage-tracing over time. However, because SuperSegger was designed specifically for rod-shaped cells, it struggled to adapt to other morphologies. As illustrated in Figure 1.12, it turns out that this is not unique to SuperSegger. Machine-learning models in general struggle with diverse morphologies.

The most recent work on segmentation in the Wiggins lab aimed to address this challenge; Omnipose [20] was developed specifically for morphologically invariant segmentation. Omnipose is a deep-learning based approach to image segmentation, which we will examine in more detail in the following chapter as it is the starting point of our work. Morphology is a persistent source of cell variation across imaging types. For any new imaging modality, we need to be able to segment diverse morphologies. Much of this variation comes from variation in species, but morphological changes in a given species can occur as a result of mutations or drug treatments. These mutations are important in studying some of the most

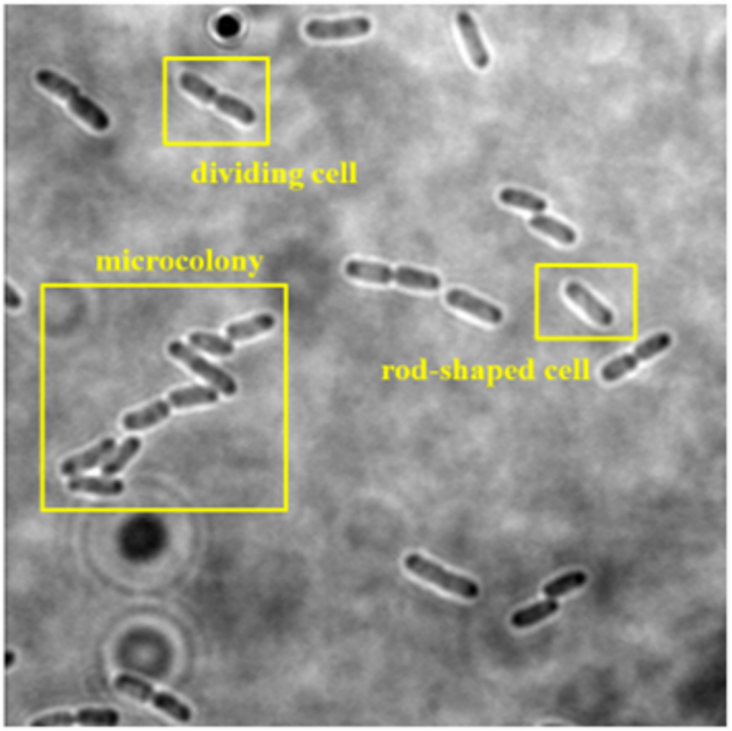


Figure 1.9: **Growth phases of *E. Coli* in brightfield.** [15]. We can see three distinct growth phases in this image. Rod-shaped cells elongate, divide, and repeat this process to form microcolonies. Each phase of growth has distinct challenges in image segmentation. Classical approaches such as simple thresholding may work well for for cells like the one labeled rod-shaped because there is a clear contrast with the background and is not in contact with any other cells. Cells that are in contact with other cells are a bit more challenging, but for sufficiently low cell-density we can use a watershed (or similar) algorithm on thresholding masks to distinguish between neighboring cells. The dividing cell, however, complicates this step because the visible septum formation could easily lead to this single cell being segmented as two neighboring cells. As cells proliferate and form microcolonies, occurrences of cell-to-cell contact and dividing cells become more frequent.

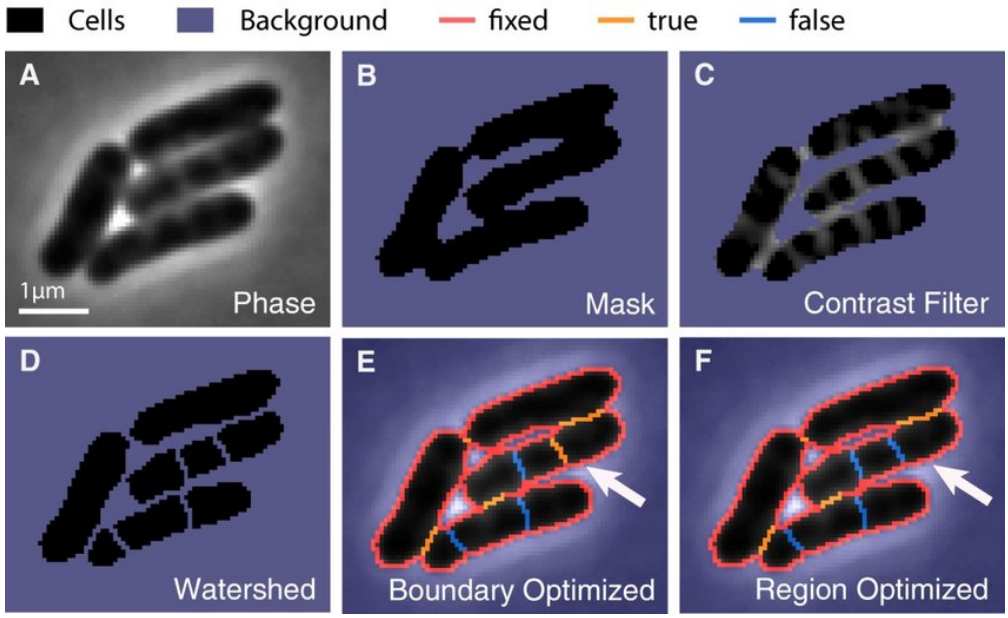


Figure 1.10: **SuperSegger: Single-cell segmentation of rod-like morphologies**

interesting questions we have about things like essential gene function, so we want to make sure we can reliably segment any morphology that may occur. Omnipose *is* able to segment diverse morphologies, but its success still ultimately hinges on - you guessed it - training data availability.

1.4 ML problems require ML solutions

An almost inescapable tradeoff when working with machine-learning models is that they require an extremely large and representative training dataset. Models learning tasks that require high-precision, such as image segmentation, must be trained on equally high-precision data. To compile such a training set, images must be manually labeled, typically by hand-annotation, which can take hundreds of hours and requires domain-specific imaging expertise. Even seemingly minor experimental changes (e.g., growth conditions, genetic background on the cell, or drug treatments) can lead to sufficiently large changes in cell morphology to result in significant reductions in segmentation performance, as illustrated in 1.12. Limited

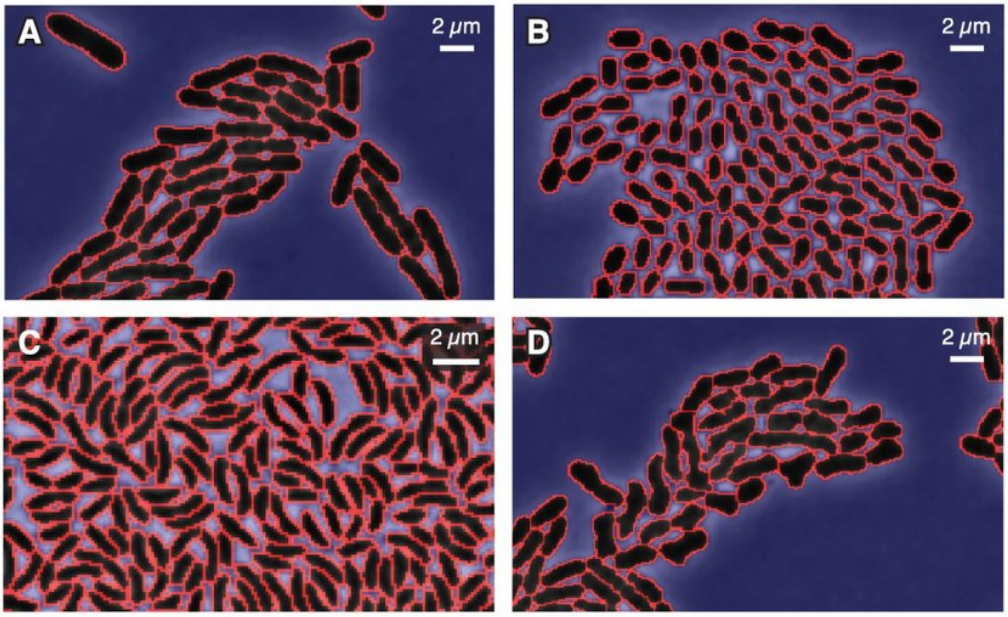


Figure 1.11: **SuperSegger: High-Throughput Segmentation** [67]. SuperSegger is a high-throughput bacterial microscopy segmentation and analysis tool. It is a MATLAB based software package that was originally designed specifically for rod-shaped bacteria.

training data availability can render deep-learning based segmentation models impractical, particularly in research settings where experimental conditions vary most often. This is a major hurdle for building tools that researchers can not only trust, but *actually* use.

1.4.1 The ML Problem: Never. Enough. Data

The major bottleneck in the dataset curation process is the labeling of images. Each image must be carefully annotated at the pixel-level, with each pixel assigned to a specific cell for each image. This process takes time and significant domain-specific expertise, thereby limiting the amount of images that can be reasonably attained to be used for model training. This problem is exacerbated by the variability in experimental conditions or imaging modalities, all of which can often result in images that differ sufficiently from those used to

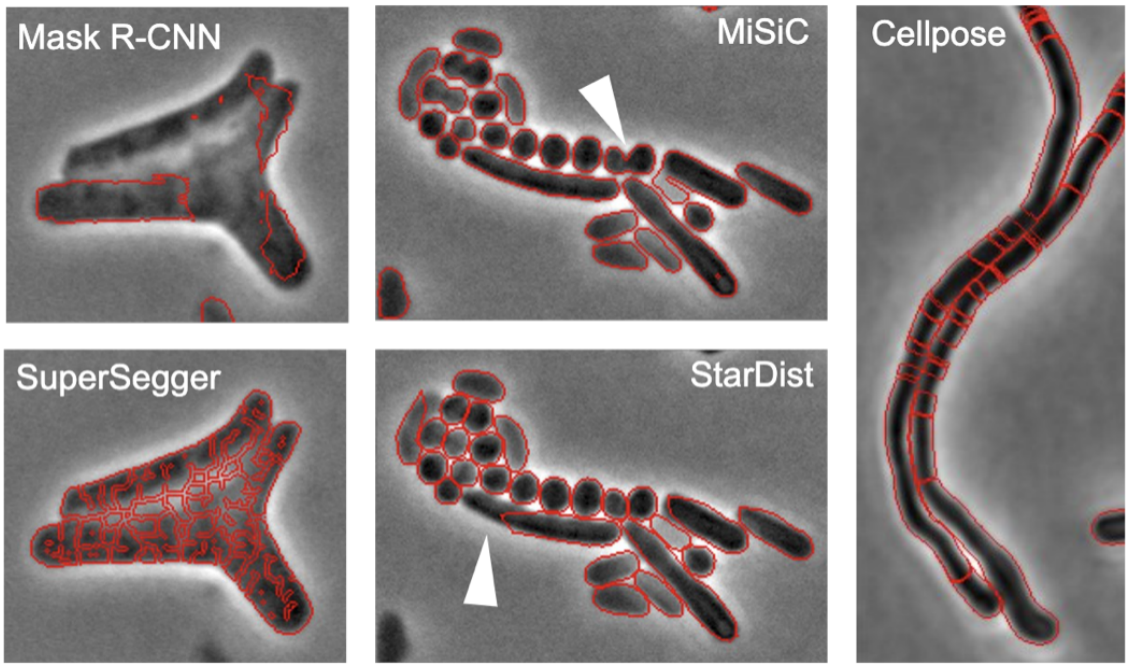


Figure 1.12: **The data diversity challenge.** These are sample segmentation predictions on diverse morphologies, produced by leading segmentation models. We can see that models are unable to adapt to new features in data such as changes in cell morphology, leading to reduced segmentation performance from otherwise high-performing models. Models shown are: Mask R-CNN by Meta AI [30], MiSiC [57], SuperSegger [67], StarDist [36], and Cellpose [66]

train existing models such that those models will not perform reliably on the new dataset. The most obvious solution to this problem would be to annotate new sample from the new dataset and train new models with those samples, but this again takes time and expertise.

A common computer vision approach when working with limited training samples is to train models using a combination of real- and synthetic-training data . High-quality synthetic data can drastically reduce (or in some cases eliminate) the number of real training samples required for reliable model performance [46]. While the precision required for single-cell analysis makes it unlikely that need for training data can be eliminated entirely, machine-

learning based microscopy segmentation would be more practical if models could be trained using fewer images without compromising performance.

In the case of microscopy images, creating sufficiently realistic synthetic images is in itself a challenge. Some approaches exist for simulating the underlying optics in microscopy images [29], but these can be computationally intense, and often fail capture the randomness present in real images such as artifacts or the visible internal structures of cells. But how much do we really care about the underlying optics? It is not necessary for the synthetic to mimic the optics perfectly. We just need an algorithm that can produce sufficiently convincing images with which to train segmentation models. We need to be able to generate images efficiently, and we care more about getting an algorithm that works than being able to understand how the images are generated. These factors make machine-learning a good candidate for this task. Therefore, our solution to the problem of limited training data for machine-learning models, is to generate synthetic training samples...with another machine-learning model.

1.4.2 Our Solution: Generative Machine-Learning

The same sources of noise and complexity that make these images hard to segment must be adequately represented in any synthetic samples intended to be used during training. Like for segmentation, we don't really care *how* the image is generated. We just need that the image is sufficiently convincing to use as training data for segmentation models, not that it accurately depicts the underlying optics.

An important consideration for synthetic data generations it that we want to be able to generate out-of-distribution samples to be used to fine-tune (or entirely retrain) a model to perform better on out-of-distribution test samples. These generated samples must also be annotated to be used in downstream segmentation model training. For these reasons, we implement a *conditional* generator model takes in user-provided masks. The process for generating images is explained in thorough detail throughout Chapter 3, Figures 1.13 and 1.14 illustrate the basic concept of generative and conditional-generative models, respectively.

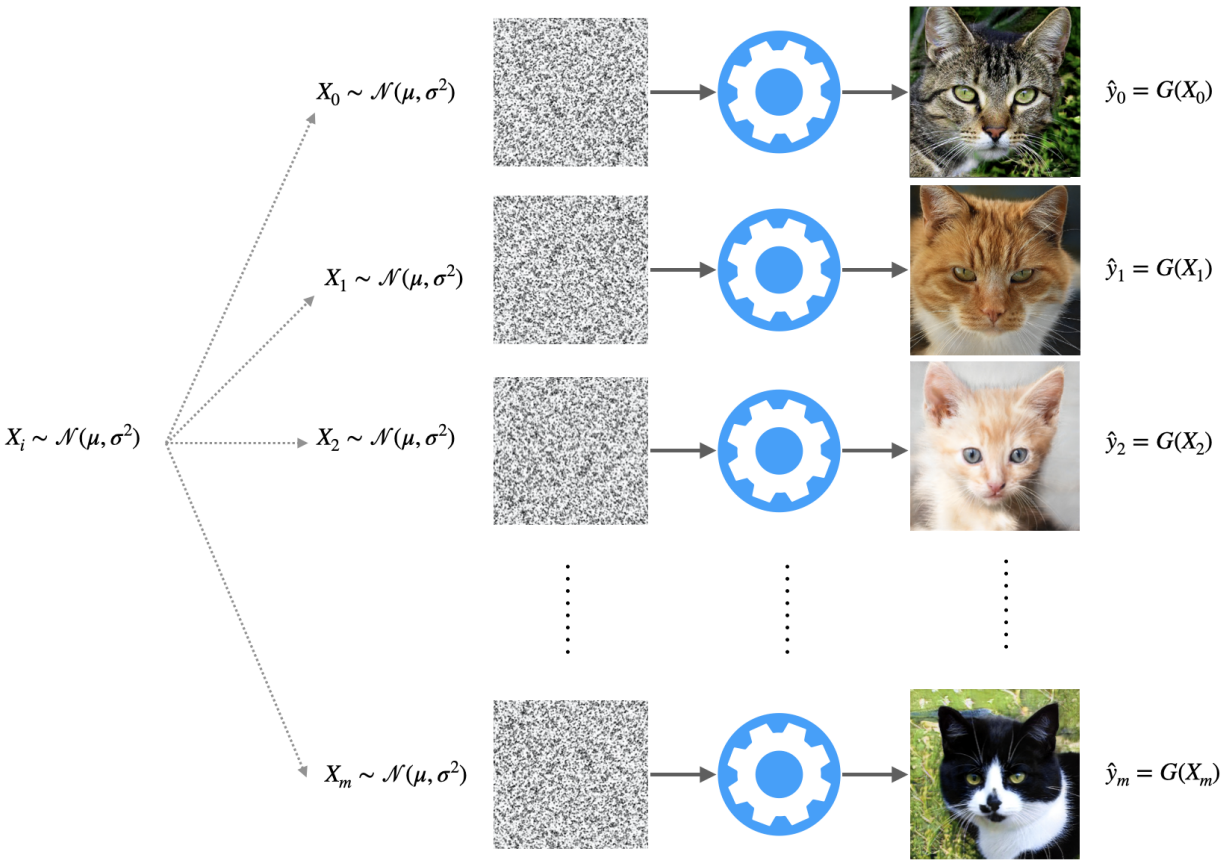


Figure 1.13: **Generative Models: Images from Noise.** Generative models create new data by transforming random noise into realistic outputs—such as images—through a learned mapping. Each input of random noise results in a unique generated image.

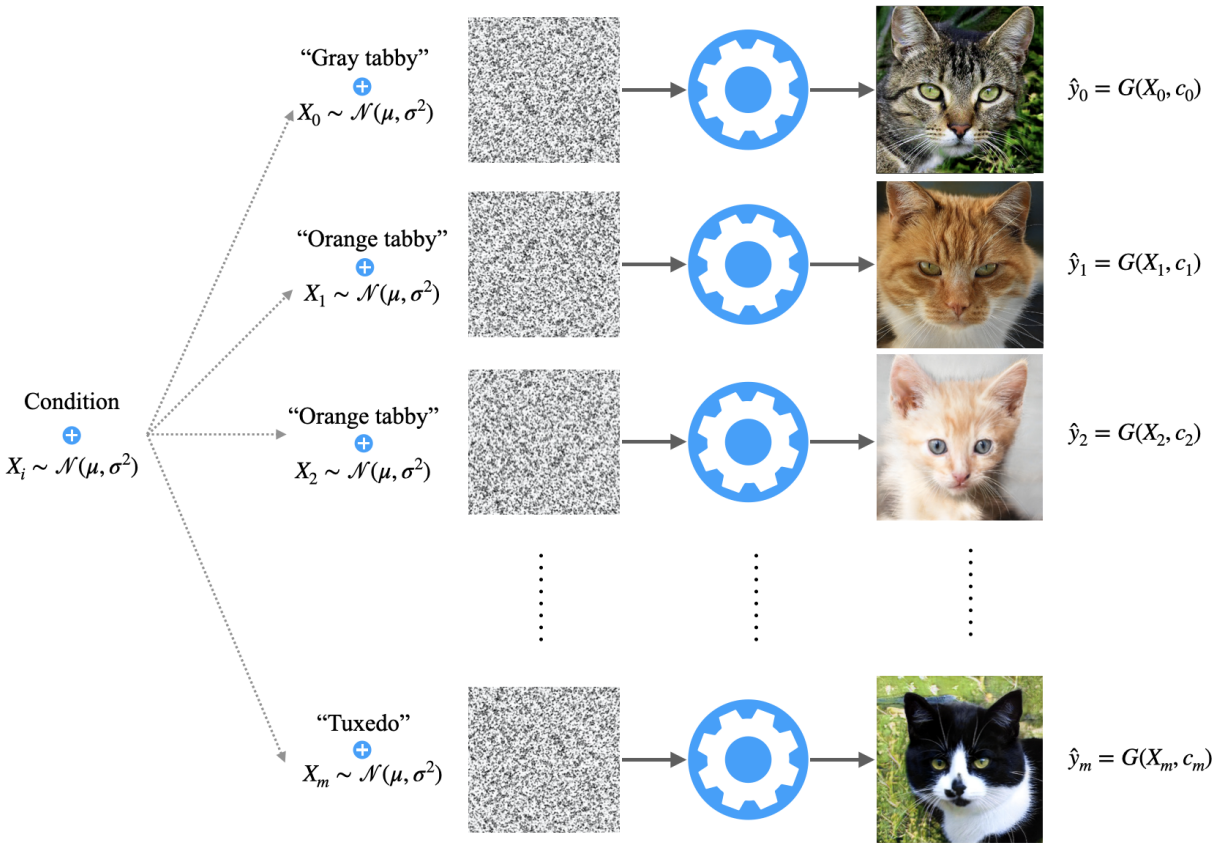


Figure 1.14: ***Conditional* Generative Models: Images from Noise + Condition.** Conditional generative models a variation of generative models in which a second input is passed to the model on which it conditions its generated output. In this example, the images are conditioned on text input describing the desired quality of the generated cats; The condition "gray tabby" results in a generated image of a gray tabby, while the condition "orange tabby" results in an orange tabby, and so on. This allows users to dictate what kind of images they want to generate in a more targeted way than regular generative models. It is important to note that the same condition can lead to two distinct images because new noise is sampled for the input. Here, the same "orange tabby" condition produced two distinct individuals.

Chapter 2

BACKGROUND AND MOTIVATION: DEEP-LEARNING FOR IMAGE SEGMENTATION

Machine-learning represents a fundamental shift in algorithmic problem-solving. Unlike traditional "rules-based" algorithms, machine-learning models are designed to optimize a given task by detecting patterns directly from the data. This data-driven approach has been remarkably applicable for a wide array of problems across domains and data types (such as language, images, signal processing, etc). But what does it actually mean to "learn" from data? How are machine learning models optimized without any explicit directions or information from (human) developers? This chapter first covers machine-learning principles relevant to the challenges in segmentation model development, with a focus on supervised-training for deep-learning-based segmentation models.

2.1 Learning from Data

Algorithm development approaches generally fall into one of two fundamentally distinct categories – classical (or "rulesbased") programming and machine learning based models. There are trade-offs between the two approaches. For well defined problems with exact solutions, rules-based algorithms can be more reliably accurate, and are often faster than their machine learning counterparts. They are also, in general, more interpretable and therefore easier to debug or modify in targeted ways. However, developing generally applicable logic can be challenging, particularly for complex datasets. The diversity and noise inherent to biological systems are therefore extremely difficult to capture successfully with rules-based algorithms. This is a significant limitation on their use in the world of computational biology, particularly for data-intensive work such as large-scale image analyses.

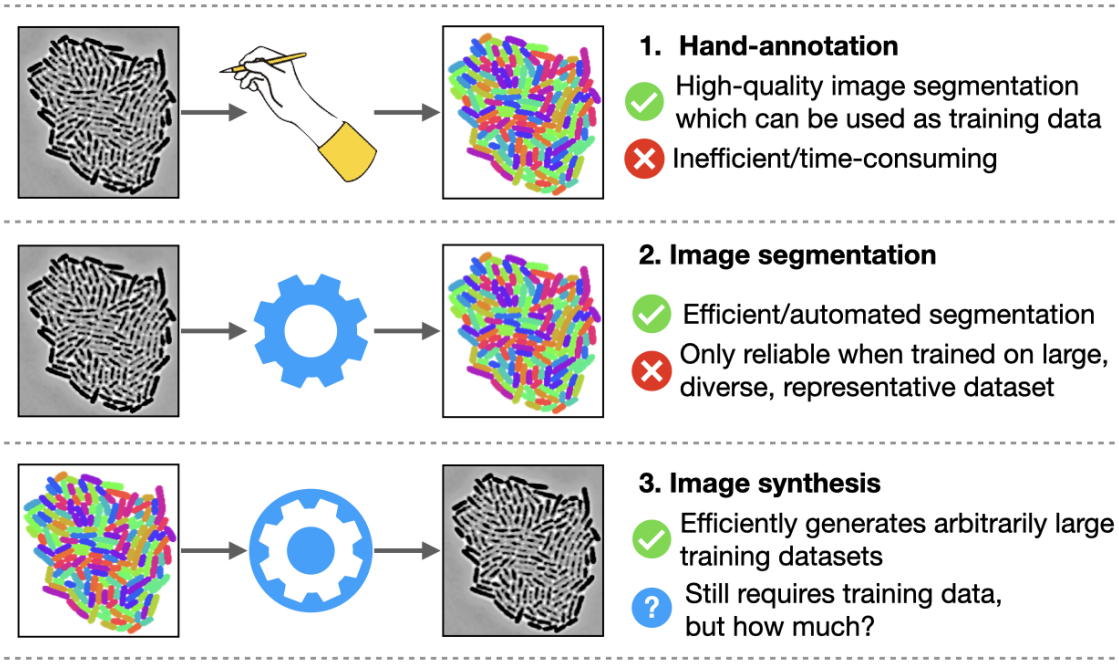


Figure 2.1: **Distinction between three key processes used throughout this thesis** 1) Hand-annotation: manual pixel-level classification performed by domain expert. 2) Image segmentation: model that automatically detects cell regions from an image. For the rest of this thesis, we will focus only on machine-learning based segmentation. 3) Image Synthesis: model that creates synthetic microscopy images from masks.

