Accelerating Biological Discovery with Deep Learning and Spatial Optical Barcodes

Thesis by
Morgan Sarah Schwartz

In Partial Fulfillment of the Requirements for the
Degree of
PhD in Biology

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2024
Defended May 6, 2024

Morgan Sarah Schwartz
ORCID: 0000-0001-8131-9125

ACKNOWLEDGEMENTS

A PhD can feel like a long, rollercoaster ride when you are stuck in the middle. There are many people who helped me through the many ups and downs to get me where I am today.

To my advisor David Van Valen, thank you for your steadfast belief in these projects and my ability to bring them to their full potential.

Thank you to my committee members, Matt Thomson, Ellen Rothenberg, Paul Sternberg, and Long Cai, for their support and guidance during my PhD. I appreciate your enthusiasm and interest in my work during our meetings. In particular, I would like to thank Ellen for serving as a mentor and a sounding board during my moments of doubt.

When I started in the Van Valen lab, I had some nascent ideas about how to write good open-source code, but the reality was that I had been flying blind up to that point. Will Graf, thank you for teaching me how to write high-quality code and for instilling good software development habits that I still use today.

The Van Valen wet lab was just getting started when I joined the lab alongside Edward Pao and Emily Laubscher. I could not have asked for better friends and team members for this crazy journey. Thank you for putting up with me when I was stressed, grouchy and convinced that it would be better to just scrap my project and start over. The successes in this thesis would not have happened without you.

John Soro started on the barcode project as an undergraduate in the midst of pandemic lockdown. Despite his initial interest in working at the bench, he jumped into a computational project and delivered a key piece of software for identifying new pattern-generating gRNAs. Thank you for the three dedicated years that you put into this project both at the computer and at the bench.

Danielle Gallandt joined the lab as a technician and quickly became my second pair of hands at the bench. Thank you for your willingness to enthusiastically adapt to whatever new protocol variation I threw at you while always delivering excellent results.

For the past few years, I have had the opportunity to spend a few weeks every summer at the MBL teaching a deep learning course for biological microscopy. It is an exhausting and exhilarating experience and I always return to the lab afterwards

with a fresh outlook on my work. Thank you to the many wonderful people who make this course such a special experience, especially Jan Funke, Florian Jug, Shalin Mehta, Larissa Heinrich, Diane Adjavon, Caroline Malin-Mayor, Will Patton, Alex Hillsley, and Benjamin Gallusser.

I was very fortunate to be surrounded by many wonderful mentors during my time as an undergraduate at Smith College. To Laura Katz and Michael Barresi, thank you for teaching me to love research and giving me the freedom and support to pursue my interests. To Rob Dorit and Stacie Hagenbaugh, you opened my eyes to opportunities that I did not expect to be within my reach. Thank you for believing in and encouraging me.

Thank you to my family members for their enthusiasm and support at every step of my academic career. Regardless of your background and understanding, I appreciate your questions and curiosity about my work. Thank you to my parents for encouraging me to explore research and continuing to support me when I diverged into the realm of biology.

Finally, to my husband and partner for both lab and life, you have found so many ways to support me during the course of this PhD: designing protocols, troubleshooting experiments, nudging me to go into lab over the weekend to do an important step of an experiment, cooking far more delicious dinners that I can manage on my own, celebrating successes, sympathizing with failures, reviewing practice presentations, editing my writing, and overall being my rock. Thank you for getting me to the finish line.

ABSTRACT

Methodological advances in biology have given us a powerful suite of tools for measuring the state of the cell. Among these methods, next-generation sequencing, including single-cell methods, enables comprehensive measurement of gene expression; however, sequencing-based methods often preclude the collection of other visible phenotypic information. In contrast, light microscopy supports many different measurements that can be acquired in sequential rounds of labeling and imaging because light microscopy does not destroy the sample. Furthermore, light microscopy supports live cell imaging, including the use of fluorescent reporters to observe signaling dynamics in real time. In order to fully understand cellular function, multimodal data collection is needed that encompasses live cell response, end-point phenotypes, and finally perturbations to test the components of relevant signaling networks. In this thesis, I present key advances to create a unified experimental platform for interrogating the cell state. This platform uses light microscopy to collect multimodal measurements of cell state while supporting high-throughput perturbation screening. This platform is supported by a suite of deep learning analysis tools to enable quantitative analysis of these high-dimensional datasets. In Chapter 2, I introduce Caliban, our deep learning method for nuclear segmentation and tracking. In Chapter 3, I present a new method of optical barcodes to enable microscopy-based pooled perturbation screens. Finally, in Chapter 4, I describe preliminary work that leverages the previously described cell tracking and barcoding methodologies to explore the interdependencies of signaling pathway dynamics.

PUBLISHED CONTENT AND CONTRIBUTIONS

1. Schwartz, M. S. *et al.* Caliban: Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning. *bioRxiv* (2023).
   M.S. participated in the conception of the project, development of the HITL labeling methodology and associated data versioning methodology, development of the nuclear segmentation and cell tracking methodology, development of the benchmarking framework and writing the manuscript.

PUBLISHED CONTENT NOT INCLUDED IN THESIS

1. Greenbaum, S. *et al.* A spatially resolved timeline of the human maternal–fetal interface. *Nature* **619,** 595–605 (2023).
   M.S. contributed to the development and writing of software for cell segmentation and clustering.

2. Israel, U. *et al.* A Foundation Model for Cell Segmentation. *bioRxiv* (2023).
   M.S. contributed to data engineering for the project.

3. Schwartz, M. *et al.* Scaling biological discovery at the interface of deep learning and cellular imaging. *Nature Methods* **20,** 956–957 (2023).
   M.S., along with other authors, conceived the research directions described in the manuscript.

4. Greenwald, N. F. *et al.* Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning. *Nature Biotechnology* **40,** 555–565 (2022).
   M.S. contributed to the development Mesmer's deep learning architecture and the development of metrics software.

5. Bannon, D. *et al.* DeepCell Kiosk: scaling deep learning–enabled cellular image analysis with Kubernetes. *Nature Methods* **18,** 43–45 (2021).
   M.S. contributed to the code base and associated documentation.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

Technological advances have given the field many different methods for evaluating the state of the cell. In addition to gene expression measurements, next-generation sequencing methods can also capture protein-binding patterns and chromatin accessibility either from bulk populations of cells or, more recently, single cells. However, while these sequencing measurements capture incredibly informative datasets, they also have inherent limitations. Namely, their use currently precludes the acquisition of visible phenotypic or spatial information from sampled cells. This loss includes dynamic information about the cell's behavior and its interactions with the environment. To overcome the existing challenges posed by these separate measurements, we need to collect "multimodal" measurements that include as many axes of cell state as possible (Figure 1.1). For this purpose, light microscopy emerges as a powerful tool for capturing multimodality data at single-cell resolution. Light microscopy is generally nondestructive to acquired samples, which can enable repeated rounds of labeling and imaging, where each round captures a different type of information about the state of the cell.

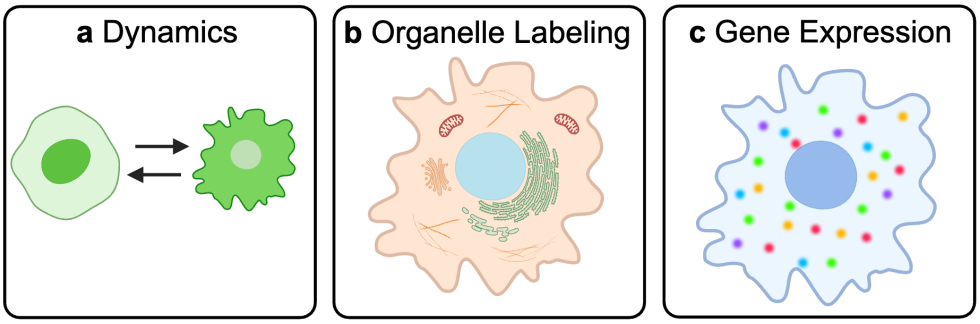


Figure 1.1: **Multi-modal measurements of single cells using light microscopy.** Light microscopy enables different types of data to be captured from the same cell using repeated rounds of staining and imaging. (a) Live cell imaging is used in the first round to capture signaling dynamics using a fluorescent reporter. (b) Organelles are then labeled in the fixed cell to capture subcellular organization. (c) Finally, gene expression is captured using spatial transcriptomics. Created with Biorender.com.