Machine learning, by contrast, is well suited for high-throughput data tasks due to its inherently data-driven nature. In general, the more data available for training, the better. Machine learning methods have achieved significant breakthroughs in handling increasingly complex tasks and datasets. As a result, deep learning models have demonstrated remarkable success, in some cases matching—or even surpassing—human performance in biological imaging applications [76, 12].

Counterintuitively, data can also be the limiting factor in machine learning applications. Different machine learning approaches have different requirements for the data with which they can be trained, and the quality of any machine-learning algorithm is inherently limited

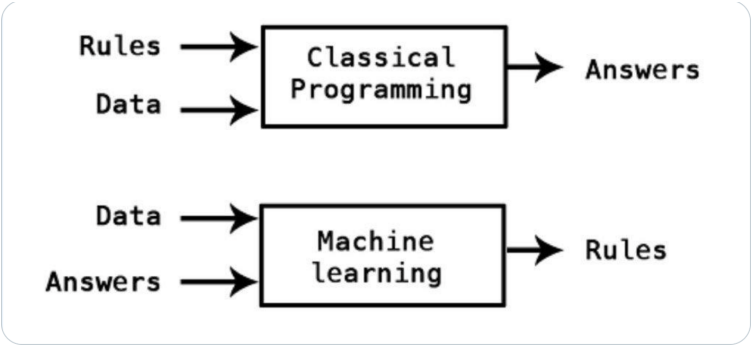


Figure 2.2: **"RulesBased" vs. Machine-Learning Algorithms**[72]. Classically programmed ("rulesbased") algorithms dictate a sequence of (human-defined) steps to solve problems or make decisions, and are optimized by (human) developers to produce the correct output. Machine-learning models are instead the *result* of a rules-based optimization strategy, where the final model parameters have been optimized around a specific dataset.

by the quality (and quantity) of available training data. The specific data requirements for machine-learning depend on the specific approach being used. In this thesis, we use *supervised machine learning* and *generative machine learning* for image segmentation and synthesis, respectively. This chapter focuses specifically on supervised learning approaches for training deep learning models.

Deep learning approaches have proven to be highly effective for the complex task of image segmentation [48]. Most models are trained using supervised machine learning and, once trained, can segment large volumes of images rapidly and with high accuracy. However, it's important to acknowledge that training deep learning models is both time- and resource-intensive. They require large amounts of high-quality, labeled data from which to learn—data that often takes significant time and effort to collect and annotate.

Manual annotation is essential in supervised learning, as it provides the ground truth needed to evaluate and guide model predictions during training. A high-level overview of supervised machine learning for image segmentation is illustrated in Figure 2.4.

2.2 Supervised Learning

Supervised machine learning can be framed, in part, as an error minimization problem. During training, an untrained model iteratively updates its parameters—i.e., it *learns*—to reduce the discrepancy between its predictions and the correct outputs. The ultimate goal is to develop a model that generalizes well, meaning it performs accurately not only on the training data, but also on new, unseen data.

Models that achieve perfect performance on the training set often fail to generalize and tend to perform poorly on test data due to overfitting—a scenario in which the model has essentially memorized the training data without learning underlying patterns. Conversely, underfitting occurs when the model fails to capture the relevant structure in the training data, leading to poor performance on both training and test sets. The ideal solution lies between these two extremes: a model that performs well on both training and test data, indicating a good balance between bias and variance. This trade-off is illustrated in a simple curve fitting example in Figure 2.3.

A general algorithm for supervised learning is outlined below in Alg. 1 that can be applied to a variety of machine learning model types. A (simplified) high level-schematic of how we use supervised learning for image segmentation can be seen in Figure 2.4.

2.2.1 Learning from loss functions

A loss function serves to quantify the prediction error throughout training. It is also commonly referred to as an error, cost, or objective function. Loss functions are a central component of machine learning [17]. Model parameters are systematically updated to minimize the loss function for the data in the training set. Loss curves like those in Fig. 2.5 can be used to monitor the loss on training and validation datasets throughout the duration of training. Logging loss values for the duration is useful for model selection because we can use loss curves to evaluate the performance of the model. In general we look for model convergence. When the training loss no longer improves with more iterations, we can say

Figure 2.3: **Optimal model fitting** [6]. The under-fit curve (left) lacks the complexity needed to model the data, which results in both high training and test error. The overfit curve (middle) matches the existing data points too closely, so while the error on the training set is low, any new unseen data points are likely to have high error rates. The balanced fit (right), strikes the balance between the two extremes, which is what we hope to obtain in model fitting.

the model has converged.

2.3 Deep learning

Deep learning is a subfield of machine learning that has been of significant interest to the machine learning field in recent years [41]. Deep learning models are typically referred to as artificial neural networks (ANNs) or deep neural networks (DNNs). The information flow in an artificial neural network (ANN) is designed to mimic that of the human brain's biological neural network [10, 24]. ANNs use artificial neurons—simplified models of biological neurons—that receive input from neighboring neurons and transmit a signal onward if a certain activation threshold is reached.

The concept of an artificial neural network has been around since the late 1950's [10], but they were too computationally intractable at that time to be used in practice. This has, of course, changed. Powered by the rapid improvements in computational efficiency

Algorithm 1 Supervised Machine Learning.

let N_{epochs} = number of epochs

let N_{batch} = number of batches

for epoch in N_{epochs} **do**

for batch in $N_{batches}$ **do**

- Make model prediction: $\hat{y} = F(\vec{x})$
- Compute loss: $\mathcal{L}(y, \hat{y})$
- Backpropagate and update weights

end for

if stopping condition **then**

skip to end

end if

end for

and performance in GPUs, deep-learning models have become increasingly accessible to developers with relatively limited computing capacity [51].

One of limitations of deep learning approaches in general is that models are notoriously difficult (if not impossible) to interpret due to the nonlinearity of their architectures. This poses significant challenges in model development, as it can be difficult to decipher the source of any unexpected model predictions.

Of particular importance to imaging based tasks are *convolutional neural networks* (CNN), which have been at the forefront of computer vision for many years [42, 40].

2.3.1 Numerical error optimization: Gradient descent

Deep-learning models are highly over-parameterized. Models regularly have millions of parameters to be optimized, and any analytical optimization is therefore impossible. Instead, we use numerical optimization methods to update all model parameters in parallel. The gen-

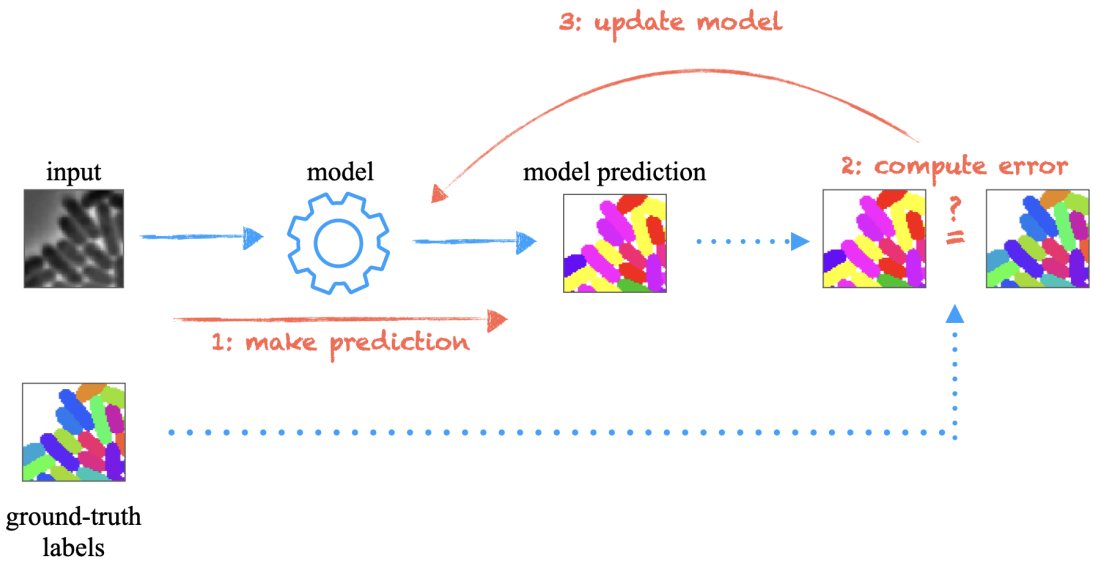


Figure 2.4: **Supervised Learning: predict, compare, update, repeat.** Training a segmentation model using supervised-machine learning involves three basic steps 1) Predict: The model makes a segmentation prediction for the input image. 2) Compare: prediction error is calculated by comparing the the predicted segmentation output to the ground-truth segmentation. 3) Update: model parameters are updated to minimize the error calculated in step 2. All error minimization techniques in this thesis use gradient descent. 4) Repeat: These steps are repeated until model performance converges on the train set (or diverges on the validation dataset, whichever comes first).

eral approach is called gradient descent [62], where we iteratively optimize the loss function by computing the gradient of the loss function with respect to each model weight. Weights are updated by "stepping" in the direction of steepest descent, visualized in Figure 2.9 in three dimensions. Let $\vec{w}$ represent the weights of our model, which we update using gradient descent with learning rate α from its current value at training step t to its new value at training step $t+1$ as follows;

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \alpha \nabla_{\vec{w}} \mathcal{L}$$

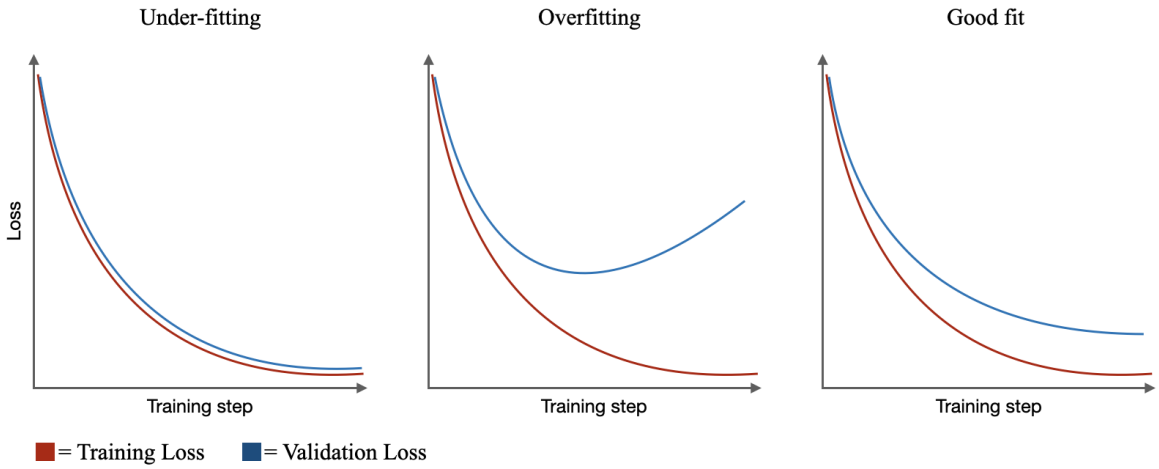


Figure 2.5: **Model selection using loss curves.** In this figure we can see three characteristic behaviors of training and validation loss curves. In all three cases, the training loss appears to be converging, and so we can use the validation loss to evaluate which model has the "best" fit. In the case of underfitting, the training and validation loss are very similar, if not identical. We should expect our model to perform slightly better on training data, so if the model is performing as well on data not used for optimization that is a sign that there is still room for improvement, but because we are reaching convergence the model likely will not improve substantially. In the case of overfitting, the training loss converges while the validation loss diverges. This is a clear sign of overfitting or memorization because the model continues to optimize around the training dataset while making increasingly bad predictions on data not used for optimization. We aim for a model somewhere in between these two states.

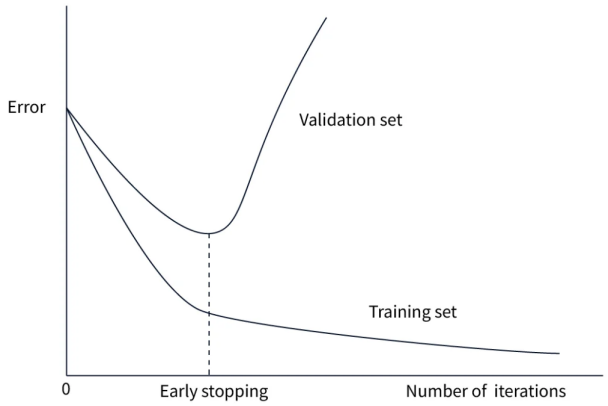


Figure 2.6: **Early stopping** [49]. Early stopping is a method for model selection during training. We track both training and validation losses and save model checkpoints regularly throughout training. This enables a model to train until it starts to over fit at which point we choose the best saved checkpoint.

The above can be expressed for an individual weight w_i

$$w_i^{(t+1)} = w_i^{(t)} - \alpha \frac{\partial \mathcal{L}}{\partial w_i}$$

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_{ij}}$$

2.4 Garbage In, Garbage Out: The Importance of Training Data

Machine learning algorithms are special because they can learn directly from data. This means they are only as good as the data they are trained with. The model learns to model the relationship between the input and output it is shown during training. For example, if we mix up the label in a classification problem, then our model is going to learn to map inputs to the incorrect classes during inference. Therefore, the provided training inputs (data) and outputs (labels) *must* be correct for the model to work as desired.

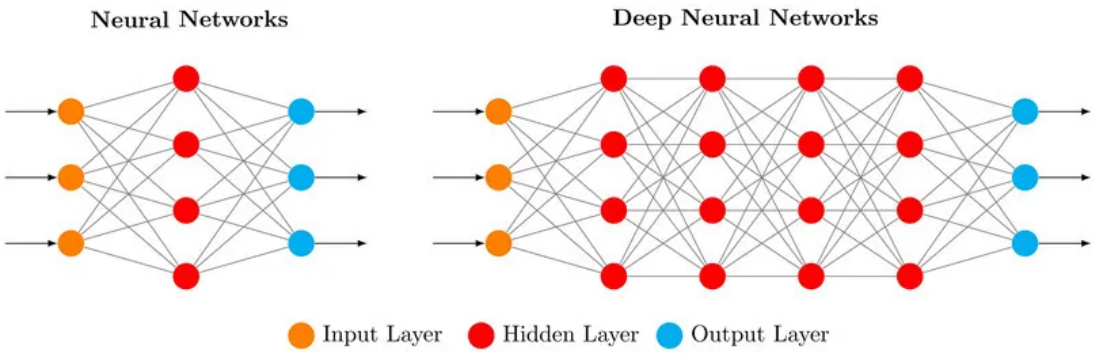


Figure 2.7: **Fully-connected deep neural networks** [75].

Algorithm 2 Forward pass [56]

let N_l = number of layers

for l in range(1, N_l) **do**

if $l = 1$ **then**

$z^{(l)} = w^{(l)}x + b^{(l)}$

else

$z^{(l)} = w^{(l)}x + b^{(l)}$

end if

if l¡N_l **then**

$o^{(l)} = g(z^{(l)})$

else

$\hat{y} = z^{(l)}$

end if

end for

Algorithm 3 Backpropagation [56]

let N_l = number of layers

for l in range(1, N_l =) **do**

  $j = N_l - j + 1$

  if $j = N_l$ **then**

    $\delta^{(j)} = y - \hat{y}$

  else

    $\delta^{(j)} = g'|_{o^{(j)}}[(w^{(j+1)})^T \delta^{(j+1)}]$

  end if

end for

for l in range(1, N_l) **do**

  $\Delta b(l) = [\delta^{(l)}\mathbf{1}]$

  if l=1 **then**

    $\Delta w^{(l)} = [\delta^{(l)} x^T]$

  else

    $\Delta w^{(l)} = [\delta^{(l)} (o^{(l-1)})^T]$

  end if

end for

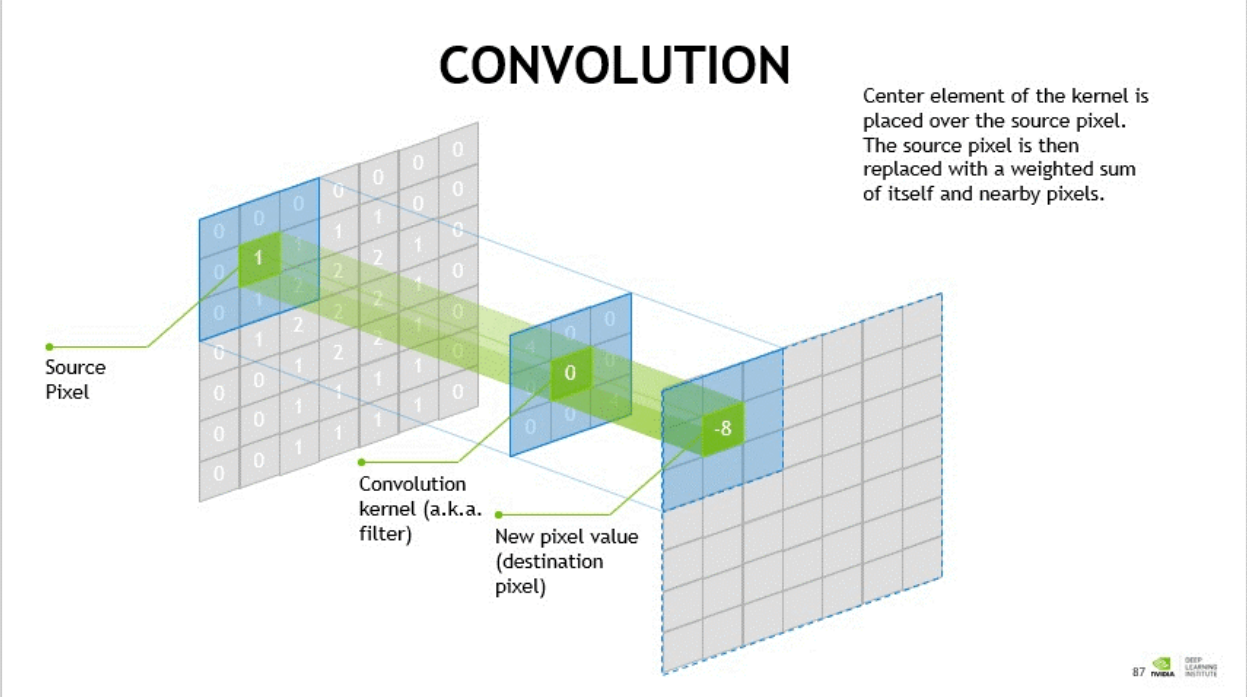


Figure 2.8: **Convolutional Neural Network Layer** [50]. Convolutional neural network layers optimize the elements of a kernel which acts as a filter on the image. The kernel slides across the image, iteratively centering on a new pixel and performing element-wise multiplication with each of the pixels in the current location.

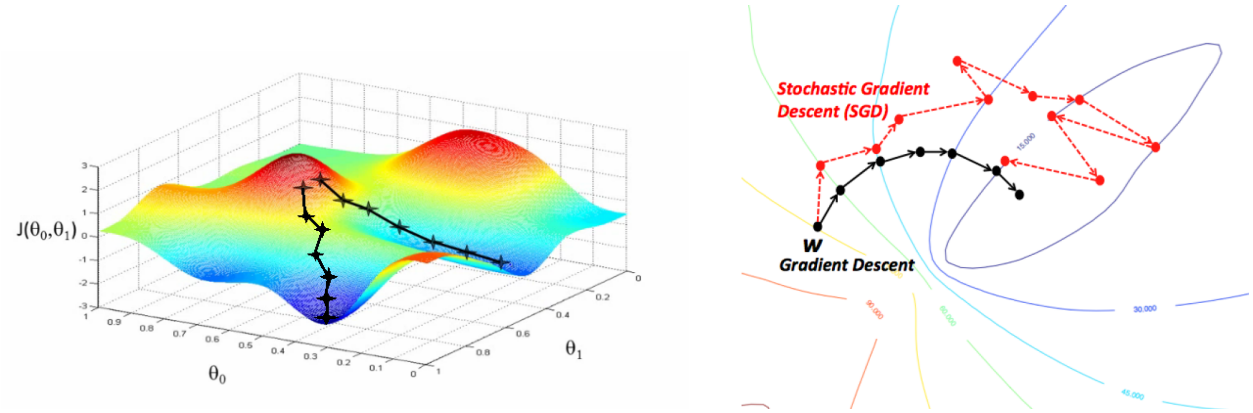


Figure 2.9: **Gradient Descent.** Numerical optimization of machine-learning performance

2.4.1 Data Augmentations

A standard practice in machine-learning is to augment training samples to effectively increase the number of training samples available. Data augmentation has two primary purposes during model training - increasing training dataset size/diversity and improving model robustness to reasonable perturbations in data, both of which can prevent overfitting [59]. Augmentation involves selecting and applying a predefined set of allowable transformations to training samples. Common augmentations for imaging data include rotating, resizing, cropping, warping, or flipping the input image.

2.5 Distribution shifts

Distribution shifts are difficult to describe in real-world scenarios because they depend heavily on the specific context and application domain. Image distributions can shift for many possible reasons such as population shift, device shift, methodology shift, and more. The way these shifts manifest in the data can lead to unpredictable performance in deep-learning based models. This also makes it difficult to measure or predict what constitutes a distribution shift in imaging data, because it is often unclear whether the features that change under

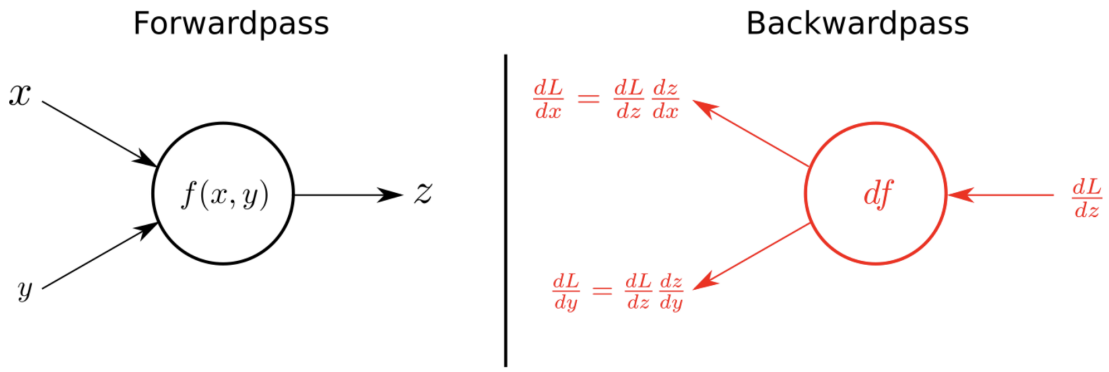


Figure 2.10: **Forward vs. backward pass in neural networks.** The forward pass (left) is used to make model predictions during training and inference - the input is passed "forward" through the neural network and produces the output. During training, the output is compared to the target value and the loss (a.k.a. cost or error) is used to update the network weights (i.e. model parameters). Weights are updated using a process called backpropagation [63], in which the loss function is optimized with respect to *all* tunable model parameters. The backward pass (right) is the first step in backpropagation. To optimize the loss function with respect to a given parameter, we must first determine the gradient of the loss with respect to that parameter. In deep-neural networks, many of these gradients require the chain rule. The backward pass therefore computes the gradients starting with those closest to the model output and works backwards through the model until it reaches the input.

the shift are of strong importance to model output. The black-box nature of deep-learning models makes it impossible to identify and manually modify only the necessary components of the model to adapt to the shift. This is a major motivation in the study of interpretable machine-learning methods which could be directly relevant to the future of work like ours.

Machine-learning models require large, carefully-curated, and *representative* training datasets. Their performance is only as good as the data they are trained on, and many models typically do not generalize well to data outside of the original training set distribution. Variation in experimental condition can lead to sufficient enough changes in key image features that existing models would require retraining on the new data to ensure reliable performance. Due to the significant time and expertise required to assemble such datasets, training dataset curation can therefore (understandably) be a limiting factor for researchers looking to use ML-models in practice. Reducing the burden of training dataset curation would directly improve the usability of ML-based segmentation models and make larger-scale quantitative analyses of imaging data more achievable.

Recently, there has been a focus on developing models that generalize more effectively to out-of-distribution data. In the context of microscopy segmentation, we have generalist algorithms such as Omnipose that are geared toward morphologically invariant bacterial segmentation. While this is indeed an improvement, there is still a heavy reliance on a representative dataset, and variation of imaging modalities still requires users to provide a new set of training samples. This continues to be a rapidly evolving landscape - with new computer vision breakthroughs come even more new approaches for model generalization. For example, the Segment-Anything Model (SAM) [35] is a foundation model designed to — as the name suggests — segment any image. However, these approaches do not *yet* perform with sufficient precision to reliably apply to microscopy images without significant error unless they are retrained on representative data, and are thus subject to the same (albeit lower) data limitations.

2.6 Omnipose: The challenge of morphology

Omnipose [20] is a bacterial microscopy image segmentation package developed to segment cells of any morphology. Based on its predecessor, Cellpose [66], Omnipose uses similar flow-field latent representations of masks during training. These flow fields are especially useful because they allow use to perform instance segmentation without first having to do object detection to distinguish between multiple objects. The flow-field representation is created from training masks and are used as the model outputs. That is, given an input image the model learns to predict the flow-field representation of the segmentation. These flow-fields can then be used to determine cell masks using an Euler integration process. The flow-fields themselves are generated by simulating a heat-sink in the center of the cell and assigning pixels a two-component vector pointing towards that center. The advantage of this approach is that even neighboring cells are distinguishable from one another, and these boundaries can be identified by a positive divergence of flow-fields along that boundary. Omnipose expands on Cellpose by generalizing the flow field generation to handle new and unusual morphologies. Because Cellpose assumes a cell "center" that is located inside the cell boundaries to create flow fields, if that center instead falls outside of the cell, the flow-fields become nonsensical and impossible to learn from. Omnipose on the other hand identifies a cell's central skeleton from the cell mask and generates a flow field pointing towards the skeleton. This allows for long cells to be segmented more effectively, as the flow-field will be more consistently orthogonal to the boundary of the cell, which is important for the identification of cell boundaries, especially in the case where two cells are in contact.

2.6.1 Representative data is critical.

Training data plays a central role in the development of effective machine learning models, and it is critical for that training data to closely resemble the target data. This becomes increasingly challenging in contexts with diverse datasets, as model generalization often comes at the cost of reduced model performance. Even Omnipose, which has been specifically

engineered to handle diverse morphologies, still relies heavily on representative training data. This is illustrated in Figure 2.18, which shows the segmentation output of an Omnipose model trained on *only* the short morphologies in the Omnipose dataset.

The canonical solution to this problem is retraining with new representative data; however, this once again requires time-consuming hand-annotation by an expert. Such limitations pose a fundamental problem in the utility of deep-learning models in biological research contexts, as any existing automated segmentation model is undermined by the need for new hand-annotated training data whenever new phenomena are observed or a different imaging modality is used during data collection. Reducing the number of hand-annotated images necessary for training (or retraining) is therefore an important step in developing segmentation models that are actually usable in research settings.

2.7 Limited data means limited applications

By now we know that limited training samples is a common bottleneck in applied machine-learning settings, particularly when working with diverse datasets. Generally speaking, models trained with more diverse the datasets require more training samples to ensure the training data is truly representative. As discussed in the previous chapter, this poses a fundamental challenge in biological contexts with effectively endless sources of data diversity. In practice, this leaves researchers in a constant state of data-collection to fine-tune models for their own purposes, unless they happen to find an existing model that has been trained on data similar to their own.

In the case of segmentation, hand-annotation of training data is exceptionally time-consuming as compared to the annotation process for other image-based machine-learning tasks (such as image classification or object detection) because segmentation requires annotations for every individual pixel in every single image. High-resolution imaging applications make this situation worse. Individual pixels carry more weight in the overall accuracy of segmentation than in standard segmentation applications with lower precision requirements or where objects are made up of more images, and therefore have more room for error if a

few pixels are missed during segmentation. Furthermore, microscopy images can be challenging to interpret, at times even for experts. Hand-annotation of microscopy imaging data therefore requires expertise in the specific imaging domain, which makes image-annotation a difficult task to outsource without significant training.

The next chapter will cover our approach to reducing the training data curation bottleneck, specifically for diverse morphologies. Recent work has been done on using synthetic images to increase the size of imaging datasets with a similar approach [74], but we take this a step further and examine the extent to which synthetic images can be used in the face of morphological distribution shifts.

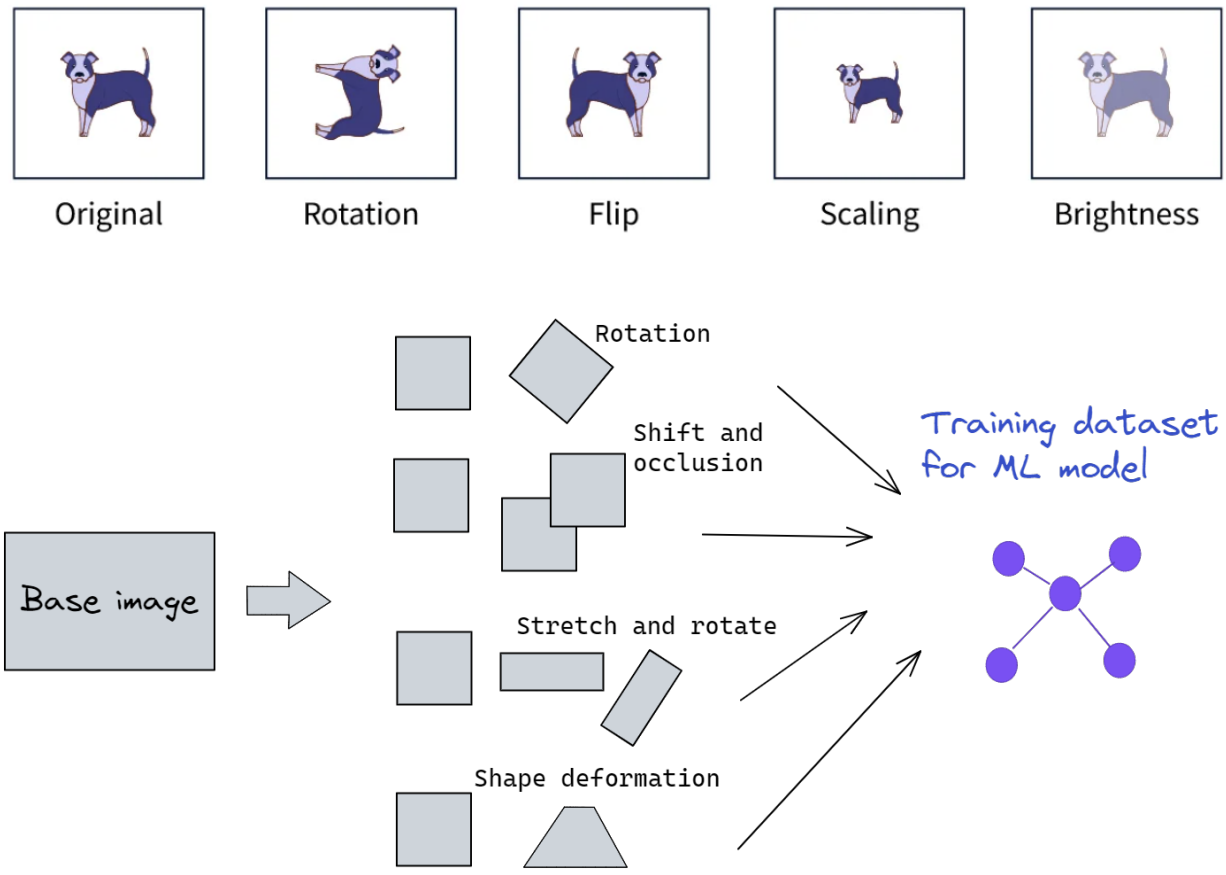


Figure 2.11: **Data Augmentation.** Top [49]: Examples of standard image transformations used for data augmentations during training. Bottom [18]: Data augmentation can be used to increase the effective size and diversity of datasets.

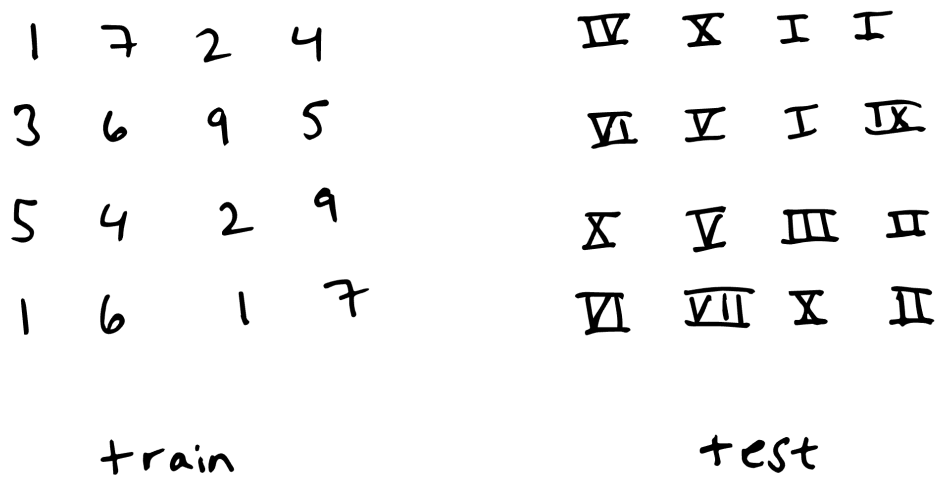


Figure 2.12: **Example Distribution Shift** [7] A model trained on digits will not successfully classify roman numeral representations of those same numbers. A generalized approach would train a model to identify them both successfully, which would involve training on data from the roman numeral dataset in addition to the original dataset.

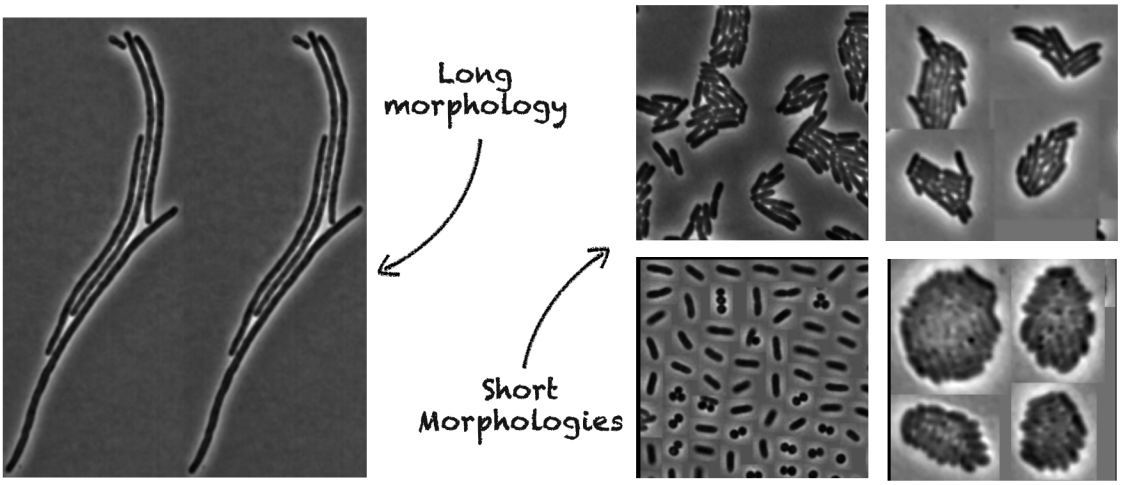


Figure 2.13: **Distribution Shift in Cell Morphologies.** Morphology is a high-order image feature, therefore new morphologies introduce a distribution shift in our datasets, even within a single imaging modality.

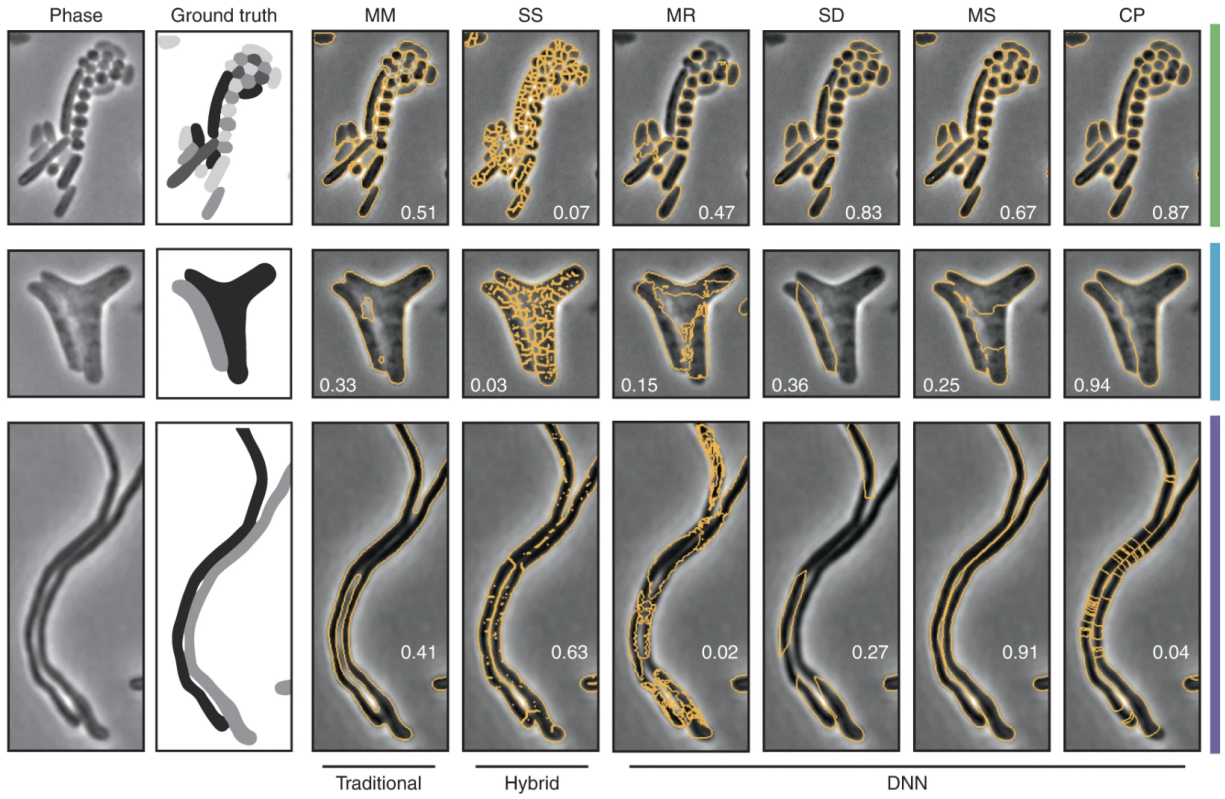


Figure 2.14: **Diverse Data Dilemma** [20]. Diverse morphologies are challenging to segment, even for machine-learning based approaches

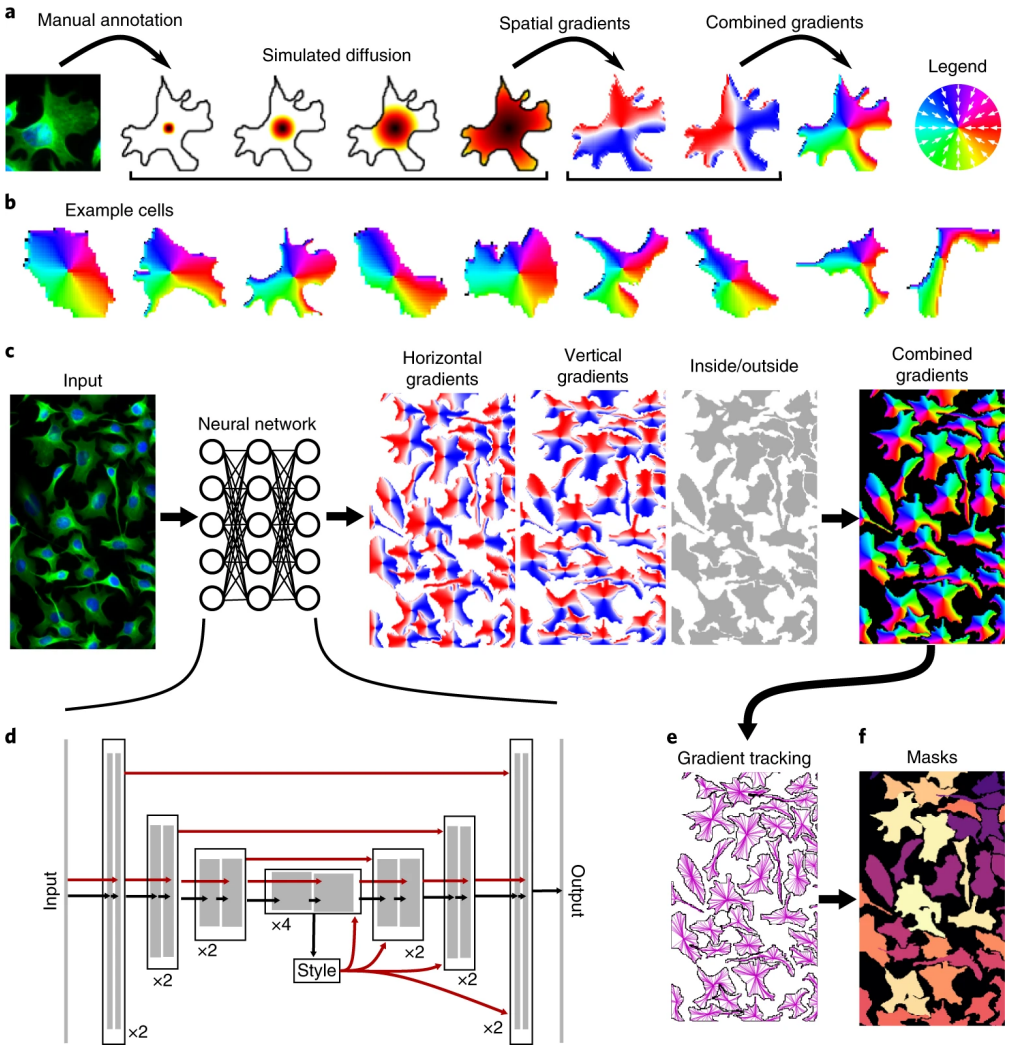


Figure 2.15: **Training with latent mask representation** [66]. This image is from the original Cellpose paper, and shows the process for generating and training with a latent representation of single-cell masks. The idea of optimizing this latent representation for more effective and efficient model optimization is a critical concept for Omnipose and our work with synthetic images, as it allows for a representation of single cell masks that is compatible with deep-learning model training. The model learns to predict the latent variable, which can then be used to identify single-cell regions by following flow lines.

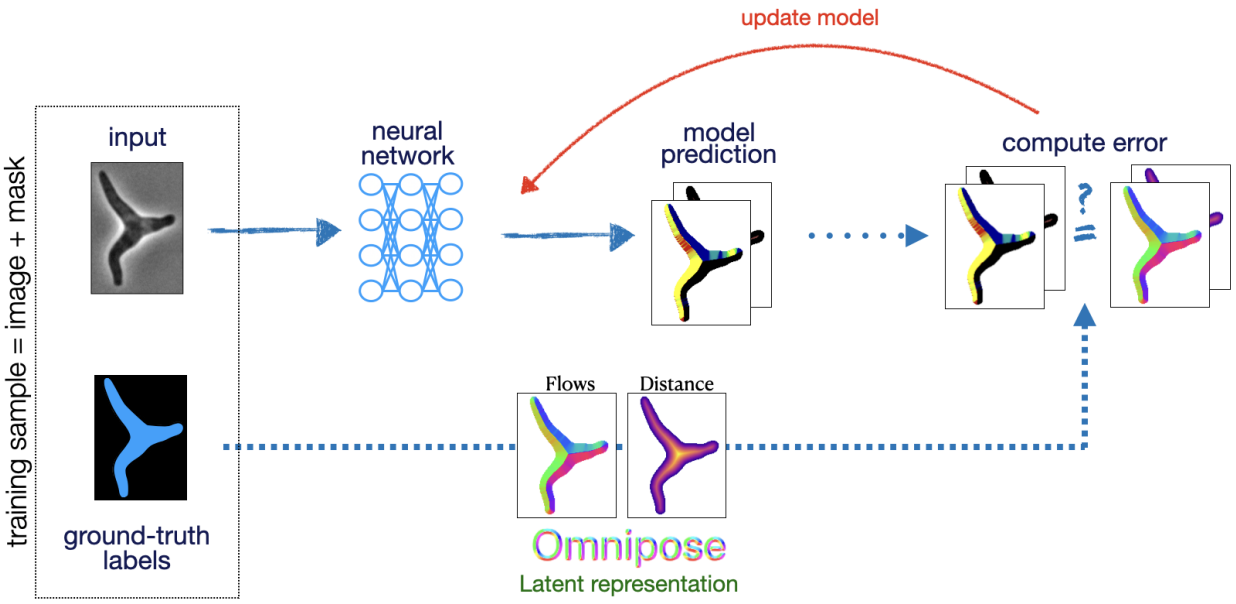


Figure 2.16: **Omnipose uses latent mask representations.** Connected cell region labels are not actually suitable for training machine-learning models. The arbitrary label ID numbers have no real meaning to the deep learning model, so we transform these masks into a latent representation that is compatible with the neural network but also maintains the integrity of the masks.

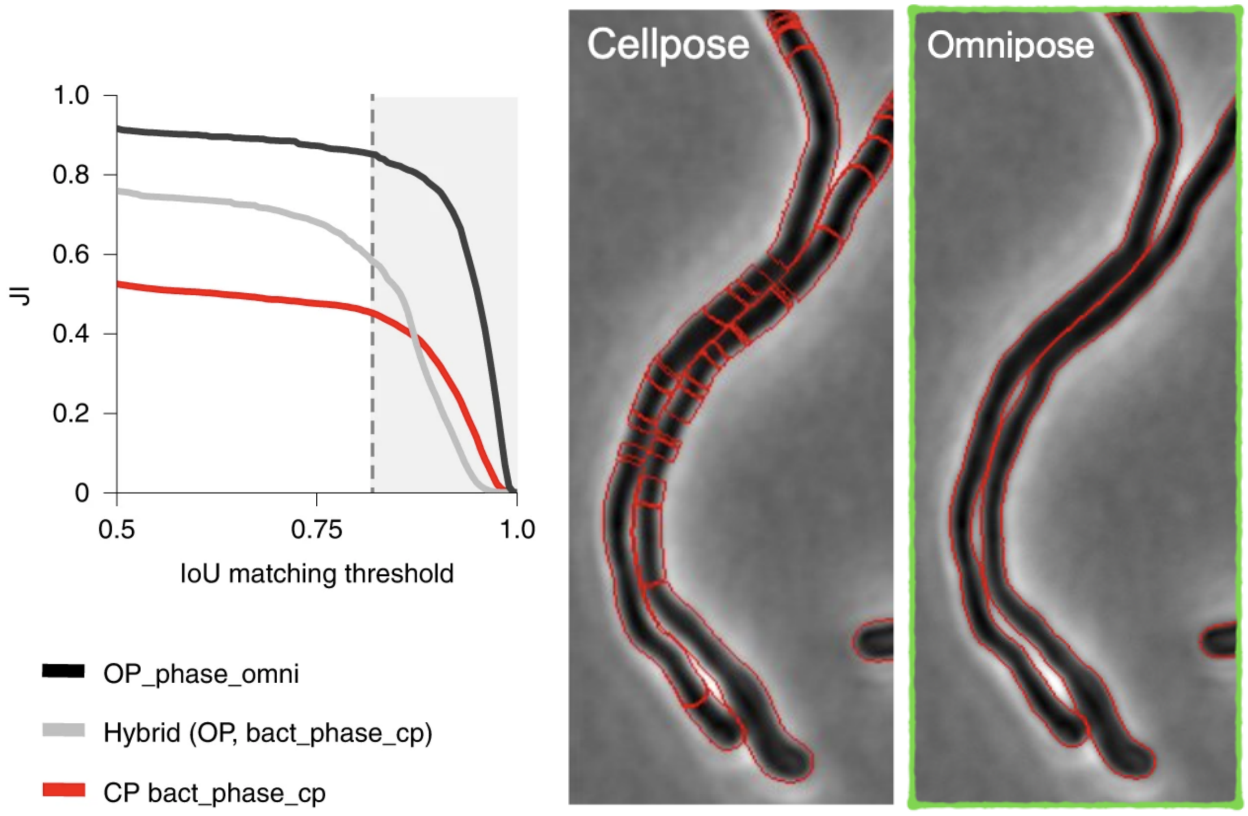


Figure 2.17: **Omnipose for morphological invariance in segmentation** [20]. When trained on extensively diverse morphologies, Omnipose is able to reliably segment similar morphologies in test images. Here we see a direct comparison to Cellpose [66].

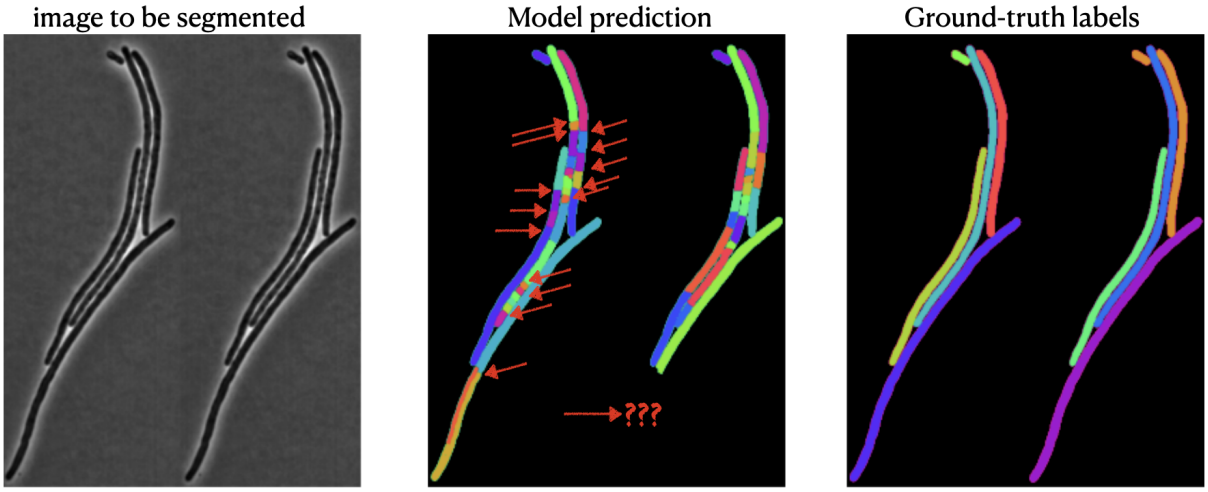


Figure 2.18: **Representative training data is critical.** Omnipose trained with short cells over-segments long cells.