1.1 Cellular and molecular dynamics are essential dimensions underlying biological processes

In pursuit of a comprehensive understanding of the cell state, fluorescent microscopy serves as a powerful tool for collecting single-cell datasets that capture multiple phenotypic dimensions. The development of fluorescent reporter constructs has facilitated our ability to monitor the activity of signaling pathways of individual cells in real time.[1–3] This advancement is in stark contrast to previous approaches that relied on destructive measurements of whole populations at a single point in time. Although these population measurements are still useful, they cannot resolve the biologically significant differences that exist between otherwise seemingly identical individual cells. Single cell investigations of signaling pathways have revealed that the dynamic response of a pathway can serve to encode information about the type of input stimuli received and trigger different downstream responses (Figure 1.2).[4] Although it has become clear that these dynamic processes are multifaceted and governed by a variety of internal and external inputs, all of which contribute to the cell's state and fate, detecting and measuring these processes remains a critical undertaking in the field of biology.

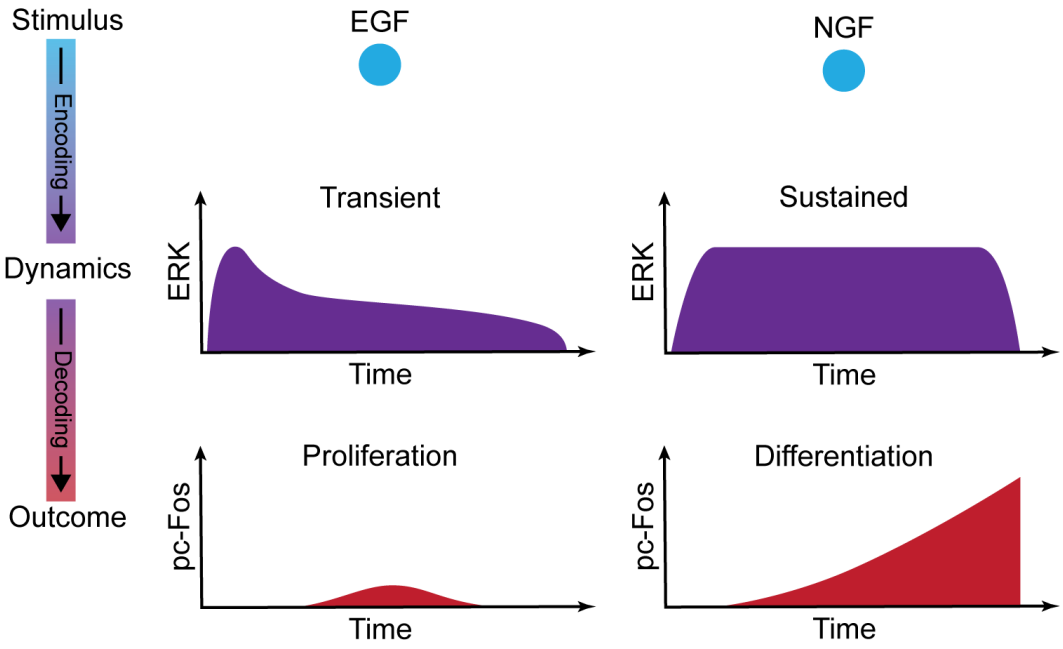


Figure 1.2: **Signaling dynamics lead to different downstream responses.** The shape of ERK signaling dynamics serves to encode the type of input received. EGF leads to transient ERK activation while NGF triggers sustained activation. Downstream pc-Fos decodes ERK dynamics to produce different outcomes: proliferation or differentiation. Adapted from Purvis & Lahav.[4]

The construction of a platform capable of supporting multimodal measurements, including signaling dynamics and downstream responses, poses several challenges. First, the platform needs to be able to capture live cell measurements, as snapshots in time of individual cells are not sufficient to determine the full course of a dynamic response. Second, the platform must be able to collect endpoint measurements, such as gene expression, to determine which genes are expressed in an individual cell following a specific signaling pattern. Third, the platform should have the ability to incorporate perturbations, to facilitate deconvolution of how specific genes and pathways contribute to specific dynamics. Although there are a variety of tools that currently exist to enable the collection of data addressing the first two points, the development of a platform that incorporates these ideas at scale has proven challenging. The sheer number of potential candidate genes that could be manipulated in an investigation of a signaling pathway requires a reevaluation of how to perform forward genetic screening.

1.2 The power of pooled screens for biological discovery

Forward genetics are a powerful tool for biological discovery when designed with high-throughput capability. These experiments target a particular known gene for perturbation in order to observe the resulting phenotype, which can vary from a change in cell shape to a change in how that cell interprets its environment. A variety of perturbation methods have been developed that enable these experiments to be executed with increasing ease in mammalian systems, with the most recent notable addition being the CRISPR-Cas9 system. Although there are many platforms that enable targeted forward genetic manipulations, including RNAi, ZFN, TALEN, and CRISPR-Cas9, CRISPR-Cas9 in particular has shown greater ease of use and efficacy in gene knockout and knockdown in mammalian systems than its predecessors.[5,6] Cas9, a DNA nuclease, is capable of inducing dsDNA breaks in the host genome and targeting specific locations in the host genome by an RNA sequence, called guide RNA (gRNA), which has a 5' region that can be changed to direct Cas9 targeting specific regions of the host genome.[7,8] As Cas9 targeting is directed solely through the gRNA sequence and short RNA synthesis is relatively trivial compared to manipulations at the plasmid level needed for TALENs and ZFNs, these features have made CRISPR-Cas9 a valuable tool for forward genetic screens. In addition, multiple derivatives of the Cas9 protein and related effector proteins have been developed that allow greater specificity of use. This includes a wider range of gRNAs to extend the available population of potential regions in the

genome capable of being targeted for double-stranded DNA break, and Cas9 mutants, such as dCas9, which is capable of binding DNA, but lacks nuclease activity and therefore remains persistently bound to DNA instead.[9] However, to fully leverage the power of the CRISPR-Cas9 system for forward genetics, perturbation screens must be performed at high throughput to support both a large number of targets and to achieve a sufficient sample size to enable statistical detection of phenotypes.

Large-scale forward genetics experiments, which are predominantly performed in cell culture, can be performed in one of two formats: arrayed or pooled. In an arrayed screen, each well receives a different experimental perturbation and is thus analyzed separately (Figure 1.3a). As a result, arrayed-format screens are expensive in terms of both time and reagent cost, and often require laboratory automation to reach a significant scale, but provide greater ease in linking perturbation to phenotype. In contrast, pooled screens combine all perturbations into a single reagent pool, but control the delivery so that each cell receives only one perturbation. As such, pooled screens are highly effective in screening large numbers of perturbations while limiting the experimental cost. However, they introduce a challenge associated with discovering which perturbations produced the desired phenotype.

The first pooled genome-wide screens utilizing CRISPR-Cas9 relied on enrichment-based methods to discover positive hits in the screen (Figure 1.3b). In this regime, an enriched population of cells is created using positive selection for fitness, negative selection with a drug or FACS.[9,11–13] The enriched population of cells is then sequenced to reveal which gRNAs are present in the population. Although this approach has proven to be effective, it requires the question to be reduced to a single phenotype of interest that can be selected with a single screening step. This decision excludes unexpected and potentially rare phenotypes from the analysis and limits the scope of potential discovery. However, clever applications of photoconvertable proteins have extended this approach to the microscope, so live imaging and phenotyping can be performed in real time while cells displaying the phenotype of interest are photoconverted for later capture with FACS and bulk sequencing.[14] Despite these advances, even this approach leads to the loss of single-cell phenotypic variation and the powerful insights that are possible from data where perturbations and their phenotypic profiles are directly linked. As a result, profiling screening methods that perform phenotypic analysis on a single cell level have become more popular, as they allow for greater phenotypic nuance and for the discovery of phenotypes beyond the single hypothesized phenotype that drives enrichment-based screens (Figure 1.3c).