Chapter 3

BACTERIAL DEEPFAKES: GENERATING SYNTHETIC TRAINING SAMPLES

In this chapter, we outline our approach for generating synthetic bacterial microscopy images and examine the extent to which real samples can be replaced and/or supplemented with synthetic samples during training. We utilize generative machine-learning to develop a model that produces new images to use as training data for downstream segmentation models. Generative machine-learning models are intended to produce brand new instances of the data on which it was trained. This is fundamentally different from most machine-learning tasks, in which the model learns to exploit patterns in data to make accurate predictions. In generative learning, the aim is to generate "new" samples, so the error of a generated output is poorly defined. Ultimately, the generative models are trying to learn the underlying distribution of the training data to successfully produce a "new" sample from that distribution rather than simply replicating instances from the training set.

Our aim is generate synthetic bacterial microscopy images. Image segmentation and image synthesis are effectively inverse tasks. The image synthesis model is based on a style-transfer algorithm, in which an image is transformed from one 'style' to another. We use a conditional generative-adversarial network (cGAN) because it allows us to synthesize *paired* data by transforming images of predefined single-cell masks into plausible microscopy images that are consistent with the given masks. This allows us to create paired image-mask samples which can be used to train the segmentation model.

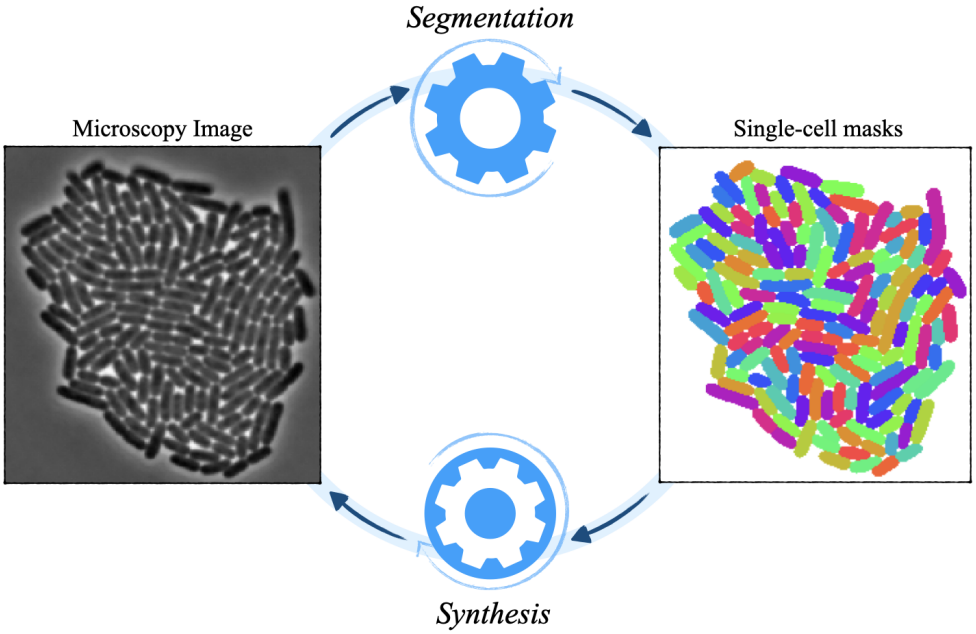


Figure 3.1: **Synthesis = Segmentation**$^{-1}$

3.1 Overview of Approach

The purpose of our work is to maximize the utility of existing annotated training samples. This approach is not intended to replace hand-annotated samples altogether, but rather to reduce the overall number of annotated samples required for training a successful segmentation model. Supervised machine-learning models require *annotated* ground-truth training samples, so we need to generate data *with* labels.

Specifically, we need to generate synthetic images *and* their corresponding ground-truth masks. It is critical that the images be consistent with the ground-truth masks for downstream segmentation training. This can be achieved by generating images from single-cell masks, which is effectively the inverse problem of segmentation.

For a given annotated dataset of real microscopy images, we train two models. First, we train a model to generate synthetic images from ground-truth masks. We then use that

model to generate synthetic training samples to train a segmentation model.

3.1.1 Style Transfer

Style transfer is a generative task in which the content of an input sample (such as image, video, text, or audio etc.) is transformed into a new "style," as illustrated in Figure 3.3. This can be accomplished with numerous approaches, but we will focus specifically on conditional generative adversarial networks. The degree to which the content image is altered to match the new style can be tuned in the algorithm. In our case, it is critical that the structural content of the input is preserved because we need to generate an image that is consistent with ground-truth mask inputs. The basis for our approach comes from the Pix2pix image-to-image translation model [31], which was used in the work on material data mining [44] which inspired this project.

3.1.2 Generative Adversarial Networks

Generative-adversarial networks [26] (GANs) are a type of model used to generate "new" instances of data. The "adversarial" nature of these models lies in how they are trained. The model is comprised of two networks, a generator network which learns to produce the sample, and a discriminator network which learns to differentiate between real samples and the fake ones produced by the generator. Like an arms race, the idea is that as each network gets better at its own task, the other has to work harder to do its job successfully; as the generator produces increasingly realistic samples, the discriminator must learn increasingly sophisticated differences between the real and fake samples. Conversely, as the discriminator gets better at distinguishing real from fake, the generator must learn to mimic those features that the discriminator is using to determine which samples are real.

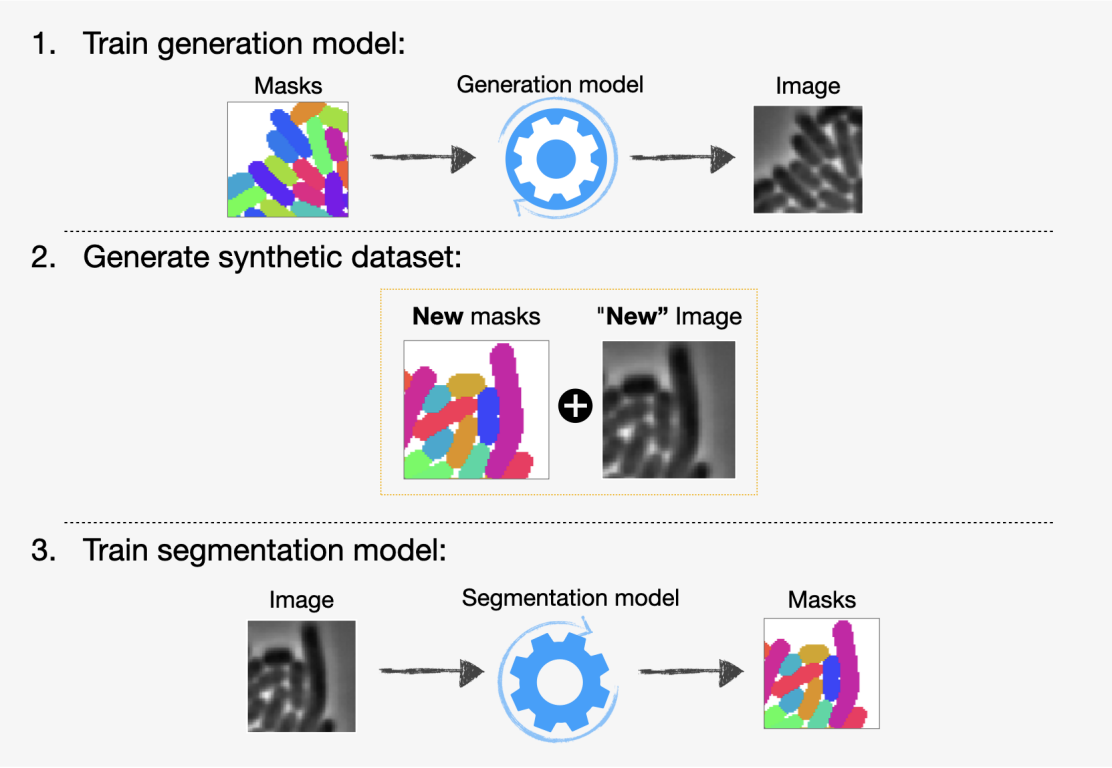


Figure 3.2: **Synthetic training data.** Overview of approach for training segmentation models with synthetic data samples. (top) The generator is trained using exclusively *real* training samples. (2) We then use *new* masks to generate synthetic data. At this stage, we can opt to use masks that are similar to the original training set (in-distribution morphologies), or masks that are different (out-of-distribution) from those in the training set to increase diversity. (3) Combine real and synthetic samples to use as training samples for training the segmentation model.

Algorithm 4 Minibatch stochastic gradient descent training of generative adversarial nets, as outlined in the original GAN paper [26]. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, ..., \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{data}(\boldsymbol{x})$
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log\big(D(x^{(i)})\big) + \log\left(1 - D\Big(G\big(z^{(i)}\big)\Big)\right)\right]$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, ..., \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\Big(G\big(\boldsymbol{z}^{(i)}\big)\Big)\right).$$

end for The gradient-based updates can use any standard gradient-based learning rule.

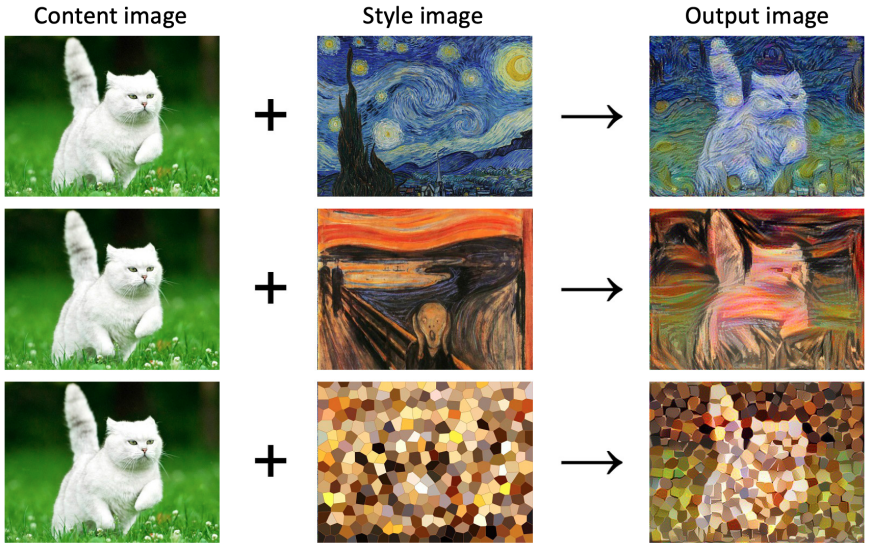


Figure 3.3: **Example of Style-Transfer Concept** [9]. The content image (cat) is transformed into three distinct styles as defined by the style image. The resulting stylized images are very different from one another, but clearly share the same subject. Importantly, this allows us to generate *new* images in the desired style that did not exist previously, which makes this a generative machine learning problem.

3.1.3 Conditional Generative Adversarial Networks

Conditional-generative-adversarial (cGANs) networks are a slight variation on the traditional GAN in which the generated sample is conditioned on some user input in addition to the typical GAN noise input. The generator produces a sample conditioned on that user input, and the discriminator must decide whether the generated (fake) sample is plausible *given* the user input. This requires "paired" or training samples, that is, a sample and it's associated ground-truth label.

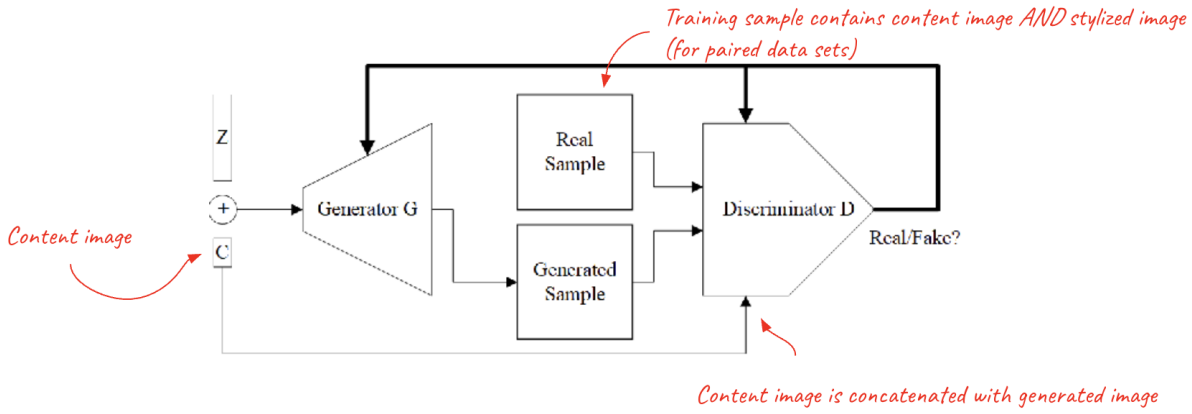


Figure 3.4: **Conditional Generative Adversarial Network**.

3.2 Image Synthesis Results

Model selection for generative models is far less straightforward than for those trained with supervised methods. Loss curves are noisy and uninformative when it comes to model selection [37]. As we will discuss more thoroughly in the next chapter, we are actively developing better strategies for model selection with improved generalization across imaging datasets. At this stage, however, we ultimately selected our model based on which images looked the most convincing to members of our group. A qualitative comparison of between in-distribution and out-of-distribution synthetic samples is shown in Figure 3.7. Perhaps unsurprisingly, the in-distribution sample is more visually convincing. The out-of-distribution sample appears much smoother than the real image, particularly inside cell regions. In real images, a single-cell region typically exhibits more random variation in pixel intensity due to, for example, internal cell structures, which can often lead to over-segmentation. However, many of the other important features are still present, such as the bright halos surrounding microcolonies and bright edges between most neighboring cells.

It is important to note that other generative models such a diffusion models have also been used to great effect, and have been shown to produce higher quality images than those

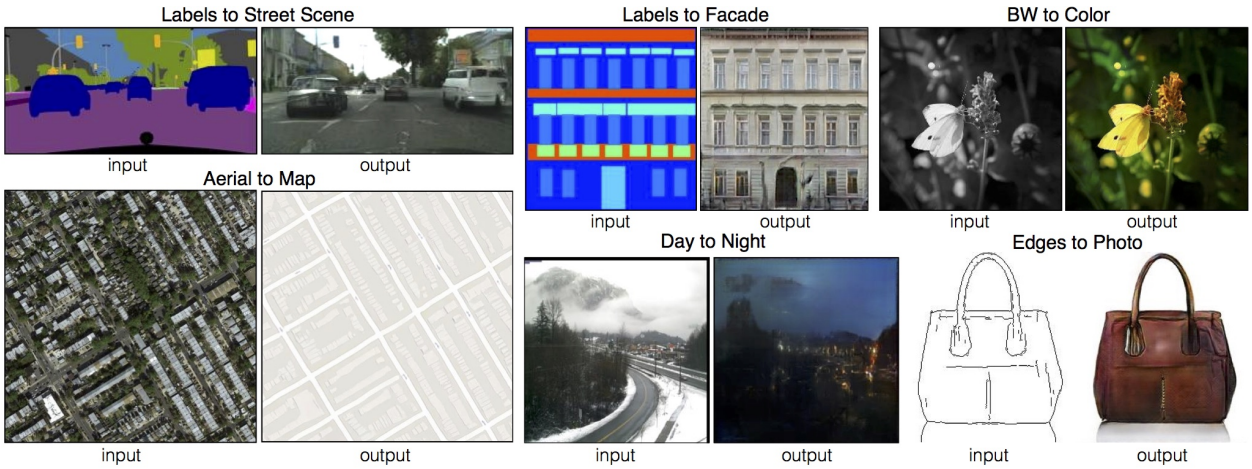


Figure 3.5: **Pix2pix: Style Transfer with cGAN** [31].

generally produced by adversarial learning [23, 22]. A major advantage of our approach, however, is the relative size of the model we are working with.

3.3 Segmentation results

It is important to remember that our ultimate goal is to improve image segmentation. We can quantify the quality of generated images by their impact on segmentation performance for both in and out of distribution test samples. We use segmentation performance as a proxy for measuring synthetic image quality. Specifically, we investigate how segmentation performance is affected by artificially increasing the size and diversity of existing datasets with synthetic images. There are approaches for determining how similar a synthetic image is to the real image, but there is no way to know if we are successfully reproducing the *relevant* features. There are likely a lot of features that are entirely irrelevant to the image segmentation task, but we have no direct way of identifying which ones are important due to the black-box nature of neural networks. The nonlinearity that makes neural networks so powerful also makes them very difficult to interpret. There is a lot of work being done to better understand and direct specific model features, but this again is complicated by the

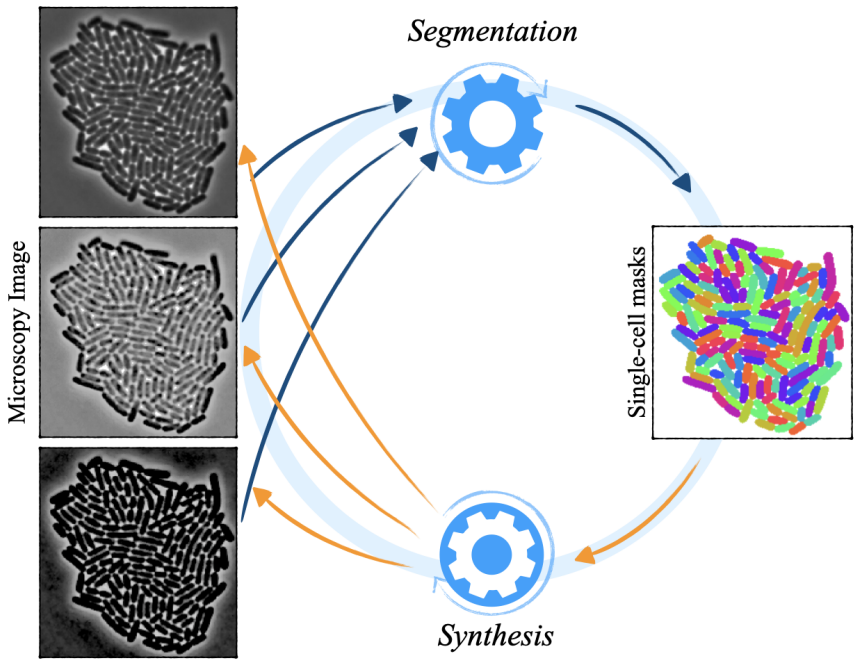


Figure 3.6: **Many-to-One vs. One-to-Many** Image segmentation is a many-to-one problem whereas image synthesis is a one-to-many problem. A single set of masks could plausibly correspond to any number of images, whereas a given image only has one correct set of ground-truth masks.

amount of noise and diversity in our datasets. Our approach of using segmentation performance as a proxy for image quality allows us to bypass this stage entirely, and is ultimately the metric we care about. To that end, we need to compare segmentation performance across models trained on different datasets (real, synthetic, and mixed).

3.3.1 Segmentation Performance Metrics

Before we can compare models, we must first ask: what constitutes a good segmentation? For a test sample with known ground-truth masks, what metrics best quantify the quality of the masks predicted by our segmentation model? This turns out to be a fairly complex question; indeed, establishing standardized benchmarks for segmentation of diverse biological images

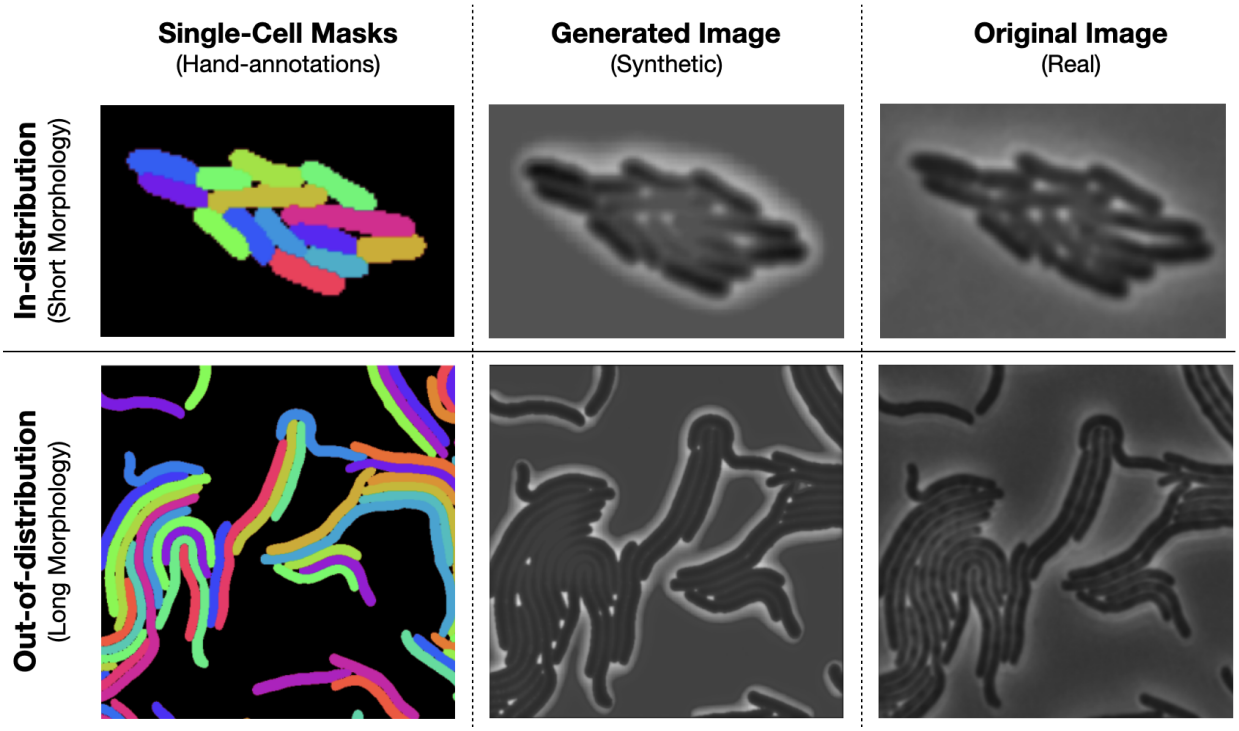


Figure 3.7: **Examples of Generated Images.** We train Omnistyle using *only* images of real cells with short morphologies. After training, we can generate synthetic in-distribution images of short cells (top row) or out-of-distribution images of long cells (bottom row). In this example we use the hand-annotated ground-truth masks (left column) of real test images from the Omnipose dataset as our input masks for the generator model. We can then directly compare the synthetic samples (middle column) with the corresponding original images (right column).

is an active area of research [39]. That said, we use a standard metric called the *Jaccard Index*, otherwise known as the intersection over union (IoU) between predicted and ground-truth masks. We consider a cell correctly segmented if the IoU reaches the desired precision, see Figure 3.8. The IoU between two sets (in our case, sets of pixels) A and B is defined as follows;

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

For a given test image, we compute the pairwise IoU between all predicted and ground truth masks and select the best match for each ground truth mask based on which predicted mask overlaps with it most. If that match reaches or exceeds the desired precision threshold, it is counted as an accurate segmentation. A perfect match results in an IoU of 1, while a complete miss results in an IoU of 0.

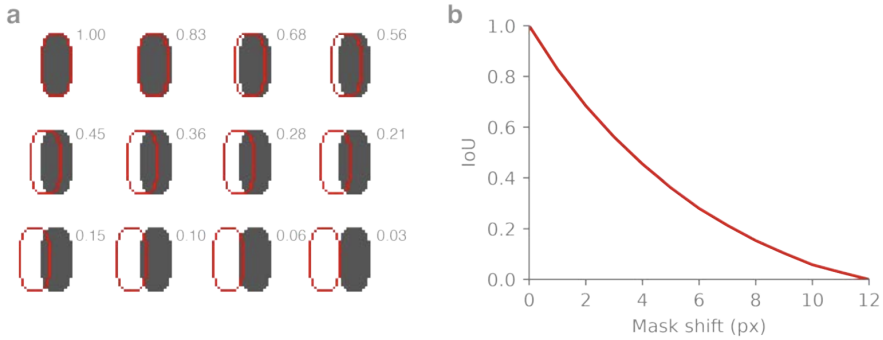


Figure 3.8: **Intersection over Union** [19]. $IoU(y, \hat{y}) = \frac{|y \cap \hat{y}|}{|y \cup \hat{y}|}$, also known as the Jaccard Index, is used in segmentation to measure the similarity between ground-truth masks (y) and those predicted by the model ($\hat{y}$).

3.3.2 Segmentation of out-of-distribution data

In general, models perform best when trained on large, diverse datasets. One of the primary challenges in machine learning is maintaining high performance when evaluating models on out-of-distribution data - data that is different from that used to train the model originally. In general, models perform worse on out-of-distribution samples, and therefore require re-training with new data from the new distribution. As discussed previously, curating training samples for segmentation models is very time consuming and requires domain expertise. Therefore, our aim is to improve the out-of-distribution performance of our segmentation models *without* hand-annotating any additional samples.

To test this, we carried out an experiment in which we use real masks exclusively so that we can isolate the effect of synthesizing the image style and begin to understand the relative information content of a synthetic image. The image synthesis model is trained using exclusively real, hand-annotated samples from the in-distribution dataset. After training is complete, we generate out-of-distribution samples by using out-of-distribution mask morphologies as the input for the image synthesis model. The synthetic out-of-distribution samples are then added to the original dataset of real in-distribution samples to create a combined training dataset to train the segmentation model.

Once trained, the image synthesis model generates synthetic training samples using new (unseen), user-provided masks. The masks do not necessarily need to resemble those of the original training set. For instance, a synthesis model trained exclusively on short, rod-like cells can be given an input with long, worm-like masks, thereby allowing us to artificially increase the morphological diversity of our segmentation training dataset.

3.3.3 Effect on In-Distribution Data

While increased sample diversity may help a model perform well over more kinds of data (i.e. the performance across on more diverse test sets), the trade-off is that it may perform less well on any single sub-class of that data. For instance, adding out-of-distribution samples

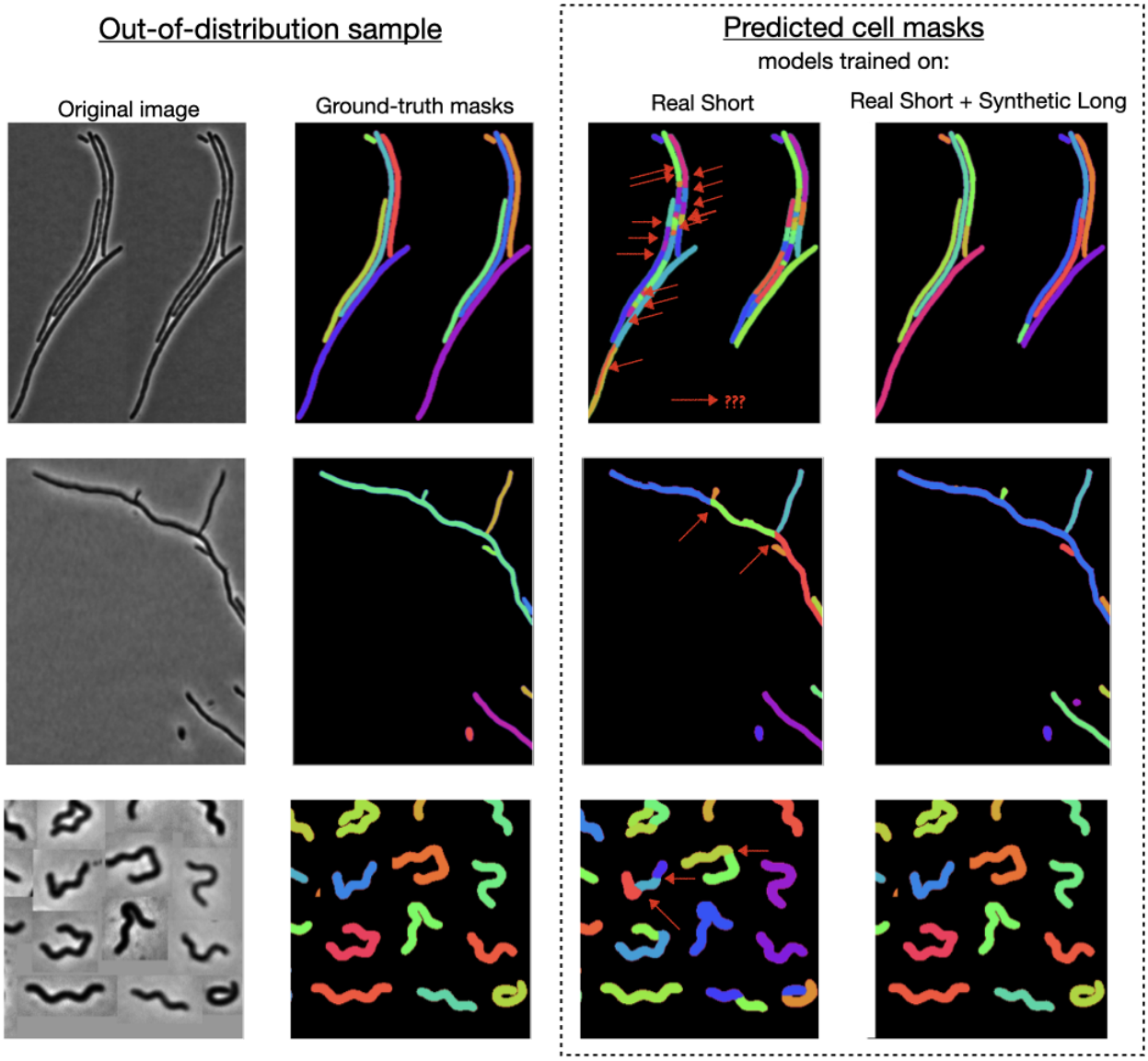


Figure 3.9: **Synthetic data improves segmentation of diverse morphologies** We can see a qualitative improvement in segmentation performance when the model is trained using synthetic long cells. There are still some errors, but the overall segmentation is much better. Note that this required *zero* real samples of the new out-of-distribution data and therefore *zero* additional hand-annotations.

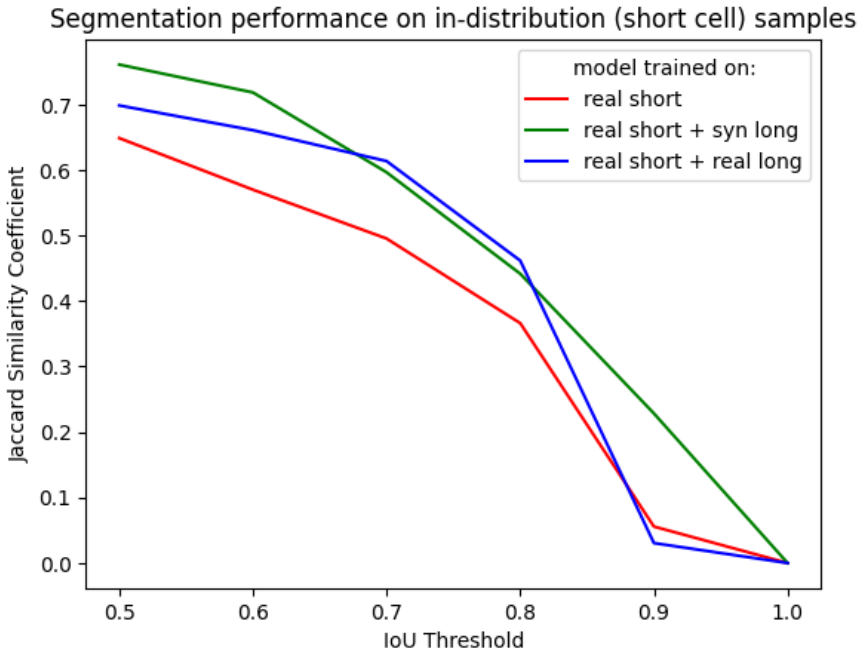


Figure 3.10: **Segmentation Results for Out-of-Distribution Images.** This figure compares the overall segmentation performance on unseen, out-of-distribution images for three identical models, each trained on a different dataset. The test images come from the Omnipose test set, where we have selected the subset of all images with long cell morphologies.

to a training set can cause the model to perform worse on the original in-distribution data. This is indeed what we observed in our results, shown in Figure 3.11, in which any addition of long cells into the training set leads to a decrease in performance.

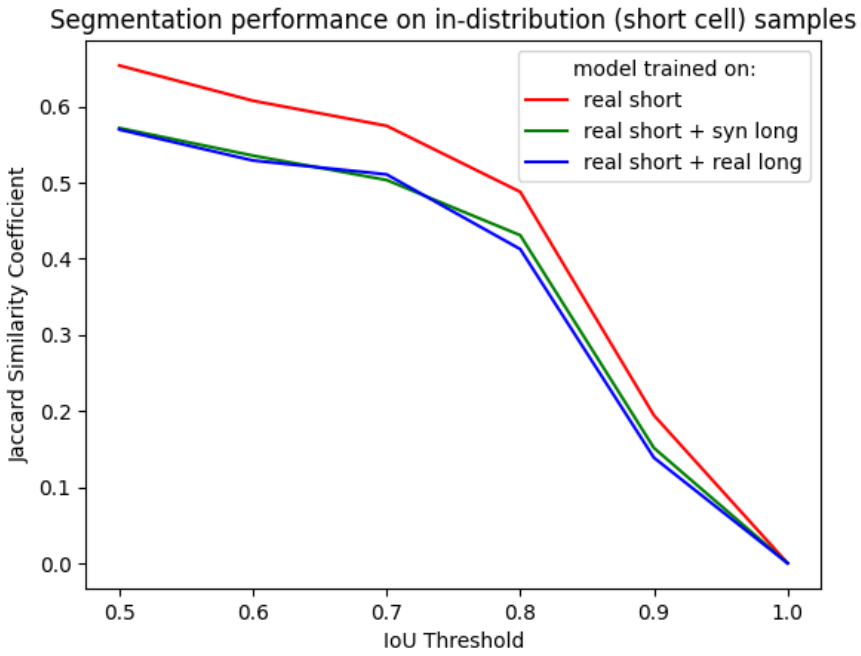


Figure 3.11: **Segmentation Results for In-Distribution Images.** The addition of any long samples, real or fake, leads to a decrease in performance on short cells. This is perhaps somewhat unsurprising, because the addition of any new morphologies (in this case long cells) into the training set introduces new allowable or plausible high-order features the model predictions.

3.4 Discussion: Hierarchical Image Features

Image features are hierarchical. Low-order features have local information, the lowest-order feature being a single pixel. A collection of pixels (lowest-order) make up the next-lowest-order features such as edges, corners, or curves. Collections of these features in turn gives the one-order higher features and so on and so forth until we reach high-order features such

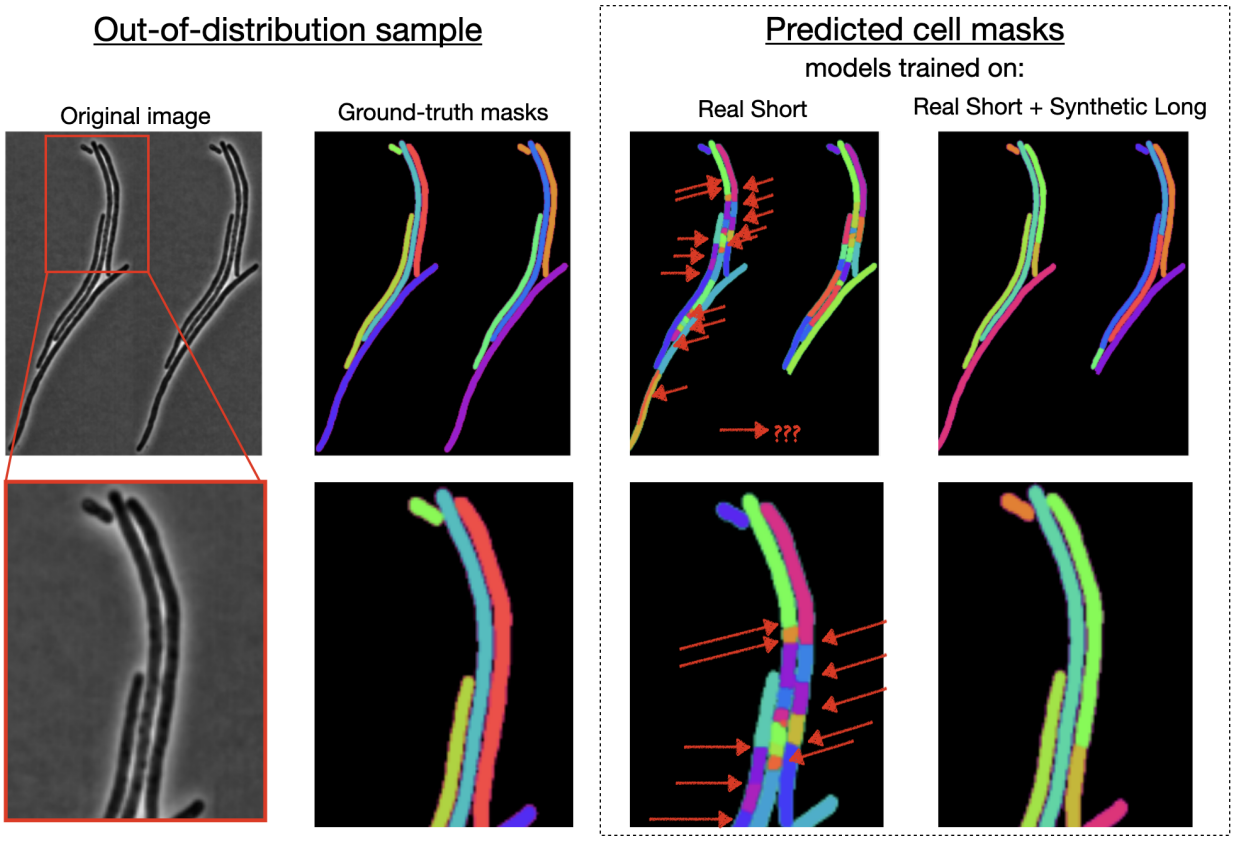


Figure 3.12: **Synthetic images capture high-order feature shift in training data.**

as complex shapes or even recognizable semantic objects. In deep-learning, we cannot easily probe models to determine which features have been deemed important, or even really how or where those features are encoded in the network. Most imaging tasks involve convolutional neural networks (CNNs) [42], and generally each subsequent convolutional layer roughly corresponds to another higher-order of features being extracted from the image [60].

Chapter 4

OMNISTYLE: IMPLEMENTATION DETAILS

This chapter covers important details of our image synthesis algorithm, Omnistyle. Omnistyle is a cGAN-based style-transfer model that has been optimized specifically for generating synthetic bacterial microscopy images. Our model implementation is based on that of pix2pix [31], a style-transfer algorithm that utilizes a conditional generative adversarial network to transform between two image styles. Our model has been modified from the original pix2pix algorithm to handle high-resolution microscopy images, which requires a significantly higher degree of precision due to the low pixel count for a given object in the image. Like Omnipose, Omnistyle was developed using PyTorch [58], which allows us to take advantage of relevant code within the Omnipose ecosystem to train our image synthesis model. As a discriminator, we utilize the RibCage network (see Fig. 4.4), which was developed for microscopy images [3].

4.1 Adversarial Training

The generator and discriminator models are trained in tandem. First, the generator generates an image from the input labels (and noise). The generated image and input-label are passed to the discriminator, a binary classifier that predicts whether or not the sample is real. The objective for the generator is to produce a sufficiently realistic image to fool the discriminator into classifying the fake generated image as real. The discriminator is tasked with distinguishing real image-label pairs from those produced by the generator. The better the generator gets at creating realistic images, the harder it becomes for the discriminator to differentiate between real and fake samples. As a result, the discriminator learns to examine increasingly smaller details when making its prediction. This in turn means the generator

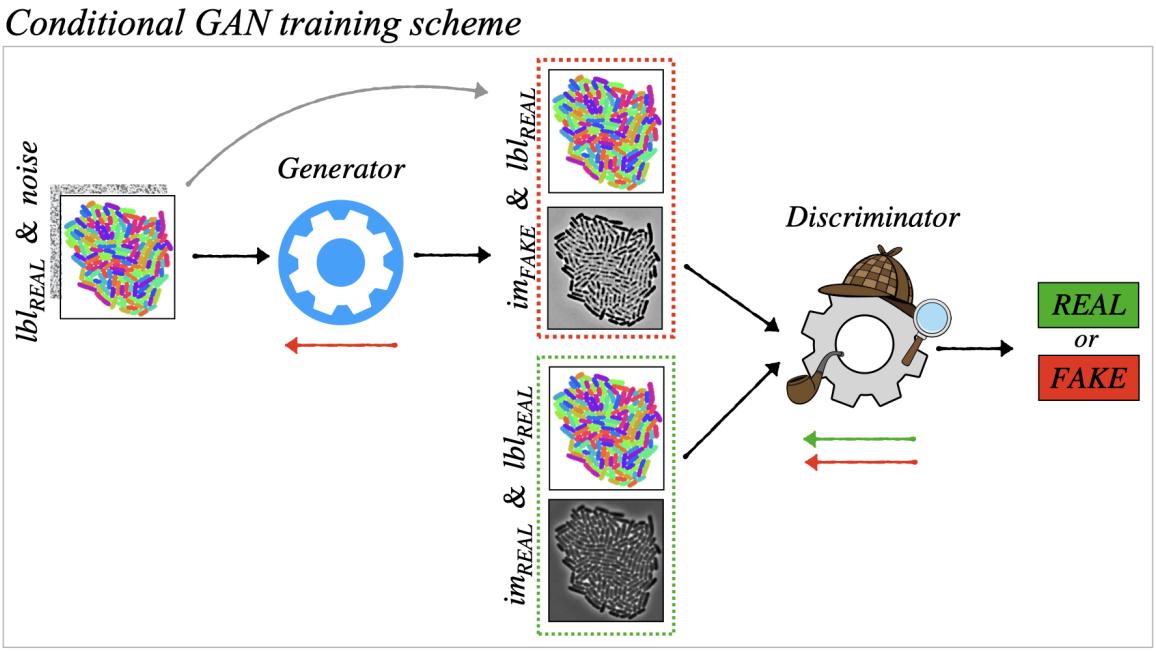


Figure 4.1: Conditional generative adversarial network (cGAN) training scheme

must find a way to replicate those higher-order features in its output images, or risk losing to the discriminator.

It is critical that both networks are trained *together* for the duration of training. If either network one is pretrained, the adversarial nature of training is undermined. If the discriminator completely dominates from the onset of training, the generator stands no chance at catching up and is unlikely to ever produce realistic images. This would be like learning physics for the first time by only ever taking graduate level exams in which the only feedback you get is whether or not you passed. There is simply too large of a knowledge gap and insufficient feedback for learning to expect any actual learning to occur. On the flip side, if the generator were already producing decently realistic images but the discriminator is completely untrained, the process of training the discriminator is likely to confuse the generator as it fumbles through early stages of training.

4.1.1 Adversarial loss

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\log[D(x)] + \mathbb{E}_{z\sim p_z(z)}\log[1 - D\big(G(z)\big)]$$

Regular binary classification uses logistic regression cost function

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\big(h_\theta(x^{(i)})\big) + (1 - y^{(i)}\log\big(1 - h_\theta(x^{(i)})\big)\right]$$

In our case, the samples of each class (real or fake) come from different sources. We can therefore write our the discriminator loss function as

$$-\frac{1}{m}\sum_{i=1}^{m}\left[\log\big(D(x^{(i)})\big) + \log\Big(1 - D\big(G(z^{(i)})\big)\Big)\right]$$

The discriminator is a binary classifier between real and synthetic samples. It must learn which features are most important in distinguishing a synthetic sample from a real one.

$$\mathcal{L}^D_{total} = \mathcal{L}^D_r eal + \mathcal{L}^D_{fake}$$

The generator loss, $\mathcal{L}^G_{total}$ has two terms, supervised and adversarial loss. The supervised term, $\mathcal{L}^G_{supervised}$ compares the generated output to the corresponding real image. The adversarial term, $\mathcal{L}^G_{adversarial}$, how well it fooled the discriminator

$$\begin{aligned}\mathcal{L}^G_{total} &= \mathcal{L}^G_{supervised} + \mathcal{L}^G_{adversarial} \\ &= \mathcal{L}^G_{supervised} - \mathcal{L}^D_{fake}\end{aligned}$$

4.1.2 Hyperparameters

Hyperparameter tuning was a significant challenge in the development of this framework. It is not possible to isolate the effects of each hyperparameter so tuning always involves a degree of trial and error. There was noise already inherent to generative adversarial network, which makes many of the model-selection criteria discussed in the previous chapters obsolete. There

	Generator	**Discriminator**
Architecture	Unet	RibCage
Loss function	$\mathcal{L}^G_{supervised} + \mathcal{L}^G_{adversarial}$	$\mathcal{L}^D_{real} + \mathcal{L}^D_{fake}$
Optimizer	`torch.optim.RAdam`	`torch.optim.SGD`

Table 4.1: Network Hyperparameters

are also two networks, which means more hyperparameters to work with for each network individually.

In our experimentation with hyperparameter tuning, we discovered some general trends and guidelines to optimize these model's relationship. The generator and discriminator need to have different optimizers. This seemed to reduce the chance of a sort of stale-mate problem that we observed. The discriminator needs to have a much smaller learning rate. By slowing down the learning for the discriminator, we prevent it from dominating the adversarial relationship. We found that the batch size needs to be small, otherwise mode collapse becomes very problematic. The generator needs to weigh the direct loss more than the adversarial loss because the adversarial loss is just so noisy that it totally overrides the direct loss.

Adding some stochasticity to the direct generator loss also helped reduce the averaging problem that we struggled with. The internal variation found between individual cells was hard to capture, so we often wound up with uniformly darkish gray cells, rather than lighter gray cells with some darker spots. The average internal pixel value was similar, but the distribution in the synthetic images was more sharply peaked around the average. We opted to modify the direct loss only slightly by implementing a quantile loss function. The quantile loss function is intended to help the model learn the conditional probability, rather than conditional mean of each pixel [54].

Batch size	4
Epochs	400
Image dimensions	224 x 224
Noise Distribution	`torch.randn`

Table 4.2: Additional Hyperparameters

4.2 Model Architecture

During training, we have two networks - a generator and a discriminator. The generator's job is to generate images consistent with given input labels. The discriminator's job is to learn to distinguish between real and generated (fake) samples during training. During inference, only the generator is used.

4.2.1 Generator

The generator network is an instance of the `CPnet` class as defined by the original Cellpose paper [66], which is just a slightly modified U-net. Importantly, it is identical to the architecture used in Omnipose, except that the input and output channel count are switched such that the network takes in the Omnipose latent representation of the masks (flows and distance fields) and outputs the synthetic image.

During training, we used a fixed image size. This is necessary because the discriminator flattens the image.