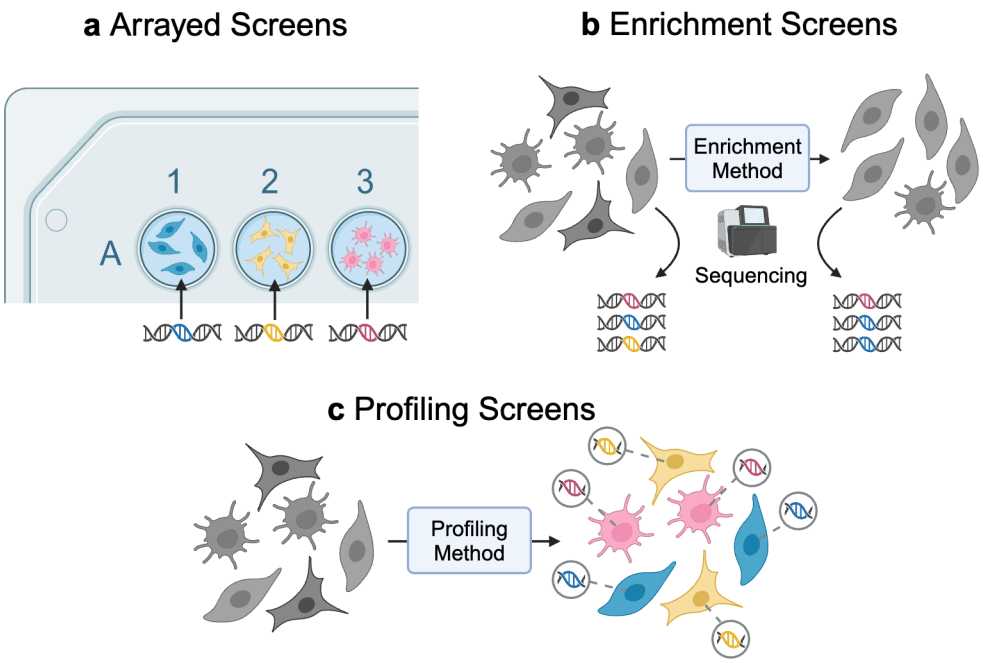


Figure 1.3: **High-throughput screening methods.**
(a) In arrayed screens, each well is treated with a single perturbation whose identity is known in advance. No additional processing is needed to identify the perturbation in each well. (b) In pooled enrichment screens, an enrichment method such as fitness selection or FACS is applied to the population. Bulk sequencing is performed on both the initial and the enriched population to measure the relative prevalence of each perturbation before and after enrichment. (c) In profiling screens, all cells in the pool are processed in order to identify the perturbation alongside phenotypic data. Adapted from Walton *et al.*[10] and created with Biorender.com.

The advent of single cell sequencing methods has led to a variety of work that couples single cell sequencing with pooled CRISPR screens, using variations of RNA sequencing to read the identity of each gRNA alongside the sequenced profile of the cell (Figure 1.4a).[15–19] These methods have proven to be highly effective in screening large perturbation panels in a discovery-driven manner, as opposed to the hypothesis-driven design of most selection-based bulk screens.

Although RNA sequencing measurements are convenient in that they easily extend to sequencing the gRNA itself, interest in pooled screens has expanded to other modalities of phenotypic measurements. To facilitate this, the construct that delivers the gRNA perturbation is modified to include a unique barcode that is designed to be compatible with the targeted measurement modality. For example, Pro-Codes expresses a combination of protein epitopes on the cell surface.[20] With this epitope

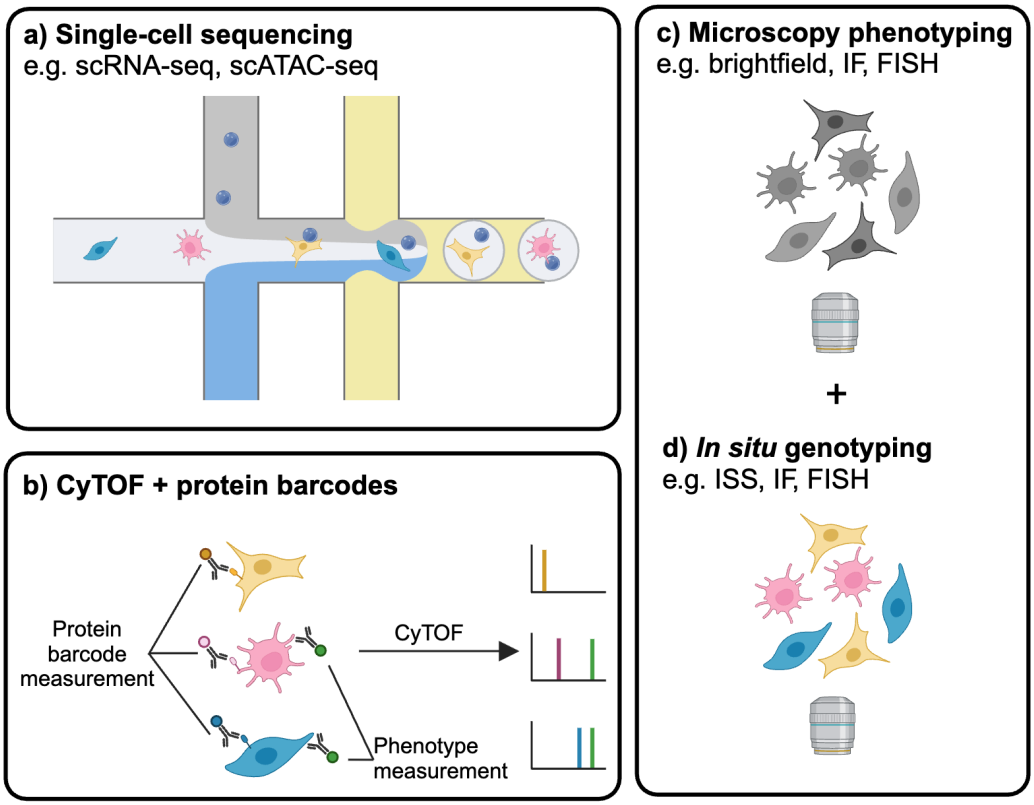


Figure 1.4: **Profiling methods for high-throughput screens.**
(a) Single-cell sequencing enables a phenotypic profile such as gene expression to be captured at the same time as identifying the perturbation in the cell. (b) Protein barcodes expressed on the surface of the cell can be read using CyTOF along with a panel of phenotype measurement antibodies. (c) Microscopy phenotyping methods can be using in sequence with an in situ genotyping method to create a combined phenotype and genotype profile for each cell. Adapted from Walton *et al.*[10] and created with Biorender.com.

barcode in place, CyTOF can be used to identify the perturbation present in each cell and to collect a high-dimensional proteomic profile of individual cells (Figure 1.4b). Beyond sequencing-based readouts, light microscopy also serves as a powerful tool for phenotypic profiling on the single-cell level. Imaging-based arrayed screens have been successfully applied to a variety of biological questions;[21–26] however, applying the same imaging-based profiling to pooled screens introduces an additional challenge of identifying the perturbation identity of each cell while maintaining its association with the phenotypic profile (Figure 1.4c,d). Deep learning methods offer a potential tool to facilitate the development, feasibility, and analysis of multimodal dataset collection and analysis.

1.3 Deep learning enables processing of large amounts of biological data

To fully maximize the potential of these multimodal data sets, data analysis pipelines are needed to extract the phenotypic profiles of single cells. The first task in most data intake pipelines is to identify the position and shape of each cell in the field of view (FOV) using methods for object detection and segmentation. Depending on the composition of the dataset, additional processing may be necessary, including object tracking for dynamic datasets or spot detection for spatial transcriptomics. Ultimately, the goal is to extract a processed data set in which each cell is represented as a row in an array with a set of associated features that describe the phenotypic presentation of the cell. For many of these tasks, deep learning has emerged as a highly effective method to reliably perform routine processing tasks on a variety of datasets.[27] The foundation of any deep learning task is a training dataset, which is made up of pairs of inputs to the model with the desired model output. For most biological problems that are suited to deep learning, training dataset generation remains a major barrier to developing new methods. Therefore, the development of methods and pipelines to efficiently collect, process, annotate, and collate large amounts of high-quality data to feed into the model is essential to developing a model that can efficiently and accurately determine changes in cell phenotype that are outside the normal bounds for a given cell type and link those changes to a given change in transcription or perturbation.

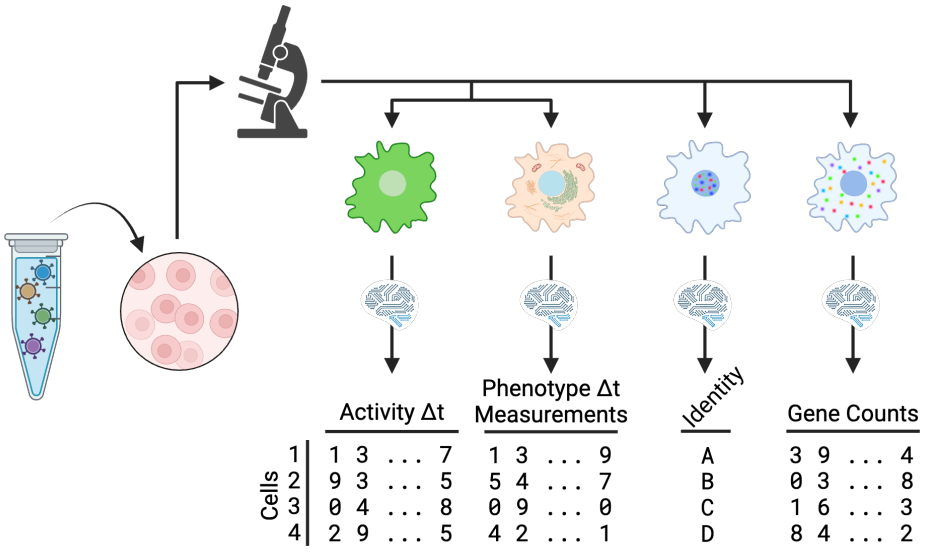


Figure 1.5: **An experimental platform to combine multimodal measurements and perturbations.**
A pooled perturbation library is introduced to a population of cells prior to utilizing microscopy, both live and fixed, to capture a variety of phenotype measurements. Additionally staining reveals the identity of the perturbation in each cell. After processing with deep learning models, the final dataset correlates the perturbed gene in each cell with the corresponding phenotypic profile of the cell. Created with Biorender.com.