4.2.2 Discriminator

The discriminator is a binary image classification model. It is tasked with classifying *paired* samples as real or fake. The general format for image based discriminators is a two part network - a feature extractor (typically some kind of convolution) followed by a fully-connected

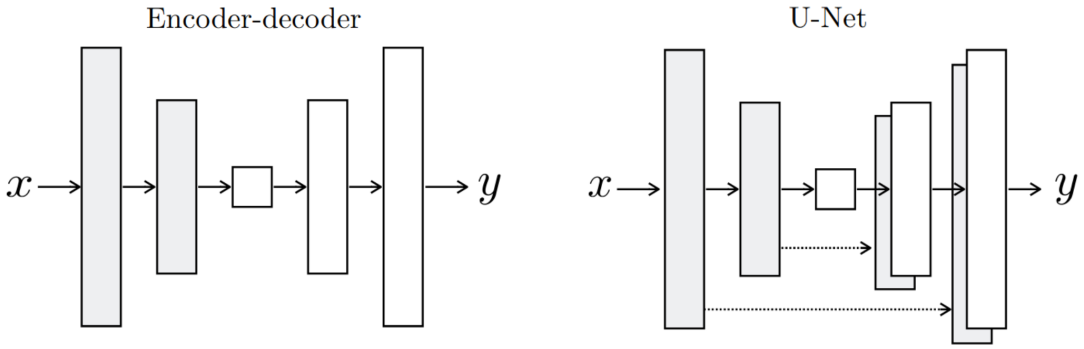


Figure 4.2: **Encoder + Decoder = U-net.** [61] The U-net model is effectively an encoder-decoder model with residual connections. The first half of a U-net is an encoder, which encodes high-dimensional images into a lower-dimensional feature space. The goal is to learn higher-order features with each layer in the encoder to capture the most important information in the image. The decoder is tasked with transforming the encoded feature back into the high-dimensional image space. The residual connections in the U-net serve to mitigate vanishing gradients and enable more explicit learning of important lower-order features.

classifier. The binary classification task is implemented with one-hot-encoded representations.

$$\textbf{Discriminator Labels:} \quad \text{real} \rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \& \quad \text{fake} \rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\textbf{Discriminator Predictions:} \quad \text{output} = \begin{bmatrix} p_{real} \\ p_{fake} \end{bmatrix} \quad \text{where} \quad p_{real} + p_{fake} = 1$$

In this case, the function being overfit is the decision boundary between the two classes. The discriminator is very prone to overfitting, especially for smaller datasets. Even with

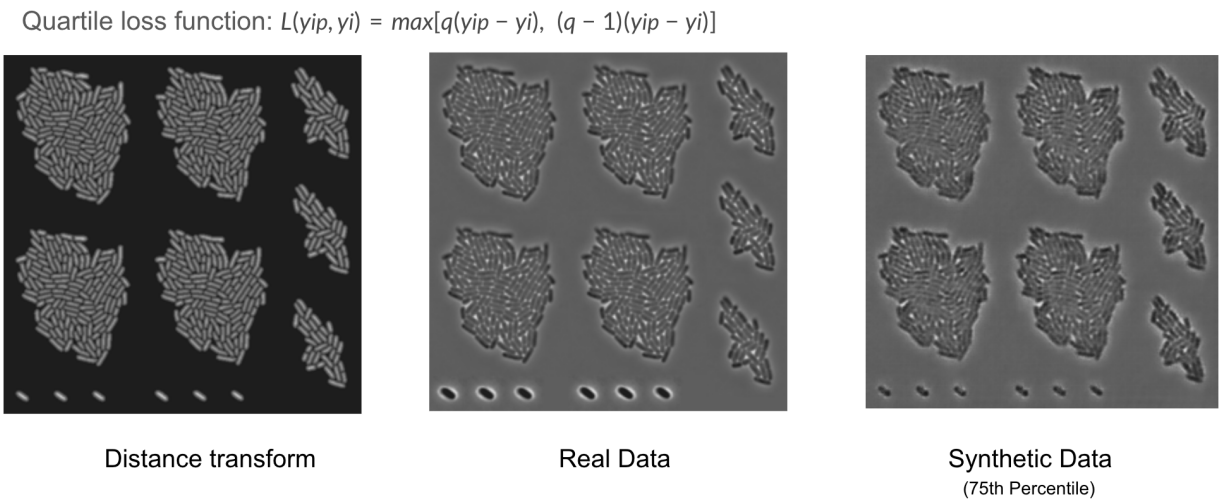
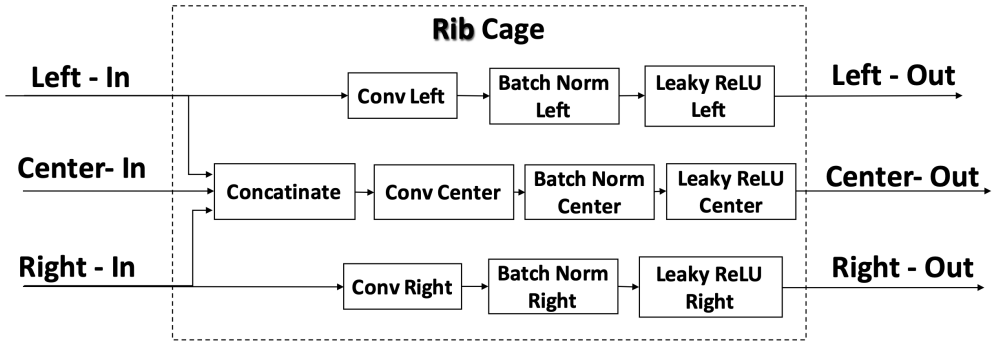


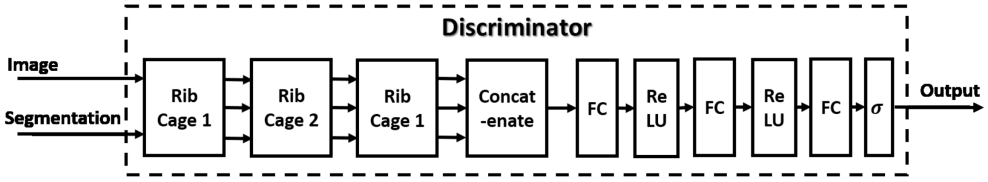
Figure 4.3: **Quantile Loss for Image Generation.** This figure shows a generated image in which the model was trained using quantile loss (in this case 75^{th} percentile) as the direct generator loss. We observed that using quantile loss improves the appearance of random internal structure in generated images. This is important because these are often the cause of over-segmentation and therefore need to be reflected in the synthetic dataset

traditional augmentations, the discriminator appeared to be memorizing real samples, which in turn makes the task for the generator even harder (if not impossible).

This was a significant challenge in our work. The discriminator would quickly overfit and proceed to dramatically outperformed the generator. The generator output was rejected so often that it was unable to optimize for the task successfully. In these situations, the generator would end up producing entirely gray images without any visible cells what so ever. Unsurprisingly, the smaller our test set, the more likely we were to encounter this problem. We quickly found that our hyperparameter tuning had to be repeated for different datasets. This was true even when training on a subset of the original data.



(a) RibCage Discriminator Architecture



(b) Single block from RibCage discriminator

Figure 4.4: **Discriminator Architecture** [3]. RibCage discriminator was specifically developed for the context of adversarial loss in microscopy image segmentation. We converted this architecture from TensorFlow to PyTorch to be compatible with our existing framework.

4.3 cGAN training algorithm

It is important to note that we train the synthesis model entirely using real data. We therefore still require images with ground-truth annotations. This is critical in our context because our goal is to generate training samples with consistent ground-truth labels.

First the masks need to be converted to a representation that is compatible with the deep-learning model. The neural network needs to be given a representation of cells that still distinguishes between individuals, but the masks are not suitable because the numerical values used to identify pixel regions are entirely arbitrary. This is the same situation as for segmentation, so we can take advantage of the solution we used in that context as well. In

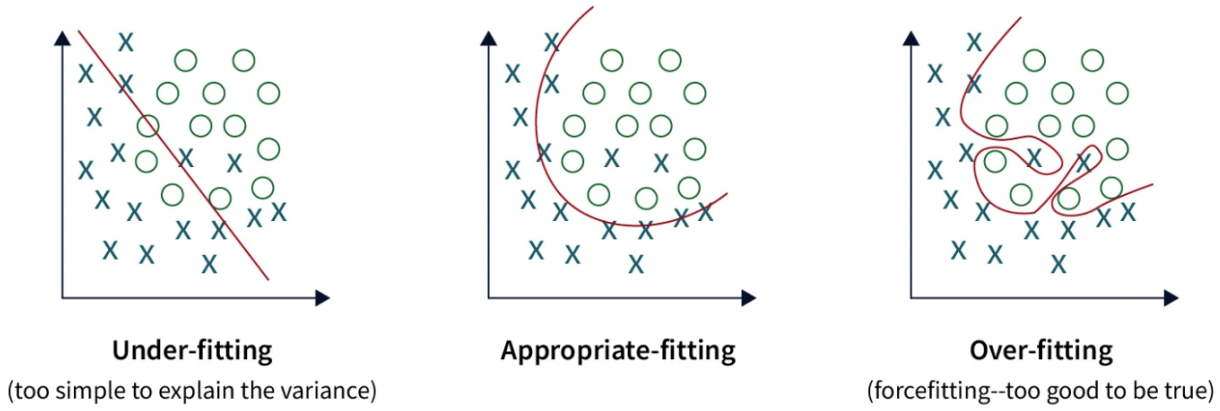


Figure 4.5: **Binary classification decision boundary fitting** [49] If the discriminator is under-fit, the generator will have an easier time fooling the discriminator with sub-optimal images. If the discriminator is over-fit, it will have effectively memorized the real training set and reject images from the generator regardless of their quality, making the generator's job impossible.

principle, the distance field representation alone would be sufficient to distinguish between cells, but we found that using the full latent training label representation from Omnipose was more successful. To generate these representation we can utilize the Omnipose data transformation pipeline. This has the added benefit that the data augmentations in Omnipose are specifically tuned for bacterial microscopy images. Once we have the appropriate latent representation for our data we can proceed to model training.

The latent representation of cell regions is now the input for our generator model. We also add a channel of noise to the input, which is an important aspect of creating generative models because it allows us to train a "one-to-many" model. The presence of noise is critical for a model to be generative, and adding noise to the input is just one way to achieve this requirement. In our case, new/distinct synthetic images can be generated using the same set of masks. Models can sometimes struggle to generate images that represent the full diversity of the dataset and instead produce the same/very similar outputs for all noise samples.

This is a mode of failure called mode-collapse, which is discussed in more detail in the next chapter.

The noise and latent cell region representation is passed into the generator, which produces a synthetic image. Early in training, we can expect these images to be fairly unconvincing. The generated image is evaluated in two ways - a supervised component which is a direct comparison to see how similar it is to the original real image, and an adversarial component which determines whether the image is sufficiently convincing overall given the masks used to generate it.

The supervised component works just like the regular supervised training described in the context of segmentation. Error is calculated on a pixel-by-pixel basis and averaged over the image to quantify how close the generated image is to the target. This is helpful during training because it speeds up model convergence, but can be problematic if weighted too heavily. Recall that our aim is to generate new images, so we don't want the generator to learn to exactly replicate images. This is particularly important for small datasets which are prone to model overfitting. Instead, we want our generator to capture the important overall features, but allow room for the generator to deviate from training images to mimic the randomness found in real imaging datasets. To do so, we add some amount of noise into this evaluation step as well in the form of quantile loss.

The adversarial component requires the second network in our model - the discriminator. The discriminator is responsible for learning to distinguish between real and fake samples.

4.4 Data

For our experiment we use the Omnipose dataset which can be found here. There is a mix of species, as well as mutations within single species in this dataset. We train models on different subsets of data. We replace the images that were left out with synthetic versions instead. This way we can isolate and measure the effect of using synthetic images compared to real ones. One of the advantages of this approach is that it allows us to make direct comparisons between model performance. To do so, we also keep all other training variables

dataset	short images	short cell count	long images	long cell count
train set	94	19322	118	3907
test set	38	14521	74	3596

Table 4.3: Breakdown of the original full Omnipose dataset by morphology.

the same between approaches.

The shift in distribution is a population shift, specifically morphological variation. For testing distribution shift performance, we can train on all short cells and then generate synthetic images from long masks.

We can also vary the size of datasets without varying the relative diversity of dataset. That is, we can train using a representative subset of the training data, and see how the added synthetic samples compare.

4.4.1 Training label latent representation

As was the case for segmentation, single-cell masks are transformed into a latent representation during training. The most simple representation would be semantic masks, but that is not sufficient because we want to train a model that is consistent with single-cell regionsIn the case of instance labels, using pixel value to distinguish between cells is ultimately incompatible with our deep learning framework.

Transforming these connected components into a more suitable latent representation is the primary focus of both Cellpose and Omnipose [20, 66, 55]. After experimentation, we found that the same latent representation used for segmentation training in Omnipose also worked well for this application, which is detailed in Figure 4.8.

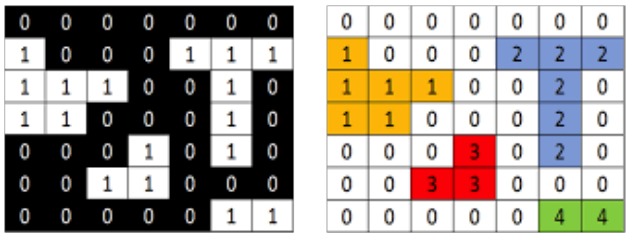


Figure 4.6: **Connected component labels** [43]. Mask regions are originally represented with connected components. On the left is an example of semantic labels, where all connected components are labeled with a pixel value of 1, and all background pixels are zero. On the right is an example of instance labels, in which each distinct cluster (in our case cells) has a unique identifying label value.

4.4.2 Integration with OmniPose

Omnistyle takes advantage of the image pre-processing pipeline developed for Omnipose [20]. The custom image-augmentations performed during training are specifically engineered to ensure masks and images are consistent and viable under the chosen augmentation.

One of the major contributions of Omnipose is the data processing pipeline for model training. The small size of bacteria can become problematic during data augmentation. Some augmentations can distort images enough such that the corresponding ground-truth labels are no longer compatible with the augmented image. Our initial attempts with pix2pix were limited by the small number of augmentations that we could use. We wanted to be able to use the existing Omnipose infrastructure for image augmentations.

For example, when a cell is only a few pixels wide at a particular point, resizing or rotating the image can cause a discontinuity in the labeled region, and thereby creating poor ground-truth training information. We also utilize the same latent representation of cell masks as the one developed in Omnipose, and have found that this produces significantly better results than semantic or distance-field representations.

Incorporating these aspects of Omnipose also allows us to integrate our image synthesis directly into the Omnipose segmentation algorithm, thereby enabling the option to create synthetic images and immediately use them to train a segmentation model without additional steps required from the user. The model can be trained from the main script in Omnipose. This makes it easy to use the synthetic data generator for segmentation training.

4.4.3 Code specifics

The Omnistyle model is defined as a class and is currently implemented using the CellposeModel as a parent module. This means we are able to inherit all the relevant methods used in the Omnipose training loop, which was written for the CellposeModel class. Omnistyle has several new methods, but overrides only one inherited method - the training step. All training configurations from the original model work as before, but where the model would otherwise perform a single forward-backward pass and update weights with each train step, we instead perform the forward and backward passes for both the generator and discriminator, and update weights for both networks.

There are some training hyperparameters that must be further specified because they do not exist in the regular CellposeModel. This includes all hyperparameters for the discriminator (architecture, loss function, optimizer, learning rate) or other adversarial related variables such as noise distribution. Default values were set based on manual hyperparameter tuning (this is a challenging task for reasons discussed later), but can be over-ridden through keyword arguments in the command line.

This has the advantage of allowing us to use identical model architectures for segmentation and image generation, which is reasonable because we can consider each process as the inverse of the other. One important note is that the labels that are used to compute loss in Omnipose are not entirely identical to those actually generated by the model. The generator produces only three channels comprised of two flow channels and one channel for the distance field. The extra terms returned by the data loader are used to compute higher-order loss terms for segmentation model training, but are not used during the synthesis process.

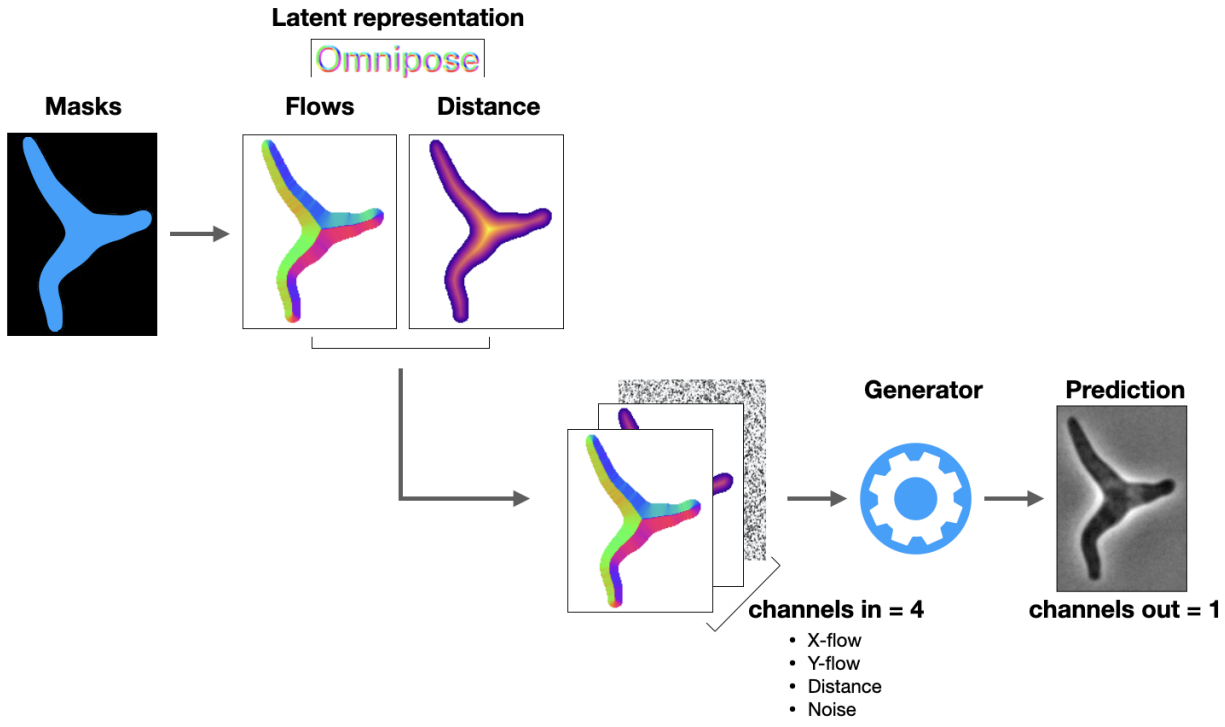


Figure 4.7: **From Mask to Synthetic Image.** Input masks are converted to same latent representation used in Omnipose [20]. The three channels are comprised of the flow-field (1 channel each for x and y components) and the distance transform. An additional channel is added for the necessary noise channel before the latent representation is passed to the generator, which returns a predicted synthetic image. This process is performed for training and inference. Note that during training, this image is passed to the discriminator for adversarial learning (see Fig. 4.1).

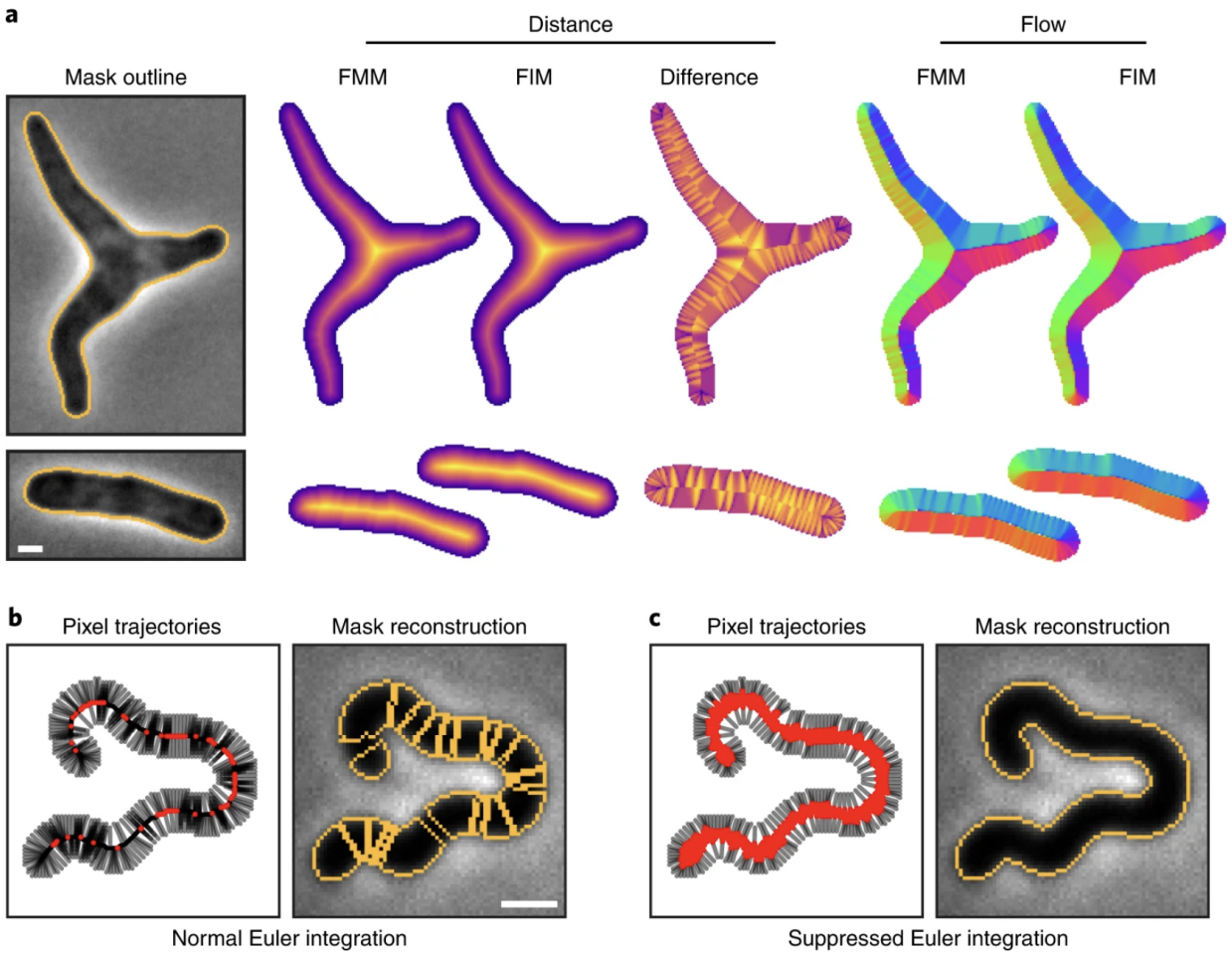


Figure 4.8: **Omnipose latent mask representation.** Omnipose uses latent mask representation to enable segmentation of arbitrary morphologies. Single-cell mask regions are converted to the latent representation during training and used as training labels to optimize model parameters. Note that masks are converted to latent representations *after* image augmentations occur that change mask shape in some way. Figure from original Omnipose paper [20].

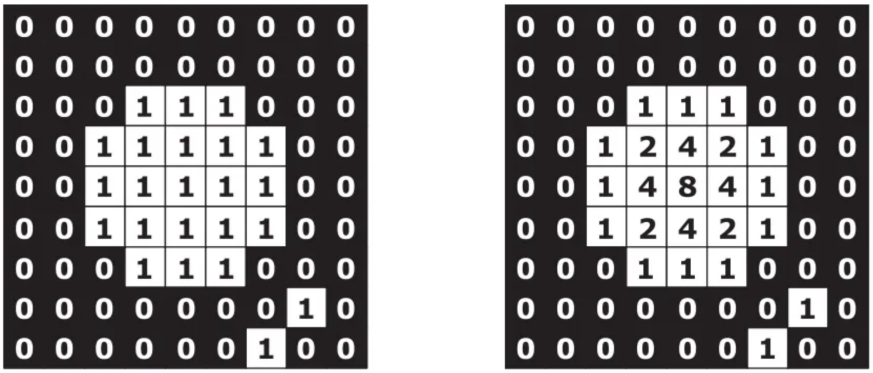


Figure 4.9: **Distance Transform** [68]. In the binary image (left) all pixels inside the object are labeled with a 1 and all background pixels are labeled 0. There is no way to represent two objects in contact with this binary representation. The distance transform (right) of the binary image replaces categorical values with distance measurements. Each pixel value is replaced with the distance (in pixels) to the nearest boundary of that object (in cardinal directions for this example). This allows us to distinguish between objects by identifying all boundary pixels with value = 1.

Chapter 5

OUTLOOK AND CONCLUSION

Our results from the previous chapter serve as a proof of concept that synthetic images can be used to improve segmentation performance on out-of-distribution morphologies. This is in many ways a surprising result which merits further investigation from an information theory and/or machine learning research perspective. But, remember that the premise of our work is to make segmentation tools more functional for researchers by actually reducing the amount of training data we need to train segmentation models.

5.1 Ongoing work: developing a production-ready framework

Ultimately, our goal is to publish a tool that can be customized without requiring specialized machine learning knowledge or extensive programming expertise. Achieving this, however, is challenging for many of the reasons discussed throughout this thesis—including the dependence on suitable training data and the need for substantial hyperparameter tuning when adapting to new datasets. In this section, we examine several of these challenges in greater detail. Specifically, we explore practical obstacles in model selection, common failure modes of generative adversarial networks (GANs), and our preliminary efforts toward identifying effective solutions. Addressing these challenges is a key step toward bridging the gap between state-of-the-art machine learning techniques and practical, user-friendly research tools.

The work presented in this section is ongoing, but we have included some preliminary findings in our effort to monitor and stabilize adversarial training. We discuss our suspicions for sources of instability and our approach towards automating model selection by introducing a form memory in the generator model.

5.1.1 Monitoring training progress

Adversarial model training is inherently noisier than standard supervised machine learning approaches. Supervised machine learning works by minimizing the error between ground-truth targets and model predictions. Tracking training progress is therefore relatively straightforward - model convergence can be directly observed from the error over time. In adversarial learning, however, the generator and discriminator are trained by simultaneously minimizing their own error while *actively* maximizing that of the other model. This means that improvement in either model directly results in a larger measured error in the other; if the discriminator is able to very successfully distinguish between real and generated samples, the generator's adversarial loss term will increase, thereby increasing the overall error associated with the generator's performance, regardless of the quality of generated images. Thus, it is possible for a a high generator error to be recorded despite producing decent images, thereby making the loss a poor indicator for model performance.

One of the key challenges we encountered was that hyperparameters required substantial tuning across datasets of different sizes. This poses a significant barrier to usability, as we aim to create a model that researchers can apply to new datasets without needing extensive hyperparameter adjustment.

5.1.2 Mode collapse

Generative adversarial models are susceptible to a form of failure called mode collapse [37], in which generator predictions become very limited or even identical, rather than effectively capturing the full diversity present in the dataset. Mode collapse refers specifically to the variety of generated data and does not necessarily reflect the quality of individual generated samples. In some cases the generator may produce a small number of very good/plausible outputs that are all very similar, or follow similar patterns. This becomes problematic if/when the discriminator is able to take advantage of that pattern as a criteria for distinguishing fake samples from real ones. In this scenario, the generator is often stuck in a

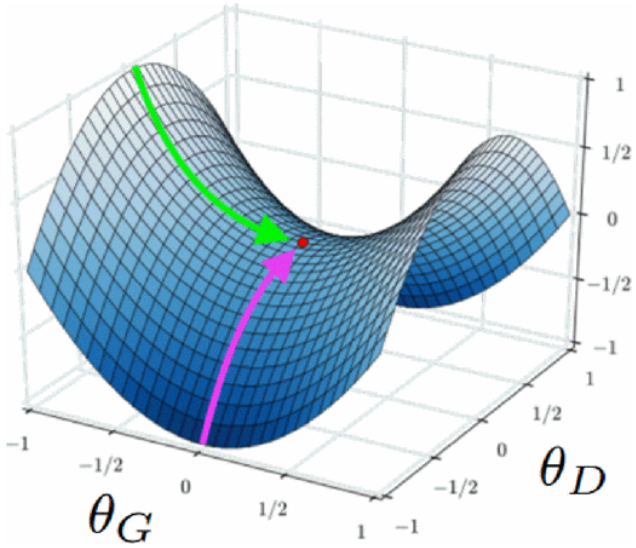


Figure 5.1: **Adversarial Parameter Optimization** [37]. In adversarial learning, we are effectively trying to optimize two sets of parameters; the generator weights (θ_G) and the discriminator weights (θ_D). Each network is updated to *minimize* loss on it's respective task, which in turn increases the loss of the other network. The optimal solution for this situation can be thought of as the identification of a saddle point in the loss landscape.

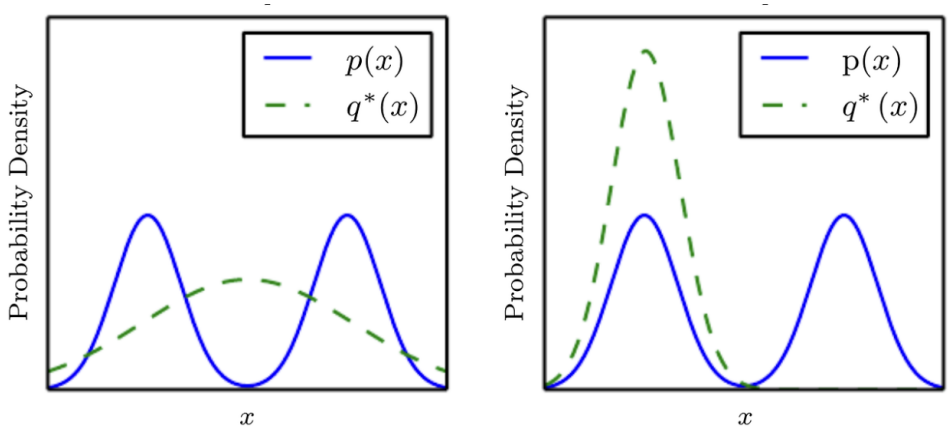


Figure 5.2: **Two manifestations of mode collapse** [32]. In some cases, the generated images capture only a subset of the distribution and produce outputs limited to that subset. In others cases, the generator may land on an average over the entire distribution.

local minimum, and the information that is back-propagated via the adversarial loss may be insufficient to dislodge the generator from that minimum.

We found that smaller datasets, in particular, were highly susceptible to mode collapse (see Fig. 5.3). In these cases, the discriminator consistently outperformed the generator, preventing the generator from converging to meaningful outputs. The discriminator loss dominated the training signal, rendering the generator loss nearly negligible. As a result, the generator frequently produced output images that were entirely gray or lacked meaningful structure.

To mitigate this imbalance and give the generator a better chance of learning, we reduced the weight of the adversarial loss in the generator's total loss function. This adjustment helped stabilize training and allowed the generator to begin producing more realistic outputs.

Ultimately, we faced a trade-off between model stability and mode collapse. When the model did converge, it often did so into a collapsed mode; when it did not converge, model behavior was highly unstable, making it difficult to systematically select checkpoints due to

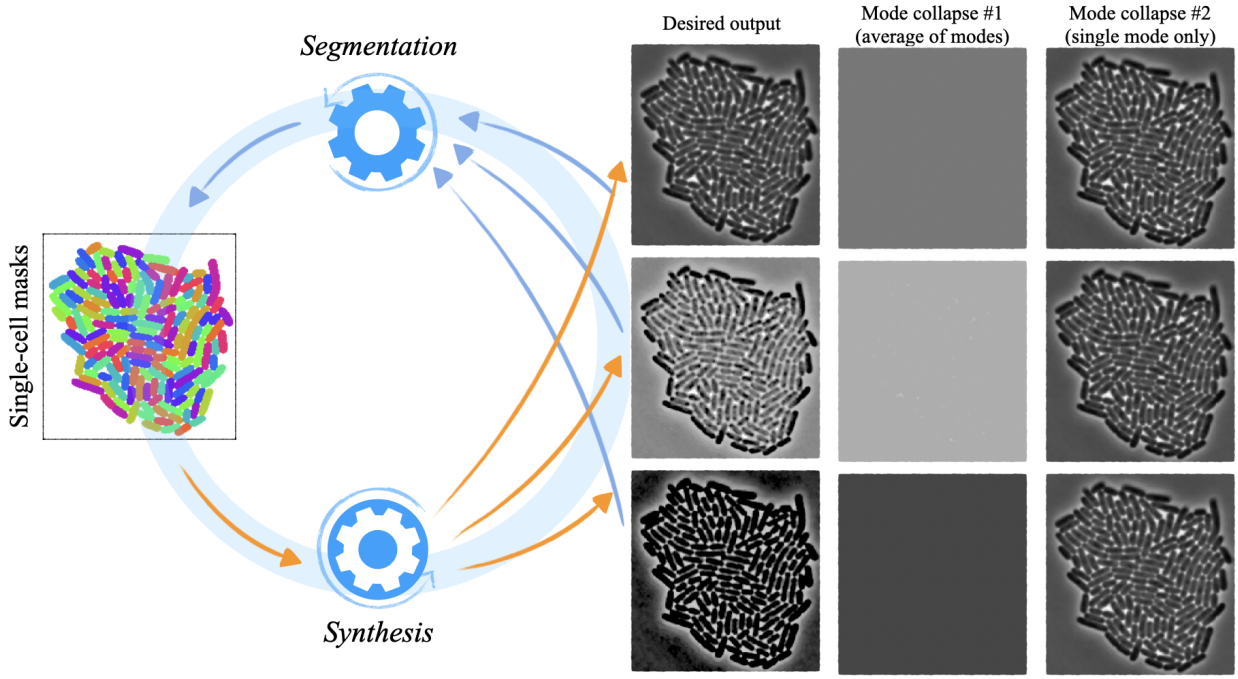


Figure 5.3: **Mode Collapse in Bacterial Deepfakes.** Image synthesis is a one-to-many process. That is, a single input can correspond to multiple allowable outputs, or *modes*. When successful, our model should therefore be able to generate a variety of plausible images that span the possible modes for a given set of input masks (left column). When mode collapse occurs in this context, it manifests in one of two ways. In the first case (middle column), the generator collapses onto an average of the modes and produces flat gray images. In the second case (right column), the generator collapses onto a single mode, producing identical images each time it receives the same mask, despite new noise sampling. The former is significantly more problematic in our context because the generated images are entirely unusable. Unfortunately, however, this was also the primary mode of failure we encountered.

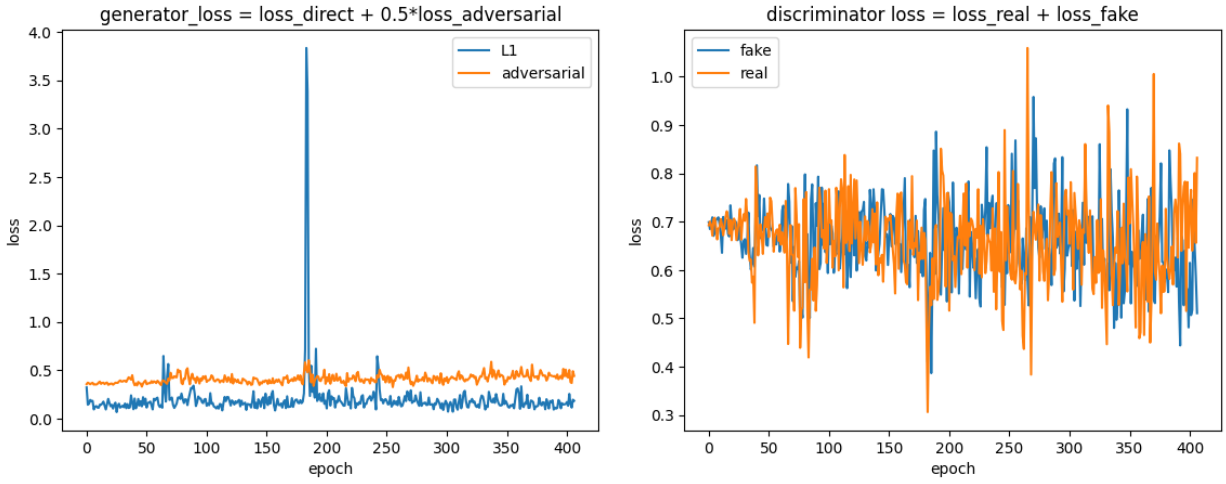
the noisy and inconsistent outputs.

5.1.3 Model selection

The most effective strategy we employed to address the issue of mode collapse was introducing memory into the generator via exponential moving average (EMA) weights. Specifically, we logged the exponential moving average of the generator's weights throughout training, which can then be applied to generator output during inference [73]. While our preliminary results at this stage are purely qualitative, this approach appears to have stabilized training dynamics and smoothed the progression of generated validation images across checkpoints, as illustrated in Figures 5.10a and 5.10b. This improved the visual consistency and convergence of our generator output, but it also introduced additional hyperparameters. These include variables such as update intervals and decay rates, which add to the seemingly endless list of hyperparameters that must be optimized empirically through trial and error. Nevertheless, we found EMA to be a valuable tool in reducing variance and mitigating volatility in generator training.

To better understand the challenge of model selection, we ran a full experiment that involved training both a generative model and a series of segmentation models. We first trained the generator and saved checkpoints at different stages throughout training. For each checkpoint, we then trained a segmentation model using the images produced by the generator at that point. Since our main goal is to achieve strong segmentation performance, we wanted to explore how the quality of generated images affected that performance. The results of this experiment are shown in Figure 5.8.

Specifically, we were interested in two questions: at what point does the quality of the generated images become good enough to improve segmentation, and is there a stage where further training might actually make segmentation worse? One of our early, qualitative observations was that segmentation results did not vary as much across different checkpoints as we initially expected. This suggests that segmentation models may be somewhat robust to small differences in the quality of generated images. This is valuable information, as it



(a) Generator loss: $\mathcal{L}^G_{total} = \mathcal{L}^G_{direct} - 0.5\mathcal{L}^D_{fake}$ (b) Discriminator loss: $\mathcal{L}^D_{total} = \mathcal{L}^D_{fake} + \mathcal{L}^D_{real}$

Figure 5.4: **Loss Curves for cGAN training.**

supports the idea that the introduction of higher-order features is the primary factor in the improvement of segmentation of out-of-distribution samples. However, more careful analysis and further experiments would be necessary to identify which features are most important to accurately replicate in synthetic images for the best segmentation performance

We noticed that the loss curves for both the generator and discriminator were very noisy. As discussed, this is to be expected in adversarial training, but also makes model selection difficult even for stable configurations because the images produced by the generator tend to fluctuate. By saving the generator model at intermittent checkpoints during training, we can observe these fluctuations directly.

5.2 Next steps

Like Omnipose, OmniStyle is intended to be an open-source project that researchers can *actually* use to support their image segmentation workflows. Our immediate objective is to finalize the conversion of the OmniStyle codebase into a production-ready module that is

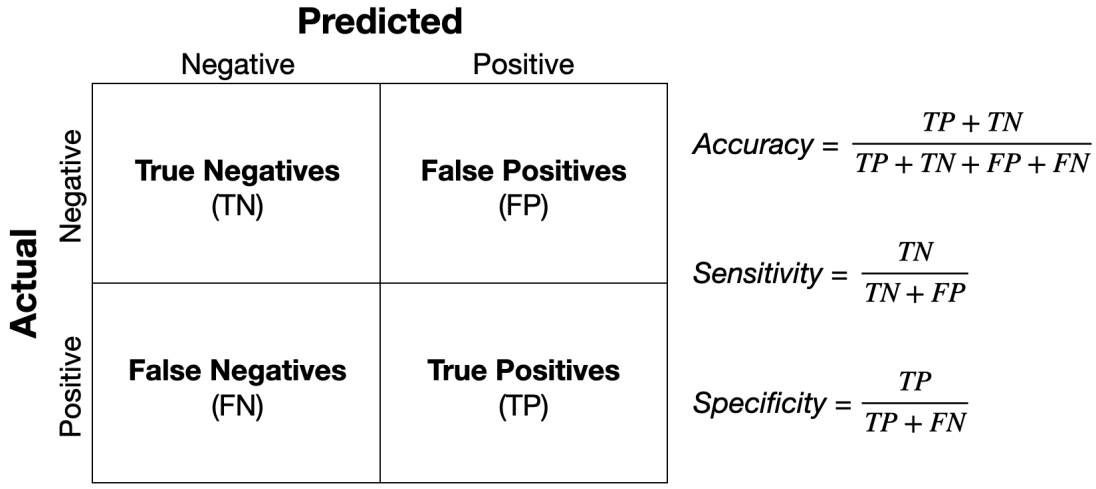


Figure 5.5: **Binary classification confusion matrix.** Confusion matrices are commonly used to examine a variety of metrics for classification tasks, such as accuracy, sensitivity, and specificity [69]. *Accuracy* measures how often the model is right overall. It tells you the percentage of all predictions—both positive and negative—that were correct and is best used when the number of positive and negative cases is roughly equal. *Sensitivity,* otherwise known as recall or true-positive rate, measures how good the model is at detecting positive cases. It tells you the percentage of actual positive cases that the model correctly identified, and is useful in situations where missing a positive case is especially harmful. *Specificity* measures how good the model is at detecting negative cases. It tells you the percentage of actual negative cases that the model correctly identified and is helpful if catching false positives is important for the task at hand.

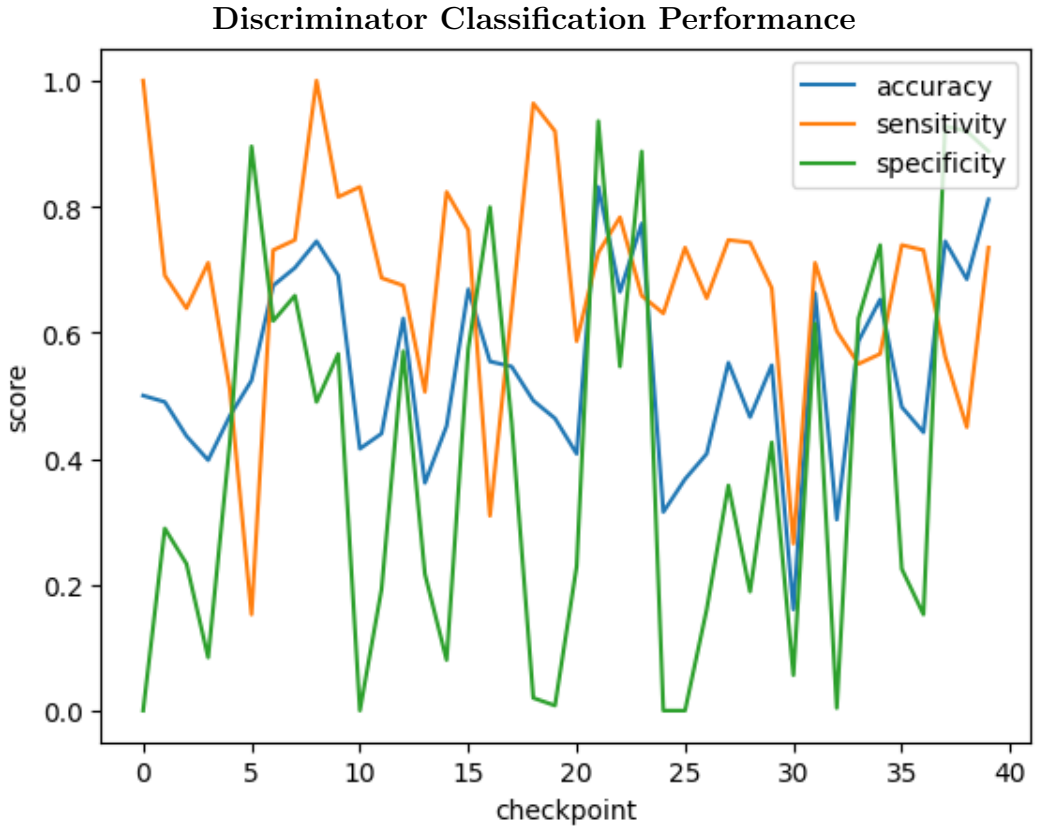
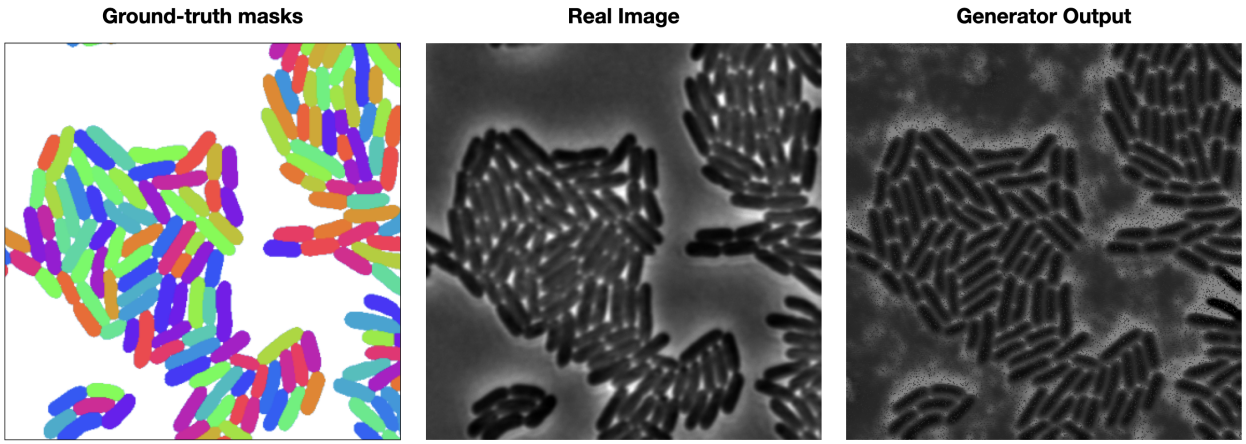
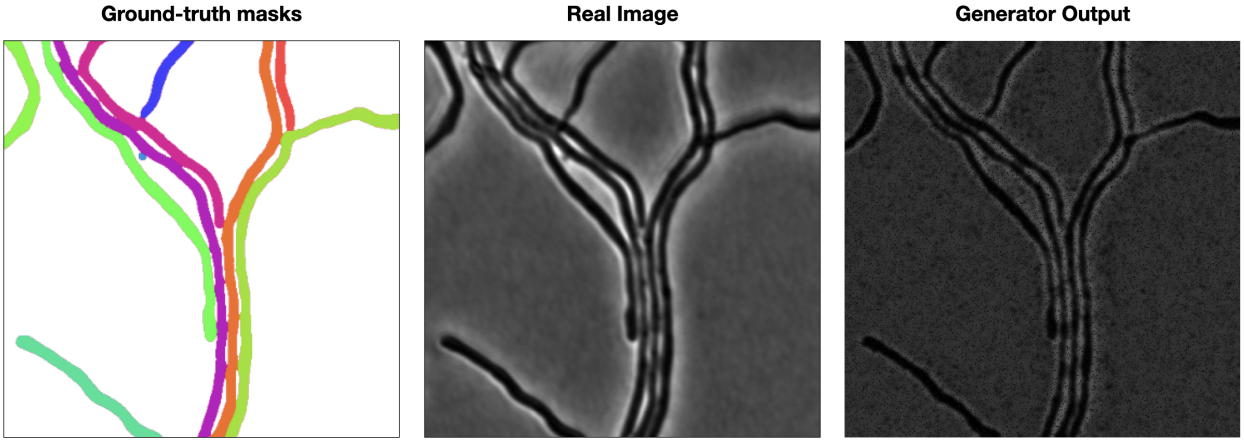


Figure 5.6: **Discriminator classification performance during training.** This plot shows the values of standard classification performance metrics (accuracy, sensitivity, and specificity) of the discriminator model over the course of training. There are 10 training epochs between subsequent checkpoints. Overall, the model has a slightly higher sensitivity than specificity which indicates that the model is generally more likely to recognize real images as real than it is to recognize synthetic images as fake. This is to be expected, as the real images stay the same throughout training whereas the synthetic images vary greatly. Checkpoints with high sensitivity and but low specificity are a good starting point for model selection because the generator was able to fool the discriminator *while* the discriminator was still successfully recognizing real images.



(a) In-distribution sample generated image



(b) In-distribution sample generated image

both modular and readable. This step is essential for improving the usability of our existing framework and lays the groundwork for future development. This section discusses our ideas for what that future development might include next.

5.2.1 Space-time segmentation

Space-time segmentation of time-lapse data is a natural next step for our work on segmentation, and was an early motivation for this synthetic image generation project. Deep-learning

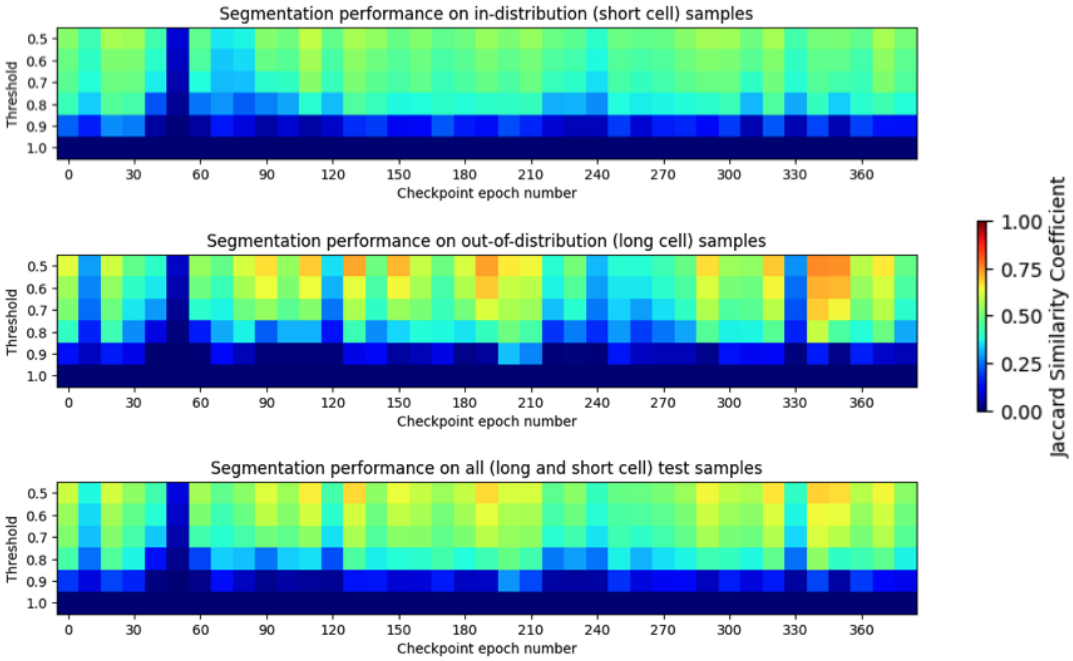


Figure 5.8: **Segmentation performance across generator checkpoints.**Segmentation performance (IoU curves) for models trained using synthetic data from different model checkpoints. A generator model saved every ten epochs during training, then used to make a synthetic out-of-distribution (long) dataset. For each checkpoint, the corresponding synthetic samples are added to the original in-distribution (short) samples to make a mixed (real and synthetic) dataset. This dataset is then used to train a segmentation model (from scratch) using Omnipose. For each segmentation model, we evaluate its performance using Omnipose test dataset, which contains both in-distribution and out-of-distribution samples. We consider performance on out-of-distribution and in-distribution samples separately and together. For every panel, column panel shows the segmentation performance corresponding to each generator checkpoint. To isolate the effects of the synthetic samples, all models were trained with identical hyperparameters and for the same number of epochs. This is important to note because models at any given checkpoint may have the capacity for higher performance with more thorough hyperparameter optimizations.