1.4 Summary

In this thesis, I present key advances towards the building of a unified experimental platform to fully decode cell state (Figure 1.5). This platform leverages light microscopy to capture multimodal measurements of cell state while introducing support for high-throughput perturbation screening. Finally, the platform is accompanied by a suite of deep learning tools to enable quantitative analysis of the rich datasets collected.

In Chapter 2, I will describe Caliban, our deep learning method for nuclear segmentation and cell tracking. This work takes an end-to-end approach to the problem of cell tracking by starting with the task of dataset development and ending with the deployment of the model for use by the community. In Chapter 3, I present our new method of optical barcodes for pooled perturbation screens. Finally, in Chapter 4, I propose an experiment to explore the interdependencies of signaling pathway dynamics using the previously described cell tracking and optical barcoding methods.

References

1. Regot, S., Hughey, J. J., Bajar, B. T., Carrasco, S. & Covert, M. W. High-sensitivity measurements of multiple kinase activities in live single cells. *Cell* **157,** 1724–1734 (2014).

2. Hochbaum, D. R. *et al.* All-optical electrophysiology in mammalian neurons using engineered microbial rhodopsins. *Nature Methods* **11,** 825–833 (2014).

3. Schmitt, D. L. Imaging Subcellular AMPK Activity Using an Excitation-Ratiometric AMPK Activity Reporter. *Current Protocols* **3,** e771 (2023).

4. Purvis, J. E. & Lahav, G. Encoding and decoding cellular information through signaling dynamics. *Cell* **152,** 945–956 (2013).

5. Shamshirgaran, Y., Liu, J., Sumer, H., Verma, P. J. & Taheri-Ghahfarokhi, A. Tools for efficient genome editing; ZFN, TALEN, and CRISPR. *Applications of Genome Modulation and Editing,* 29–46 (2022).

6. Shalem, O., Sanjana, N. E. & Zhang, F. High-throughput functional genomics using CRISPR–Cas9. *Nature Reviews Genetics* **16,** 299–311 (2015).

7. Cong, L. *et al.* Multiplex genome engineering using CRISPR/Cas systems. *Science* **339,** 819–823 (2013).

8. Mali, P. *et al.* RNA-guided human genome engineering via Cas9. *Science* **339,** 823–826 (2013).

9. Gilbert, L. A. *et al.* Genome-scale CRISPR-mediated control of gene repression and activation. *Cell* **159,** 647–661 (2014).

10. Walton, R. T., Singh, A. & Blainey, P. C. Pooled genetic screens with image-based profiling. *Molecular Systems Biology* **18,** e10768 (2022).

11. Parnas, O. *et al.* A genome-wide CRISPR screen in primary immune cells to dissect regulatory networks. *Cell* **162,** 675–686 (2015).

12. Shalem, O. *et al.* Genome-scale CRISPR-Cas9 knockout screening in human cells. *Science* **343,** 84–87 (2014).

13. Wang, T., Wei, J. J., Sabatini, D. M. & Lander, E. S. Genetic screens in human cells using the CRISPR-Cas9 system. *Science* **343,** 80–84 (2014).

14. Yan, X. *et al.* High-content imaging-based pooled CRISPR screens in mammalian cells. *Journal of Cell Biology* **220.** photoactivatable, e202008158 (2021).

15. Dixit, A. *et al.* Perturb-Seq: dissecting molecular circuits with scalable single-cell RNA profiling of pooled genetic screens. *Cell* **167,** 1853–1866 (2016).

16. Jaitin, D. A. *et al.* Dissecting immune circuits by linking CRISPR-pooled screens with single-cell RNA-seq. *Cell* **167,** 1883–1896 (2016).

17. Datlinger, P. *et al.* Pooled CRISPR screening with single-cell transcriptome readout. *Nature Methods* **14,** 297–301 (2017).

18. Adamson, B., Norman, T. M., Jost, M. & Weissman, J. S. Approaches to maximize sgRNA-barcode coupling in Perturb-seq screens. *BioRxiv,* 298349 (2018).

19. Rubin, A. J. *et al.* Coupled single-cell CRISPR screening and epigenomic profiling reveals causal gene regulatory networks. *Cell* **176,** 361–376 (2019).

20. Wroblewska, A. *et al.* Protein barcodes enable high-dimensional single-cell CRISPR screens. *Cell* **175,** 1141–1155 (2018).

21. Neumann, B. *et al.* Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes. *Nature* **464,** 721–727 (2010).

22. Collinet, C. *et al.* Systems survey of endocytosis by multiparametric image analysis. *Nature* **464,** 243–249 (2010).

23. Orvedahl, A. *et al.* Image-based genome-wide siRNA screen identifies selective autophagy factors. *Nature* **480,** 113–117 (2011).

24. Karlas, A. *et al.* Genome-wide RNAi screen identifies human host factors crucial for influenza virus replication. *Nature* **463,** 818–822 (2010).

25. Mercer, J. *et al.* RNAi screening reveals proteasome-and Cullin3-dependent stages in vaccinia virus infection. *Cell Reports* **2,** 1036–1047 (2012).

26. Agaisse, H. *et al.* Genome-wide RNAi screen for host factors required for intracellular bacterial infection. *Science* **309,** 1248–1251 (2005).

27. Moen, E. *et al.* Deep learning for cellular image analysis. *Nature Methods* **16,** 1233–1246 (2019).

Chapter 2

CALIBAN: ACCURATE CELL TRACKING AND LINEAGE CONSTRUCTION IN LIVE-CELL IMAGING EXPERIMENTS WITH DEEP LEARNING

1. Schwartz, M. S. *et al.* Caliban: Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning. *bioRxiv* (2023).

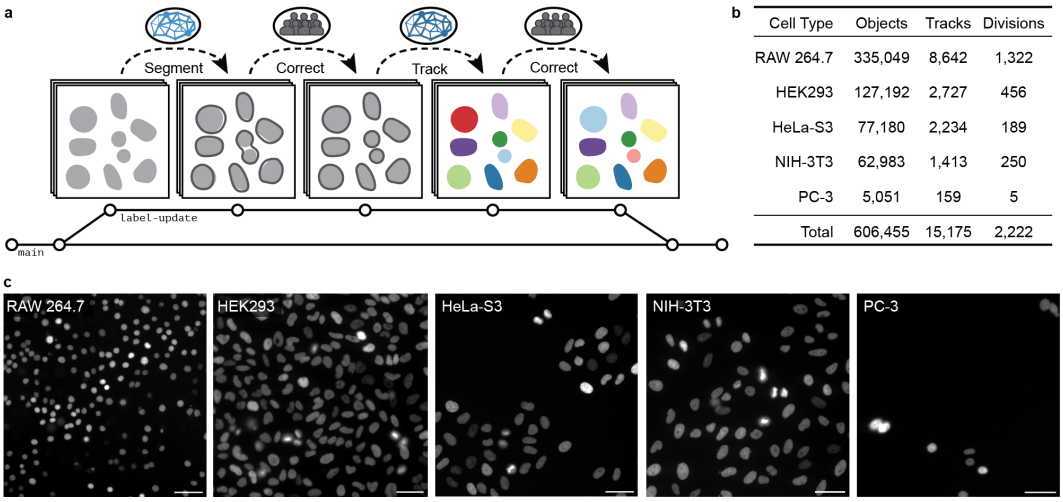
2.1 Introduction

Live-cell imaging, in which cells are imaged over time with light microscopy, provides a window into the dynamic behavior of living cells. Data generated by this class of experiments have shed light on numerous cellular processes, including cellular heterogeneity,[1–4] cell division,[5,6] morphological transitions,[7–11] and signal transduction.[12–15] New technologies that pair perturbations with imaging have led to renewed interest in using imaging to phenotype cellular dynamics.[16–20] While powerful, live-cell imaging data present a significant challenge to rigorous quantitative analysis. Central to the analysis of these data is single-cell analysis, where each cell is detected and tracked over time. Accurate solutions to these two tasks—cell detection and tracking—are essential components of every live-cell imaging analysis pipeline.

Modern deep learning methods offer a compelling path to general solutions to the computer vision problems raised by cellular imaging data. While powerful, the performance of these methods is limited by the availability of labeled data. Researchers have made substantial progress in cell segmentation, primarily because of the increased availability of labeled data and the development of human-in-the-loop (HITL) labeling methodology for static images.[21–23] Progress in deep learning solutions to cell tracking has been more limited due to a lack of similar data resources and methodology for dynamic data. Existing datasets (Table 2.1) are limited in their scope and scale,[24–31] whereas simulated datasets have not yet proven capable of creating high-performing models.[24,32,33] Further, existing datasets are limited in the resolution of their labels (e.g., point labels vs. pixel-level segmentation labels), trajectory length (the number of frames over which a cell is tracked), and the number of mitotic events (Table 2.1). These limitations are understandable, given

the time-consuming nature of labeling dynamic movies. Not only must each cell be segmented in a temporally consistent way, but lineage information must also be captured by tracking cells over time and labeling cell division events. Existing labeling methodology that has proven scalable for static images has yet to be extended at scale to these dynamic datasets.[34]

In this work, we applied a full-stack approach to the problem of cell tracking, with a specific focus on tracking fluorescently labeled cell nuclei in mammalian cells. Specifically, we combined an HITL approach to image labeling[22] adapted to dynamic imaging data, a novel deep learning algorithm for cell tracking, and new benchmarks for cell tracking to create a new labeled reference dataset for cell tracking. We used this dataset—DynamicNuclearNet—to develop state-of-the-art deep learning models for cell tracking. We further integrated these models into a pipeline called Caliban, which enables rapid and accurate segmentation, tracking, and lineage construction of nuclear live-cell imaging data with no manual parameter tuning. The source code described in this work is available at `https://github.com/vanvalenlab/deepcell-tf` and `https://github.com/vanvalenlab/deepcell-tracking`; datasets and pre-trained models are available through our lab's web portal `https://deepcell.org`.



Cell Type	Objects	Tracks	Divisions
RAW 264.7	335,049	8,642	1,322
HEK293	127,192	2,727	456
HeLa-S3	77,180	2,234	189
NIH-3T3	62,983	1,413	250
PC-3	5,051	159	5
Total	606,455	15,175	2,222

Figure 2.1: **Developing DynamicNuclearNet with HITL annotation of dynamic data.**
(a) The HITL process for generating labels alternates between preliminary models generating predictions and human annotators correcting errors generated by the model. This process is conducted twice: first for segmentation and again for tracking. Each update to the labeled data is versioned and saved with DVC.[35] (b) With a two-stage HITL process, we assembled DynamicNuclearNet, a dataset of segmented and tracked dynamic cell nuclei encompassing five cell lines. (c) Example images of each of the five cell lines. Scale bars, 50 μm.

2.2 HITL labeling of large-scale, dynamic imaging data

We employ two key strategies to label dynamic imaging data efficiently. First, we make use of crowdsourcing to parallelize our work. Second, we utilize an HITL approach to accelerate labeling efforts. Our approach has two phases: temporally consistent cell segmentations are generated in the first phase, whereas cells are tracked and cell divisions are labeled in the second phase (Figure 2.1a). The segmentation phase of our approach follows prior work,[22] beginning with a small seed dataset for cell segmentation generated by expert labelers. We then train a preliminary model to generate candidate segmentations, which are refined through a round of crowdsourced correction and expert quality control (QC). We used DeepCell Label, our browser-based labeling engine specifically designed for cellular images for this work. Critically, labelers were shown a sequence of five frames rather than individual frames to leverage temporal information to increase the label accuracy. After a sufficient amount of data was labeled, the model was retrained on a new dataset that combined the original seed dataset and the corrected predictions; the updated model was then used to generate subsequent segmentation labels. This cycle was repeated until model predictions matched expert predictions, as judged

Table 2.1: **Publicly available labeled datasets for two-dimensional temporal (2D+T) cell tracking.**

Name	Modality	Annotation Type	Cell Types	Objects	Tracks	Divisions	Source
Dynamic NuclearNet	Fluor. Nuclei	Nuclear Mask	5	606,455	15,175	2,222	This work
DeepSea	Phase	Nuclear Mask	3	100,000	2,576	137	Zargari *et al.*[31]
BTrack/CellX	Fluor. Nuclei	Nuclear Mask	1	-	1,032	-	Cuny *et al.*[36]
CTMC	DIC	Bounding Box	14	2,045,834	2,900	457	Anjum & Gurari[27]
C2C12	Phase	Centroid	1	135,859	23,400	7,159	Ker *et al.*[25]
CTC 2D+T	DIC	Part. WC Mask	1	-	70	18	CTC[37]
CTC 2D+T	Phase	Part. WC Mask	2	-	2,415	1,019	CTC[37]
CTC 2D+T	Fluor. Nuclei	Part. Nuclear Mask	3	-	1,227	395	CTC[37]
CTC 2D+T	Fluor. WC	Part. WC Mask	2	-	128	10	CTC[37]
CTC 2D+T	Brightfield	Part. WC Mask	2	-	459	242	CTC[37]

CTMC: Cell Tracking with Mitosis Detection Dataset Challenge, ISBI: International Symposium on Biomedical Imaging, CTC: Cell Tracking Challenge, Fluor: fluorescent, DIC: differential interference contrast, WC: whole cell, Part: partial.

by qualitative comparison and quantitative metrics (Section 2.5.5). In the tracking phase, labelers were given a complete movie (42–71 frames) and tasked with tracking cells and identifying cell divisions. This phase was achieved through iterative cycles of model prediction, crowdsourced correction, expert QC, and model updating. For this task, we extended DeepCell Label to include tools for labeling cell lineages and divisions. To coordinate this multi-stage dataset development process, we implemented a data and model versioning system using Data Version Control (DVC),[35] which acts alongside Git to track each data file and its associated metadata (Figure 2.1a). By automating these file associations, we removed the need for an expert user to manage the labeling pipeline and keep track of various computational notebooks or scripts.

Using this methodology, we built DynamicNuclearNet, a segmented and tracked

dataset of fluorescently labeled cell nuclei spanning five different cell lines. This dataset contains approximately 600,000 unique nuclear segmentations assembled into over 15,000 trajectories with over 2,200 division events (Figure 2.1b,c). Each trajectory begins at the cell's appearance in the FOV or birth as a daughter cell and ends when the cell disappears by leaving the FOV, dying or dividing. While it is expensive to generate pixel-level masks for each cell compared to other types of labels (e.g., centroids or bounding boxes), these masks facilitate numerous downstream analysis steps, such as quantifying signaling reporters or nuclear morphology. The 2,200 division events in our dataset surpass all previous annotation efforts that utilize nuclear segmentation masks (Table 2.1), which allows us to incorporate cell division detection into our deep-learning-based cell-tracking method.

2.3 Accurate nuclear segmentation and tracking with Caliban

In tandem with the labeling methodology advances described above, we developed Caliban, an integrated solution to nuclear segmentation and tracking. Caliban employs a tracking-by-detection approach in which cells are first identified in each frame by a deep learning model; these detections are then used to reconstruct a lineage tree that connects cells across frames and through cell division events. For reconstruction of lineage trees, we utilize a deep learning model inspired by Sadeghian *et al.*,[41] which encodes temporal dependencies for multiple features of each object. In this approach, accurate cell detection and segmentation are essential to producing faithful lineage reconstructions. To this end, we have combined our prior work on cell segmentation[22,42] with DynamicNuclearNet and a comprehensive benchmarking framework to train an accurate deep learning model for nuclear segmentation as part of Caliban.