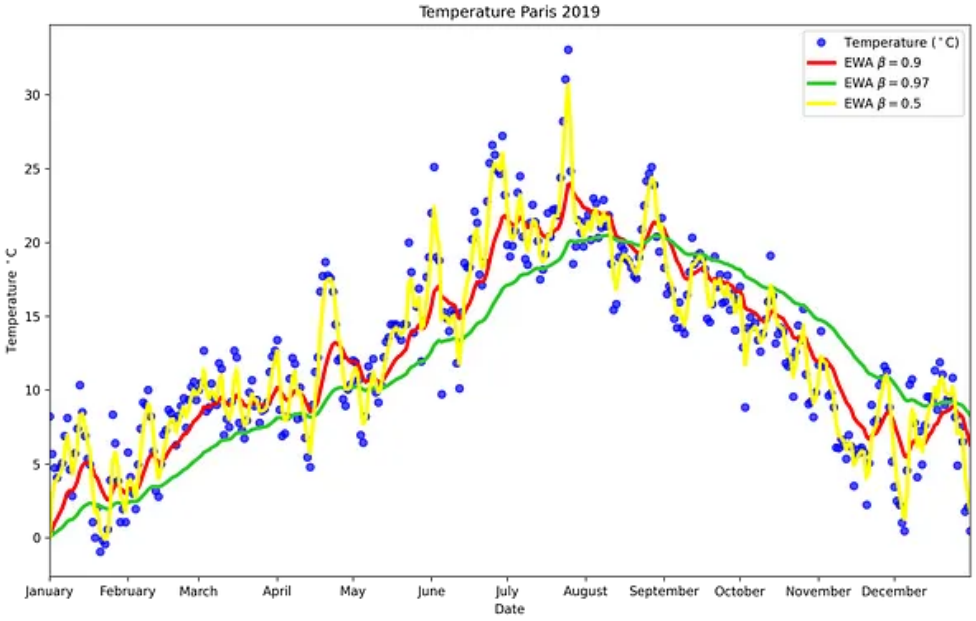
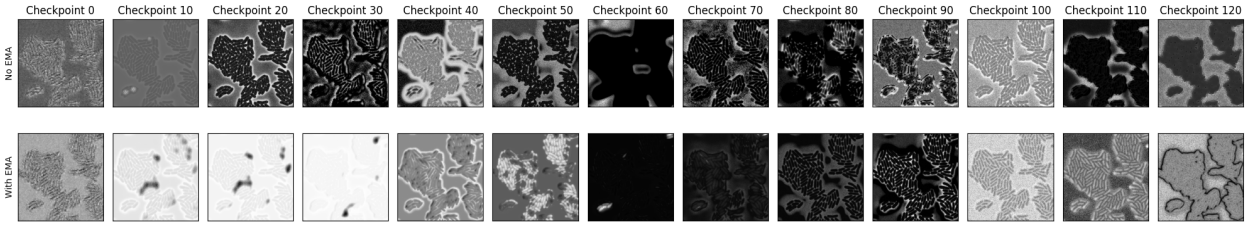
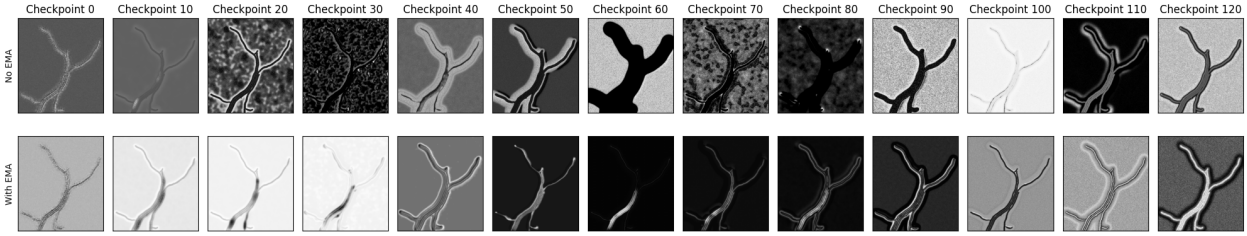


Figure 5.9: **Exponential Moving Averaging Applied to Noisy Data** [13]. In this example, the exponential moving average is used to smooth noisy temperature data (°C) from Paris over the course of one year. The blue dots represent individual data points, and the three curves are the smoothed output of exponential moving average for three weighting parameters $\beta = 0.9, 0.97, 0.5$. The weighting parameter, chosen on the interval (0, 1), effectively determines how sensitive the moving average is to new data points; if β is small (close to 0), the average is very sensitive to new data points, which is reflected in the yellow curve which is the noisiest of the three because it has the lowest β value. In contrast, a β closer to 1 yields a smoother curve; The green curve corresponds to $\beta = 0.97$ which results in a much smoother curve than that of a $\beta = 0.9$, shown in red. This is because the average is less sensitive to new data points because the weighting of older data points is still significant.



(a) In-Distribution Test Sample



(b) Out-of-Distribution Test Sample

Figure 5.10: **Exponential moving average smooths generator training.** This figure shows illustrates preliminary results of applying an exponential moving average to generator weights during adversarial training for (a) an in-distribution sample and (b) an out-of-distribution sample. For each sample, the top row shows the regular generator output at the indicated training checkpoint *without* averaging. The bottom panel shows the generator output *with* the corresponding exponential moving average applied to generator weights for that same checkpoint. Our method for tracking and applying the EMA to our model is based on [73].

based models can process sequential data and take advantage of temporal features in addition to the spatial features used for single-frame segmentation. In addition to enabling cell tracking, video or time-lapse based segmentation is improve the overall segmentation quality of each frame. Training data availability is even more limited in these cases. A single time-lapse is like a flip-book of 100 single frames, and we need thousands of flip-books for training. Training samples must also contain a time-component to link cells between frames, in addition to the standard pixel-level hand-annotation in each frame. Synthetic data could drastically reduce this burden and enable efficient fine-tuning of video-based segmentation models.

One of the most persistent challenges in the single-cell analysis of microscopy images is cell tracking between frames in time-lapse data. This is important for many kinds of analyses, such as lineage tracking, but this task is challenging for several reasons. Varying frame rates and growing cell populations are just some of the reasons for this challenge. Accurate automatic cell-tracking is also a challenging computer vision task and has become a focus for dedicated research and coordinated benchmarking such as by the Cell Tracking Challenge [47]. Segmentation is often an important step of a cell-tracking algorithm, as segmented masks offer simplified representation of the cells than the original microscopy images. Accurate and temporally consistent segmentation is therefore important for any segmentation-based cell-tracking algorithm.

Temporal consistency in frame-level segmentation, however, is also challenging. Two particularly challenging yet common situations are cell division and cells coming into contact with one another. As cells divide, there are often multiple frames in which the septum is slowly pinching the two new daughter cells apart, and the point at which the two daughter cells actually separate into two distinct cells can be hard to determine, even for an expert hand-annotator. The exact frame of septation is therefore somewhat arbitrary, but we know that after septation the two daughter cells remain separate from one another. What often happens in these situations when segmenting frame-by-frame, is that consecutive frames will result in inconsistent predictions on whether there are one or two cells present in the image.

One frame may predict that the two cells have divided into two daughter cells, but the next frame may predict a single mother cell, thereby creating temporal inconsistency. Similarly, when two cells move over time and come into contact with each other, subsequent frames my predict that the two individual cells are a single cell, which is of course impossible. Introducing a mechanism to enforce temporal consistency is therefore a useful way to deal with these kinds of mistakes.

To introduce temporal consistency, we require a model that can process video (or time-lapse) data. Deep learning approaches for sequential data exist, for example, recurrent neural networks [65]. These models are designed to handle sequential data by processing the entire sequence together and thus being able to learn temporal features of the data, in addition to the spatial features of the individual frames within the sequence [52]. There are a range of specific models that handle the sequential information in their own specific ways. The most basic recurrent neural networks essentially just store an extra hidden layer that stores information from preceding frames. This hidden layer effectively works as a latent variable for storing information that is important to know from previous frames to make a good prediction on the following frames. In our case, that would mean enforcing that a split has occurred and not allowing the cells to re-fuse together after septation, for example. These rules are not directly enforced, rather the model learns from the data over time, and since there are no examples of that happening in our datasets, it should learn to enforce that pattern over time. This approach has already been implemented for other segmentation frameworks [4] with promising results, which we can use as a basis for integrating the same or similar framework within Omnipose.

5.2.2 Foundation models and self-supervised learning

Generative machine-learning has exploded in recent years, perhaps most notably with the release of ChatGPT [11] (which stands for **G**enerative **P**retrained **T**ransformer). ChatGPT and other large-language-models (LLMs) such as Anthropic's Claude [2] and Meta's Llama [70] are all examples of generative language models that have become part of daily life in just

a matter of years. Their ability to process queries in plain language and subsequently produce intelligible responses has reshaped the way many people think about artificial intelligence. Furthermore, these models, while originally designed for natural language processing, have set the stage for generative machine-learning approaches in other applications. For instance, Google's Gemini [25] has recently been expanded to handle additional data modalities like images and audio files.

These models are all examples of *Foundation models*, and are huge in size and complexity, with average tunable parameter counts in the *billions*. They also require correspondingly enormous amounts of data to train from scratch and are therefore impractical for low-computation resource applications. Foundation models get their name because they are intended to serve as foundations for many other downstream tasks, including further machine-learning. Large-language models are examples of foundation models for language, but computer vision foundation models are already showing promise, particularly in our quest for *usable* machine-learning based tools for two main reasons. The first is that they can be pretrained in settings with extensive computational and data resources and be distilled into smaller models for downstream users. Secondly, while foundation models are vastly larger than those used in this work, and require correspondingly larger training datasets - they do *not* require labeled data for training. This is because they use a method called self-supervised machine-learning, wherein we allow the model to self-optimize the low-dimensional latent representation or encoding of the image [53] [21]. The idea behind this is that we no longer impose constraints on how images are represented, but rather we allow the model to learn which features are most relevant in recognizing an image in a lower-dimensional feature space.

Self-supervised machine-learning is especially exciting from a training data limitation perspective. Self-supervised machine learning methods, such as contrastive learning [14], allows for feature extraction *without* ground-truth labels. The model effectively learns a low-dimensional representation that encodes the most important features of an image. Computer vision foundation models, such as Segment Anything Model (SA Model) by Meta AI [35]

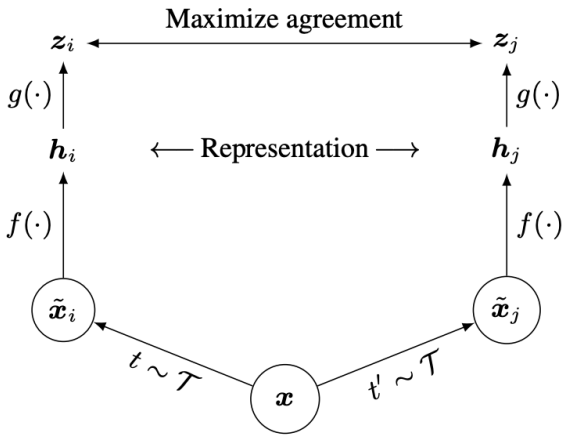


Figure 5.11: **Contrastive learning** [14]. This schematic illustrates the basic concept of contrastive learning.

are a promising step towards reducing the burden of data annotation, and are expected to dramatically shape the next generation of biomedical machine learning applications [38].

The ongoing improvement of image-based foundation-models is a promising step towards leveraging the power of such large-scale models in imaging contexts. However, we must note that foundation models are typically trained on natural images and therefore still require some degree of finetuning for successful application to biological or medical images [45], though this still requires fewer images and computational resources than training models with equivalent performance with regular supervised machine learning [34]. More recently, SAM has been finetuned on microscopy images and now acts as a microscopy image foundation model for segmentation [5].

5.3 Conclusion

Algorithms designed to quantify imaging data must demonstrate consistently high performance before any downstream analysis can occur. To reach this level of performance, many models are trained to perform narrowly defined tasks on specific types of images. Any devia-

tion in task or image type—such as those introduced by new experimental conditions—often necessitates retraining or fine-tuning, which typically requires additional training data and specialized machine learning expertise. This presents a significant barrier in research settings, where image characteristics can vary widely and unpredictably.

As machine learning and artificial intelligence continue to evolve, it's critical to recognize the practical limitations of even the most advanced models. Bridging the gap between "ML breakthroughs" and "breakthroughs that use ML" requires thoughtful implementation, which is far from trivial.

In particular, models must be developed with domain-specific objectives and constraints in mind—many of which demand deep subject matter expertise to identify, let alone address. Research environments are especially challenging in this regard, as key factors affecting model development and performance can shift with changes in experimental design.

Ultimately, the reality is that successful ML applications often require significant time, resources, and interdisciplinary collaboration. Understanding and addressing these challenges is essential for making ML tools truly useful and reliable in scientific research.

BIBLIOGRAPHY

[1] G. Aad, B. Abbott, K. Abeling, N. J. Abicht, S. H. Abidi, A. Aboulhorma, H. Abramowicz, H. Abreu, Y. Abulaiti, A. C. Abusleme Hoffman, B. S. Acharya, C. Adam Bourdarios, L. Adamczyk, and et.al Adamek, L. Search for new phenomena in two-body invariant mass distributions using unsupervised machine learning for anomaly detection at $\sqrt{s} = 13$ TeV with the atlas detector. *Phys. Rev. Lett.*, 132:081801, Feb 2024.

[2] Anthropic. The claude 3 model family: Opus, sonnet, haiku.

[3] Assaf Arbelle and Tammy Riklin Raviv. Microscopy cell segmentation via adversarial neural networks. *CoRR*, abs/1709.05860, 2017.

[4] Assaf Arbelle and Tammy Riklin Raviv. Microscopy cell segmentation via convolutional lstm networks, 2019.

[5] Anwai Archit, Luca Freckmann, Sushmita Nair, Nabeel Khalid, Paul Hilt, Vikas Rajashekar, Marei Freitag, Carolin Teuber, Genevieve Buckley, Sebastian von Haaren, Sagnik Gupta, Andreas Dengel, Sheraz Ahmed, and Constantin Pape. Segment anything for microscopy. *Nature Methods*, 22(3):579–591, 2025.

[6] Nisha Arya. How to avoid overfitting. https://www.kdnuggets.com/2022/08/avoid-overfitting.html. Accessed: 2025-05-03.

[7] Anish Athalye, Curtis Northcutt, and Jonas Mueller. Class imbalance, outliers, and distribution shift. https://dcai.csail.mit.edu/2024/imbalance-outliers-shift/. Accessed: 2025-04-04.

[8] Jeannie Bailey, Julie Cass, Joe Gasper, Ngoc-Diep Ngo, Paul Wiggins, and Colin Manoil. Essential gene deletions producing gigantic bacteria. *PLOS Genetics*, 15(6):e1008195, 2019. https://journals.plos.org/plosgenetics/article/file?id=10.1371/journal.pgen.1008195, type=printable.

[9] Roel Bertens. How to style transfer your own images. https://xebia.com/blog/how-to-style-transfer-your-own-images/. Accessed: 2025-06-08.

[10] H. D. Block. The perceptron: A model for brain functioning. i. *Rev. Mod. Phys.*, 34:123–135, Jan 1962.

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[12] Antoine Buetti-Dinh, Vanni Galli, Sören Bellenberg, Olga Ilie, Malte Herold, Stephan Christel, Mariia Boretska, Igor V. Pivkin, Paul Wilmes, Wolfgang Sand, Mario Vera, and Mark Dopson. Deep neural networks outperform human expert's capacity in characterizing bioleaching bacterial biofilm composition. *Biotechnology Reports*, 22:e00321, 2019.

[13] Tobías Chavarría. Exponentially weighted average. https://medium.com/@tobias-chc/exponentially-weighted-average-5eed00181a09. Accessed: 2025-06-05.

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 02 2020.

[15] SY Chin, J Dong, K Hasikin, R Ngui, KW Lai, PSQ Yeoh, and X Wu. Bacterial image analysis using multi-task deep learning approaches for clinical microscopy. *PeerJ Computer Science*, 10:e2180, 2024.

[16] H. James Choi, Teresa W. Lo, Kevin J. Cutler, Dean Huang, W. Ryan Will, and Paul A. Wiggins. Protein overabundance is driven by growth robustness. *bioRxiv*, 2024.

[17] Lorenzo Ciampiconi, Adam Elwood, Marco Leonardi, Ashraf Mohamed, and Alessandro Rozza. A survey and taxonomy of loss functions in machine learning, 2024.

[18] Exxact Corportation. How to create a dataset for machine learning. https://www.exxactcorp.com/blog/Deep-Learning/how-to-create-a-dataset-for-machine-learning. Accessed: 2025-04-04.

[19] Kevin J. Cutler. *Utilizing modern machine learning approaches for image cytometry*. PhD thesis, University of Washington, 2023.

[20] Kevin J. Cutler, Carsen Stringer, Teresa W. Lo, Luca Rappez, Nicholas Stroustrup, S. Brook Peterson, Paul A. Wiggins, and Joseph D. Mougous. Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation. *Nature Methods*, 19(11):1438–1448, 2022.

[21] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers, 2024.

[22] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc., 2021.

[23] Dennis Eschweiler, Rüveyda Yilmaz, Matisse Baumann, Ina Laube, Rijo Roy, Abin Jose, Daniel Brückner, and Johannes Stegmaier. Denoising diffusion probabilistic models for generation of realistic fully-annotated microscopy image data sets, 2023.

[24] B Fasel. An introduction to bio-inspired artificial neural network architectures. *Acta neurologica Belgica*, 103(1):6—12, March 2003.

[25] Google Gemini Team. Gemini: A family of highly capable multimodal models, 2024.

[26] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[27] Jeckel H and Drescher K. Advances and opportunities in image analysis of bacterial cells and communities. *FEMS Microbiology Reviews*, 2021.

[28] Georgeos Hardo, Ruizhe Li, and Somenath Bakshi. Quantitative microbiology with widefield microscopy: navigating optical artefacts for accurate interpretations. *npj Imaging*, 2(1):26, 2024.

[29] Georgeos Hardo, Maximilian Noka, and Somenath Bakshi. Synthetic micrographs of bacteria (symbac) allows accurate segmentation of bacterial cells using deep neural networks. *BMC Biology*, 20(1):263, 2022.

[30] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. pages 2980–2988, 10 2017.

[31] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[32] Hyunsu Jeong. Gan pt2: Theoretical results. https://89douner.tistory.com/331. Accessed: 2025-04-05.

[33] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[34] Mingu Kang, Heon Song, Seonwook Park, Donggeun Yoo, and Sérgio Pereira. Benchmarking self-supervised learning on diverse pathology datasets. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3344–3354, 2023.

[35] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023.

[36] Giona Kleinberg, Sophia Wang, Ester Comellas, James R. Monaghan, and Sandra J. Shefelbine. Usability of deep learning pipelines for 3d nuclei identification with stardist and cellpose. *Cells & Development*, 172:203806, 2022.

[37] Youssef Kossale, Mohammed Airaj, and Aziz Darouichi. Mode collapse in generative adversarial networks: An overview. In *2022 8th International Conference on Optimization and Applications (ICOA)*, pages 1–6, 2022.

[38] Rayan Krishnan, Pranav Rajpurkar, and Eric J. Topol. Self-supervised learning in medicine and healthcare. *Nature Biomedical Engineering*, 6(12):1346–1352, 2022.

[39] Zeki Kuş and Musa Aydin. Medsegbench: A comprehensive benchmark for medical image segmentation in diverse data modalities. *Scientific Data*, 11(1):1283, 2024.

[40] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[41] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[42] Yann Lecun, Patrick Haffner, and Y. Bengio. Object recognition with gradient-based learning. *Shape, Contour and Grouping in Computer Vision*, 08 2000.

[43] Kun Lin. Connected component labeling using mpi. https://cse.buffalo.edu/faculty/miller/Courses/CSE633/Kun-Lin-Spring-2020.pdf. Accessed: 2025-06-04.

[44] Boyuan Ma, Xiaoyan Wei, Chuni Liu, Xiaojuan Ban, Haiyou Huang, Hao Wang, Weihua Xue, Stephen Wu, Mingfei Gao, Qing Shen, Michele Mukeshimana, Adnan Omer Abuassba, Haokai Shen, and Yanjing Su. Data augmentation in microscopic images for material data mining. *npj Computational Materials*, 6(1):125, 2020.

[45] Jun Ma, Yuting He, Feifei Li, Lin Han, Chenyu You, and Bo Wang. Segment anything in medical images. *Nature Communications*, 15(1):654, 2024.

[46] Christos Manettas, Nikolaos Nikolakis, and Kosmas Alexopoulos. Synthetic datasets for deep learning in computer-vision assisted tasks in manufacturing. *Procedia CIRP*, 103:237–242, 2021. 9th CIRP Global Web Conference – Sustainable, resilient, and agile manufacturing and service operations : Lessons from COVID-19.

[47] Martin Maška, Vladimír Ulman, Pablo Delgado-Rodriguez, Estibaliz Gómez-de Mariscal, Tereza Nečasová, Fidel A. Guerrero Peña, Tsang Ing Ren, Elliot M. Meyerowitz, Tim Scherr, Katharina Löffler, Ralf Mikut, Tianqi Guo, Yin Wang, Jan P. Allebach, Rina Bao, Noor M. Al-Shakarji, Gani Rahmon, Imad Eddine Toubal, Kannappan Palaniappan, Filip Lux, Petr Matula, Ko Sugawara, Klas E. G. Magnusson, Layton Aho, Andrew R. Cohen, Assaf Arbelle, Tal Ben-Haim, Tammy Riklin Raviv, Fabian Isensee, Paul F. Jäger, Klaus H. Maier-Hein, Yanming Zhu, Cristina Ederra, Ainhoa Urbiola, Erik Meijering, Alexandre Cunha, Arrate Muñoz-Barrutia, Michal Kozubek, and Carlos Ortiz-de Solórzano. The cell tracking challenge: 10 years of objective benchmarking. *Nature Methods*, 20(7):1010–1020, 2023.

[48] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey, 2020.

[49] Subhaditya Mukherjee. Improving model accuracy in image classification. https://www.scaler.com/topics/deep-learning/improving-mode-accuracy-in-image-classification/. Accessed: 2025-04-02.

[50] Nvidia. Convolutional neural network. https://www.nvidia.com/en-in/glossary/convolutional-neural-network/. Accessed: 2025-06-10.

[51] Kyoung-Su Oh and Keechul Jung. Gpu implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314, 2004.

[52] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks, 2019.

[53] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

[54] Dvir Ben Or, Michael Kolomenkin, and Gil Shabat. Generalized quantile loss for deep neural networks, 2020.

[55] Marius Pachitariu, Michael Rariden, and Carsen Stringer. Cellpose-sam: superhuman generalization for cellular segmentation. *bioRxiv*, 2025.

[56] Rohan Paleja and Matthew Gombolay. Neural networks and backpropagation, Jan 2021.

[57] Swapnesh Panigrahi, Dorothée Murat, Antoine Le Gall, Eugénie Martineau, Kelly Goldlust, Jean-Bernard Fiche, Sara Rombouts, Marcelo Nöllmann, Leon Espinosa, and Tâm Mignot. Misic, a general deep learning-based method for the high-throughput cell segmentation of complex bacterial communities. *eLife*, 10:e65151, sep 2021.

[58] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[59] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning, 2017.

[60] Parvin Razzaghi, Karim Abbasi, and Pegah Bayat. Learning spatial hierarchies of high-level features in deep neural network. *Journal of Visual Communication and Image Representation*, 70:102817, 2020.

[61] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[62] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[63] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[64] Nicolae Sapoval, Amirali Aghazadeh, Michael G. Nute, Dinler A. Antunes, Advait Balaji, Richard Baraniuk, C. J. Barberan, Ruth Dannenfelser, Chen Dun, Mohammadamin Edrisi, R. A. Leo Elworth, Bryce Kille, Anastasios Kyrillidis, Luay Nakhleh, Cameron R. Wolfe, Zhi Yan, Vicky Yao, and Todd J. Treangen. Current progress and open challenges for applying deep learning across the biosciences. *Nature Communications*, 13(1):1728, 2022.

[65] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai kin Wong, and Wang chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting, 2015.

[66] Carsen Stringer, Michalis Michaelos, and Marius Pachitariu. Cellpose: a generalist algorithm for cellular segmentation. *bioRxiv*, 2020.

[67] S Stylianidou, C Brennan, SB Nissen, NJ Kuwada, and PA Wiggins. Supersegger: robust image segmentation, analysis and lineage tracking of bacterial cells. *Mol Microbiol.*, 2016.

[68] Kareim Tarek. Improving model accuracy in image classification. https://medium.com/@kareimtarek1972/euclidean-distance-transform-edt-introduction-5d7d7c144aa. Accessed: 2025-04-06.

[69] Ashish Tiwari. Chapter 2 - supervised learning: From theory to applications. In Rajiv Pandey, Sunil Kumar Khatri, Neeraj kumar Singh, and Parul Verma, editors, *Artificial Intelligence and Machine Learning for EDGE Computing*, pages 23–32. Academic Press, 2022.

[70] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[71] Vinorth Varatharasan, Hyo-Sang Shin, Antonios Tsourdos, and Nick Colosimo. Improving learning effectiveness for object detection and classification in cluttered backgrounds. In *Conference: 2019 Workshop on Research, Education and Development of Unmanned Aerial Systems*, pages 78–85, 11 2019.

[72] Kislay Verma. Combining rule-systems and machine learning. https://kislayverma.com/programming/combining-rule-systems-and-machine-learning/. Accessed: 2025-05-02.

[73] Amit Yadav. Exponential moving average (ema) in pytorch. https://medium.com/@heyamit10/exponential-moving-average-ema-in-pytorch-eb8b6f1718eb. Accessed: 2025-02-15.

[74] Abolfazl Zargari, Benjamin R. Topacio, Najmeh Mashhadi, and S. Ali Shariati. Enhanced cell segmentation with limited training datasets using cycle generative adversarial networks. *iScience*, 27(5), 2025/06/13 2024.

[75] ZEMIM. Deep neural network (dnn) explained. https://medium.com/@zomev/deep-neural-network-dnn-explained-0f7311a0e869. Accessed: 2025-05-08.

[76] Wenying Zhou, Yang Yang, Cheng Yu, Juxian Liu, Xingxing Duan, Zongjie Weng, Dan Chen, Qianhong Liang, Qin Fang, Jiaojiao Zhou, Hao Ju, Zhenhua Luo, Weihao Guo, Xiaoyan Ma, Xiaoyan Xie, Ruixuan Wang, and Luyao Zhou. Ensembled deep learning model outperforms human experts in diagnosing biliary atresia from sonographic gallbladder images. *Nature Communications*, 12(1):1259, 2021.