The processing steps for Caliban are shown in Fig. 2.2a. Raw images are passed through the nuclear segmentation model to produce cell masks. These masks are used to extract features for each cell, while the centroids are used to construct an adjacency matrix to identify cells in close proximity (< 64 pixels, 41.6 μm). These features and the adjacency matrix are fed into a neighborhood encoder model, which uses a graph attention network[38,39] to generate feature vectors that summarize information about a cell's—and its neighbors'—appearance, location, and morphology (Figure 2.2b). These feature vectors are then fed into a tracking model that causally integrates temporal information and performs a pairwise comparison of each cell's feature vector across frames to produce an effective probability score indicating whether two cells are the same cell, are different cells, or have a parent–child

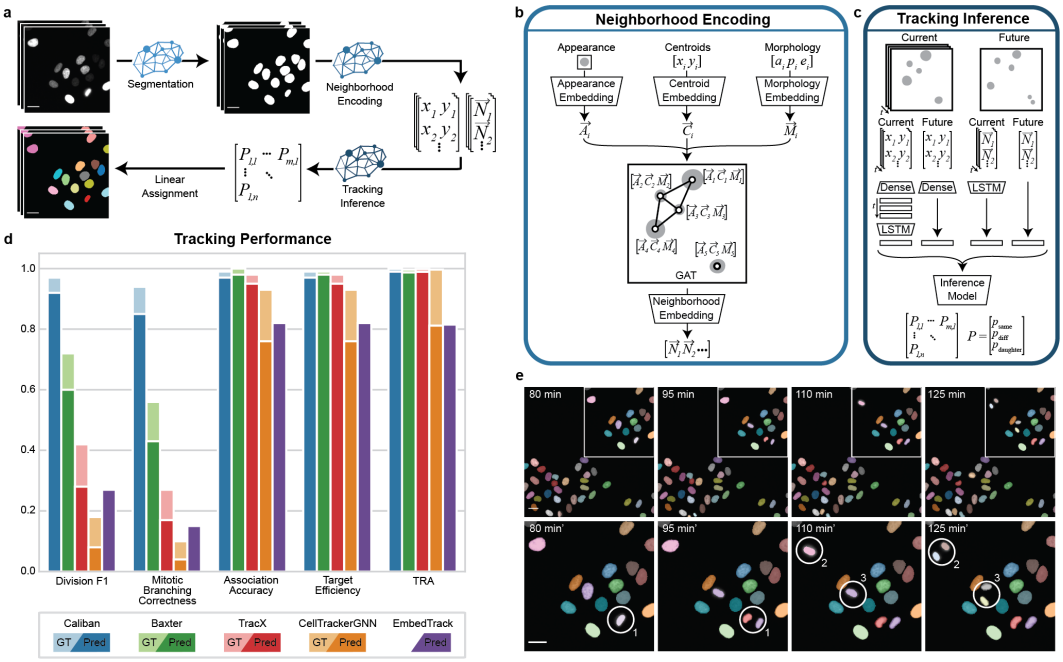


Figure 2.2: **A deep learning approach to cell segmentation and tracking using Caliban.**
(a) Caliban takes a movie of fluorescently labeled nuclei as input and then generates a nuclear segmentation mask for each frame. Features for each cell in a frame are extracted and passed through a neighborhood encoder model to generate a vector embedding for each cell. These embeddings and cell positions are passed into the tracking inference model, which predicts the probability that each pair of cells between frames is the same, is different, or has a parent–child relationship. These probabilities are used as weights for linear assignment to construct cell lineages on a frame by frame basis. (b) The neighborhood encoding model takes as input an image of each cell, its centroid position, and three metrics of morphology (area, perimeter, and eccentricity). A vector embedding of each input is used as node weights in a graph attention network,[38,39] where edges are assigned to cells within 64 pixels (41.6 μm) of each other. The final neighborhood embedding for each cell captures the appearance of that cell and its spatial relationship with its neighbors in that frame. (c) The tracking inference model performs pairwise predictions on cells in frame t_n to cells in frame t_{n+1}. The model is given neighborhood embeddings and centroid positions of cells in the previous seven frames $[t_{n-7}, t_n]$ to compare with cells in frame t_{n+1}. The temporal context of the previous seven frames is modeled using long short-term memory (LSTM) layers.[40] Ultimately, the model outputs a set of effective probabilities (p_{same}, p_{diff}, and $p_{\text{parent-child}}$) for each pair of cells between frame t_n and frame t_{n+1}. (d) The performance of Caliban and that of four other tracking methods were evaluated on the test split of DynamicNuclearNet. Tracking performance on ground truth segmentations is excluded for EmbedTrack because it is an end-to-end method that generates segmentations as part of tracking. TRA: tracking accuracy in the Cell Tracking Challenge. (e) A sample montage from DynamicNuclearNet with predictions from Caliban. Circles highlight the correct identification of three division events. Scale bars, 26 μm.

relationship (Figure 2.2c). Separating our tracking model into two pieces also facilitates rapid and scalable inference. During inference, the computationally expensive neighborhood encoder model can be run on all frames in parallel, leveraging GPU acceleration, followed by the lightweight tracking inference model, which is run on a frame-by-frame basis. The tracking inference model assigns lineages to cells by comparing the feature vectors of the last frame of existing lineages with the feature vectors of candidate cells in the current frame; model predictions are used with the Hungarian algorithm[43,44] to complete the assignment. To accommodate the entry and exit of cells in the linear assignment framework, we create a "shadow object" for each cell in the frame, which allows assignments for the "birth" or "death" of cells in each frame.[44] The methods (Section 2.5.5) provide full details of the model architecture, model training, postprocessing, and hyperparameter optimization.

In addition to Caliban, we also developed a comprehensive framework for evaluating tracking performance as part of this work. Most existing metrics for cell tracking focus on the quality of linkages between cells in lineage trees.[45–48] While useful, these approaches mask the method's performance on cell division. Many downstream analyses rely on accurate division detection; however, the relative rarity of division events makes them difficult to assess with summary metrics. To resolve this issue, we implemented several evaluation metrics that quantify a method's performance on cell division; the details of each metric are given in the methods (Section 2.5.5).

We used these metrics in combination with prior metrics[30,45–47] to compare Caliban against four alternative algorithms for cell tracking: Baxter, CellTrackerGNN, EmbedTrack, and Trac^{x}. We focused on methods that performed well in the Cell Tracking Challenge and that could be run without manual parameter tuning including using parameters published as part of the challenge submission. Baxter implements the Viterbi algorithm.[49] CellTrackerGNN constructs a global track solution by pairing a graph neural network with an edge classifier to extract cell lineages.[50] EmbedTrack utilizes a single convolutional neural network for joint cell segmentation and tracking.[51] Trac^{x} pairs a feature-based linear assignment problem with a cell fingerprint classifier to curate tracking results.[36] We tested each algorithm on ground truth and predicted segmentations. Predicted segmentations for each method were generated with that method's segmentation model or Caliban's segmentation model if the former was unavailable. We found that Caliban outperforms all algorithms on all metrics except for Baxter on target efficiency (0.97 vs. 0.98) and association accuracy (0.97 vs. 0.98) (Figure 2.2d). Importantly, on measures of division perfor-

mance, Caliban performs substantially better than all previously published methods. This performance boost is primarily attributable to Caliban's cell-tracking capability (Figure 2.2e), as the performance boost is present when tracking is performed on ground truth segmentations. Complete benchmarking results are shown in Table S2.1.

To increase the accessibility of Caliban, we have made Caliban available through our lab's GitHub (`https://github.com/vanvalenlab/deepcell-tf`) and the DeepCell web portal (`https://deepcell.org`). Converting algorithms into robust, user-friendly software can be expensive and time-consuming, particularly for cell-tracking algorithms, given the large size and varied nature of the data on which these algorithms must operate.[24,32,34,52] Here, we leverage our prior work in developing the DeepCell Kiosk,[53] a cloud-native, scalable software platform for cellular image analysis pipelines that use deep learning methods. Our cloud deployment of Caliban was facilitated by our emphasis on inference speed, scalability, and accuracy during method development. We believe that the availability of Caliban in both local and cloud versions will make this tool accessible to the broader life science community. Further, the open-source datasets, models, and benchmarking tools will facilitate future method development.

2.4 Discussion

Live-cell imaging is a transformative technology critical to elucidating cellular proliferation, migration, and other dynamic phenomena. The utility of this technology has long been limited by our ability to extract quantitative, single-cell information from these movies. In this work, we have made a significant step toward solving the computer vision challenges of dynamic cellular imaging data. By extending scalable, HITL labeling frameworks[21,22,54–57] to dynamic data, we have generated a labeled dataset of over 15,000 cellular trajectories and 2,200 cell divisions. We demonstrated that these labeled data can power accurate nuclear segmentation and tracking models. We further showed that these models can be combined into an integrated pipeline and deployed on a cloud-native platform for scalable, user-friendly inference.

While we achieved impressive performance on nuclear segmentation and tracking, several areas of improvement exist for future work. First, accurate cell segmentation remains a performance bottleneck, as highlighted by the difference in the performance of tracking methods on ground truth and deep-learning-generated segmenta-

tions (Figure 2.2). Additional performance gains on segmentation will likely arise via methods that leverage temporal information to improve performance. Newer segmentation methods that enable the segmentation of overlapping objects[23,58]—a limitation of all cell segmentation methods that currently see wide use[59]—may also help close this gap. Second, while the dataset we have collected is impressive in scale, its diversity remains limited compared with the full space of live-cell phenotypes. Our focus on cell nuclei allowed us to develop a new labeling methodology for dynamic datasets; creating a dataset similar in label scale and quality for whole cells will likely be the focus for future work. To this end, we have already extended the features available in DeepCell Label to support annotation and tracking of overlapping cells. Moreover, dynamic cell phenotypes change substantially in the setting of perturbations. Such shifts in data distribution are expected to degrade cell segmentation and tracking performance; the best approach for mitigating this issue is to expand the space of labeled data to capture these phenotypes. Doing so will require expanding our labeling framework to capture more dynamic phenotypes (e.g., cell death). We note that imaging data from pooled optical screens[16,18,20,60,61] may also be a valuable path for generating images of perturbed cell phenotypes at scale.

Our work contains several lessons for the community of researchers developing deep learning methods for cellular image analysis. First, our work highlights the importance of data labeling methodology and data scale. By developing a scalable approach for labeling dynamic live-cell imaging data, we have compiled a pixel-level labeled dataset that is substantially larger than previous datasets. The increased scale of the data allowed us to compile enough examples of cell divisions—a critical but rare dynamic event—to enable accurate detection by deep learning models. While models trained on sparsely labeled data can be effective,[34,62] increasing the amount of labeled data is essential to creating models that generalize across datasets. Second, our study demonstrates the importance of informative benchmarks. As highlighted by our work and that of others,[30,62,63] accurate cell division detection is one of the most challenging aspects of cell tracking but is critical to constructing cell lineages. This task is challenging for supervised methods, largely because of the class imbalance—cell divisions represent a relatively minor fraction of linkages in cell lineage trees. Aggregate metrics for cell tracking mask cell division events, making it difficult to judge performance gains during and after algorithmic method development.[45] Combining aggregate metrics with specific, informative metrics creates a more complete picture of performance and is critical to crafting methods that can

be used in production. Finally, this work underscores the value of model scalability. While accuracy is often the metric used to judge cellular image analysis methods, inference speed—and hence scalability—is equally important. Faster workflows can process substantially more data and provide a better user experience. A major focus of this work was scalability, which was achieved by crafting an architecture in which computationally expensive operations (e.g., the neighborhood encoder) were performed in parallel on specialized hardware (e.g., GPUs). Our model's scalability enables a responsive cloud deployment—a typical dataset (10,000 cell detections over 30 frames) can be processed in $\sim$ 40 s on an A6000 GPU, with much of the processing time taken by segmentation (Figure S2.1).

In conclusion, our work provides the live-cell imaging community with an accessible, accurate method for reconstructing single-cell lineages from dynamic imaging experiments. We believe our work will facilitate the analysis of cell phenotypes and behaviors in a wide variety of high-throughput imaging experiments.

2.5 Materials and methods

2.5.1 Cell culture

We used five mammalian cell lines (NIH-3T3, HeLa-S3, HEK293, RAW 264.7, and PC-3) to collect training data. All lines were acquired from the American Type Culture Collection. We cultured the cells in Dulbecco's modified Eagle's medium (DMEM; Invitrogen; RAW 264.7, HEK293, and NIH-3T3) or F-12K medium (Caisson; HeLa-S3 and PC-3) supplemented with 2 mM L-glutamine (Gibco), 100 U/mL penicillin, 100 μg/ml streptomycin (Gibco), and either 10% calf serum (Colorado Serum Company) for NIH-3T3 cells or 10% fetal bovine serum (FBS; Gibco) for all other cells.

2.5.2 Live imaging

Before imaging, cells were seeded in fibronectin-coated (10 μg/mL; Gibco) glass 96-well plates (Nunc or Cellvis) and allowed to attach overnight. We performed nuclear labeling via prior transduction with H2B-iRFP670 (HeLa, RAW 264.7), H2B-mClover (HEK293, NIH/3T3), and H2B-mCherry (PC-3). The media was removed and replaced with imaging media (FluoroBrite DMEM (Invitrogen) supplemented with 10 mM HEPES (Sigma-Aldrich), 10% FBS (Gibco), 2mM L-glutamine (Gibco)) at least 1 h before imaging. We imaged cells with a Nikon Ti-E or Nikon Ti2 fluorescence microscope with environmental control (37°C, 5% CO_2) and controlled by Micro-Manager or Nikon Elements. We acquired images at 5- to 6-min

intervals with a 20x objective (40x for RAW 264.7 cells) and either an Andor Neo 5.5 CMOS camera with 3×3 binning or a Photometrics Prime 95B CMOS camera with 2×2 binning. All data were scaled so that pixels had the same physical dimensions (0.65 μm per pixel) before training.

2.5.3 Data annotation

DeepCell Label

We previously described DeepCell Label,[22] our browser-based software for data annotation. We extended DeepCell Label to support labeling cell lineages and divisions in dynamic datasets. Additionally, we implemented a state machine that allows annotators to apply undo/redo functions during their work.

In this study, we utilized DeepCell Label in two stages in order to generate a nuclear tracking dataset. First, annotators were asked to correct nuclear segmentation labels for all frames in the dataset. Movies were broken into five frame sets for segmentation which allows annotators to leverage the temporal context present in the movie in order to improve the annotation of dividing cells. Second, after segmentation annotations were complete, annotators were asked to label the nuclear segmentation masks such that a single cell maintains the same label across frames. Additionally, all division events were annotated with the connection of the parent cell to each daughter cell. An expert annotator reviewed all annotated movies before incorporation into the training dataset. The appendix provides a user manual for DeepCell Label (Appendix A), along with sample instructions for segmentation (Appendix B) and tracking (Appendix C) corrections. Annotations were conducted by a team of four annotators and two expert reviewers. Each movie was annotated by a single annotator and approved by a single expert eliminating the need for resolving differences between two independent annotators.

Data versioning with DVC

Each labeled movie was versioned and tracked with DVC.[35] We recorded additional metadata in each `.dvc` file, including the data dimensions, annotation progress, and data source. These metadata enabled automatic data processing for generating segmentation and tracking predictions as well as launching annotation tasks.

2.5.4 Nuclear segmentation

Deep learning model architecture

The deep learning model for nuclear segmentation was based on the design of feature pyramid networks.[64,65] The network was constructed from an EfficientNetV2L backbone[66] connected to a feature pyramid. Input images were concatenated with a coordinate map before entering the backbone. We used backbone layers C1–C5 and pyramid layers P1–P7. The final pyramid layers were connected to three semantic segmentation heads that predict transforms of the label image.

Label image transforms

For each image, we used a deep learning model to predict three different transforms, as inspired by previous work.[22,67,68] The first transform predicted whether a pixel belongs to the foreground or background, known as the “foreground–background transform". The second transform predicted the distance of each pixel in a cell to the center of the cell and is called the “inner distance". If the distance between a pixel and the center of the cell is r, then we compute the transform as $\frac{1}{1+\alpha\beta r}$, where $\alpha = \frac{1}{\sqrt{\text{cell area}}}$ and β is a hyperparameter set to 1.[22] The final transform was the “outer distance," which is the Euclidean distance transform of the labeled image. The loss function was computed as the sum of the mean squared error on the inner and outer distance transforms and the weighted categorical cross-entropy[69] on the foreground–background transform. The cross-entropy term was scaled by 0.01 prior to the sum.

Preprocessing

Each image was required to have a minimum of one labeled object. Additionally, each image was normalized using contrast-limited adaptive histogram equalization with a kernel size equal to 1/8 of the image size to ensure that all images have the same dynamic range.[70]

Postprocessing

We fed two of the three model outputs, the inner and outer distance, into a marker-based watershed method[71] to convert the continuous model outputs into a discrete labeled image in which each cell is assigned a unique integer. We applied a peak-finding algorithm[72] with a radius of ten pixels and a threshold of 0.1 to the inner distance prediction in order to determine the centroid location of each cell. Next, we

generated the cell mask image by applying the watershed algorithm to the inverse outer distance prediction with the centroids as markers and a threshold of 0.01.

Model training and optimization

Training data were augmented with random rotations, crops, flips, and scaling to improve the diversity of the data. We used 70% of the data for training, 20% for validation, and 10% for testing. The model was trained using the Adam optimizer[73] with a learning rate of 10^{-4}, a clipnorm of 10^{-3}, and a batch size of sixteen images; training was performed for sixteen epochs. After each epoch, the learning rate was adjusted using the function $\text{lr} = \text{lr} \times 0.99^{\text{epoch}}$. Additionally, if the loss of the validation data did not improve by more than 10^{-3} after five epochs, the learning rate was reduced by a factor of 0.01.

To optimize the model's performance on nuclear segmentation, we tested ten backbones: ResNet50,[74] ResNet101,[74] EfficientNetB2,[75] EfficientNetB3,[75] EfficientNetB4,[75] EfficientNetV2M,[66] EfficientNetV2L,[66] EfficientNetV2B1,[66] EfficientNetV2B2,[66] and EfficientNetV2B3.[66] Additionally, we explored the optimal set of pyramid layers: P1–P7 and P2–P7.

Evaluation

To fully evaluate the performance of our segmentation model, we developed a set of object-based error classes that assess the model on a per-object basis as opposed to a per-pixel basis. This framework provided a perspective on model performance that reflects downstream applications. First, we built a cost matrix between cells in the ground truth and cells in the prediction, where the cost is one minus the intersection over union (IoU) for each pair of cells. We performed a linear sum assignment on this cost matrix, with a cost of 0.4 for unassigned cells, to determine which cells were correctly matched between the ground truth and prediction. For all remaining cells, we constructed a graph in which an edge was established between a ground truth and a predicted cell if the IoU was greater than zero. For each subgraph, we classified the error type based on the connectivity of the graph. Nodes without edges corresponded to a false positive or negative if the graph contained only a predicted or ground truth cell, respectively (Figure S2.2a–c)). A single predicted node connected to multiple ground truth nodes indicated a merge error (Figure S2.2d). Conversely, a single ground truth node connected to multiple predicted nodes was a split error (Figure S2.2e). Finally, any subgraphs that contain multiple ground truth and

predicted nodes were categorized as “catastrophe" (Figure S2.2f). The resulting error classes can be used to calculate a set of summary statistics, including recall, precision, and F1 score by using the true positive, false positive and false negative classes. The remaining error classes can be used to calculate (1) the number of missed detections resulting from a merge, (2) the number of gained detections resulting from a split, (3) the number of true detections involved in a catastrophe, and (4) the number of predicted detections involved in a catastrophe.

2.5.5 Cell tracking

Linear assignment for tracking

Our tracking algorithm drew inspiration from Jaqaman *et al.*,[44] where tracking was treated as a linear assignment problem. To solve the tracking problem, we first constructed a cost function for possible pairings across frames. The tracking problem was then reduced to the selection of one assignment out of the set of all possible assignments that minimizes the cost function. This task can be accomplished with the Hungarian algorithm.[76] One complicating factor of biological object tracking is that objects can appear and disappear, which leads to an unbalanced assignment problem. Cells can disappear by either moving out of the FOV or dying. Similarly, cells can appear by moving into the FOV or dividing into two daughter cells from one parent cell. In the context of the linear assignment problem, one can preserve the runtime and performance by introducing a “shadow object” for each object in the two frames that represent an opportunity for objects to “disappear” (if an object in frame t_n is matched with its shadow object in frame t_{n+1}) or “appear" (if an object in frame t_{n+1} is matched with its shadow object in frame t_n).[44] Assuming that mitotic events can be accommodated by a “shadow object” as well, division detection and assignment fit neatly into this framework. This framework can also accommodate cells that disappear from the FOV and reappear, by allowing unmatched cells from prior frames that were not assigned to cell division events to participate in the assignment. With the annotated trajectories and divisions from our dataset, it then becomes a matter of developing a deep learning architecture to extract an object’s features and learn an optimal cost function.

To construct our learned cost function, we cast it as a classification task. Let us suppose that we have two cells: our target cell 1 in frame t_n and cell 2 in frame t_{n+1}. Our goal was to train a classifier that takes in information about each cell and produces an effective probability indicating whether these two instances

are the same, are different, or have a parent–child relationship. To incorporate temporal information, we used multiple frames of information for cell 1 as an input to the classifier. This approach allowed us access to temporal information beyond just the two frames we are comparing. Our classifier was a hybrid deep learning model that blends recurrent, convolutional, and graph submodels; its architecture is summarized in Figure 2.2b,c. The three scores that the model outputs, (p_{same}, p_{diff}, and $p_{\text{parent-child}}$), which are all positive and sum to unity, can be thought of as probabilities. These scores were used to construct the cost matrix. If a cell in frame t_{n+1} is assigned to a shadow cell, i.e., if it "appears", then we assessed whether there is a parent–child relationship. This was done by finding the highest $p_{\text{parent-child}}$ among all eligible cells (i.e., the cells in frame t_n that were assigned to "disappear")—if this probability was above a threshold, then we made the lineage assignment.

Neighborhood encoder architecture

To capture the contextual information of each cell and its neighbors, we constructed a graph attention network.[38,39] There were three input heads to the model. The first head received images of each cell after reshaping to a standard 32×32 shape and converted these images to a vector embedding with a convolutional neural network. The second head received the centroid location of each cell. The third head received three morphology metrics for each cell: area, perimeter, and eccentricity. The latter two heads made use of fully connected neural networks to convert the inputs into vectors. We built an adjacency matrix for the graph attention network based on the Euclidean distance between pairs of cells; cells were linked if they are closer than 64 pixels (41.6 μm). The normalized adjacency matrix and concatenated embeddings were fed into a graph attention layer[38] to update the embeddings for each cell. The appearance and morphology embeddings were concatenated to the output of the graph attention layer to generate the final neighborhood embedding.

Tracking model architecture

Given cell 1 in frame t_n and cell 2 in frame t_{n+1}, the neighborhood encoder was used to generate embeddings for cell 1 in frame t_n and the previous seven frames $[t_{n-7}, t_n]$. Long short-term memory[40] layers were applied to the resulting embedding for cell 1 to merge the temporal information and to create a final summary vector for cell 1. The neighborhood encoder then generated an embedding for cell 2. Next, the vectors

for cell 1 and cell 2 were concatenated and fed into fully connected layers. The final layer applied the softmax transform to produce the final classification scores: p_{same}, p_{diff}, and $p_{\text{parent-child}}$.

Training and optimization

Both the neighborhood encoder and the inference model were jointly trained end-to-end such that the neighborhood embedding was tuned for the inference task. The model was trained on data that compare a set of frames $[t_{n-7}, t_n]$ with frame t_{n+1}. Each comparison of t_n with t_{n+1} contributed to the loss. For inference, the model was given single pairs of frames, e.g., t_n vs. t_{n+1}. Training data were augmented with random rotations and translations. We used 70% of the data for training, 20% for validation, and 10% for testing. Data splitting was performed with regards to the cell type such that each cell type is equally represented across the three splits. The model was trained using the rectified Adam optimizer[77] with a learning rate of 10^{-3}, a clipnorm of 10^{-3}, and a batch size of eight images. After each epoch, the learning rate was adjusted using the function $\text{lr} = \text{lr} \times 0.99^{\text{epoch}}$. Additionally, if the loss of the validation data did not improve by more than 10^{-4} after five epochs, the learning rate was reduced by a factor of 0.1. The model was trained over 50 epochs.

To optimize the performance of the tracking model, we tested the following parameters: graph layers (graph convolution layer, graph convolution layer with trainable skip connections, and graph attention convolution layer), distance threshold (64, 128, 256 pixels; 41.6, 83.2, 166.4 μm), crop mode (fixed and resized), birth probability, division probability, and death probability.

Evaluating tracking performance

To evaluate the tracking performance, we utilized two sets of metrics. The first set assessed the linkages between cells, whereas the second set focused on the linkages of dividing cells. For the first set of metrics, we calculated the target efficiency (TE) and association accuracy (AA).[46,47] Briefly, TE assesses the fraction of cells assigned to the correct lineage, and AA measures the number of correct linkages generated between cells.

Traditional metrics for evaluating tracking, including TE and AA, do not accurately reflect the ability of the method to identify divisions because divisions are relatively rare events. To overcome this weakness, we developed an evaluation pipeline that classifies each division event as a correct, missed, or incorrect division. Our pipeline

can handle tracking assignments on ground truth and predicted segmentations. First, we calculated the IoU between cells in the ground truth and the predictions to establish a mapping that can be used to compare tracking predictions. For each division in the ground truth, we checked the corresponding node in the prediction to determine whether it was labeled as a division. If the daughter nodes in the prediction match those in the ground truth, the division was counted as a correct division (Figure S2.3a). We have found that depending on the predicted segmentations, a division can sometimes be assigned to the frame before or after the frame that is annotated as a division in the ground truth data. We treated these shifted divisions as correct. If the predicted node was not labeled as a division, it was considered as a missed division (Figure S2.3b). Finally, if a predicted parent node was identified as a division, but the daughters do not match the ground truth daughters, the division was counted as incorrect (Figure S2.3c).

We utilized the classified divisions to calculate a set of summary statistics, including recall, precision, and F1 score. Additionally, we utilized the mitotic branching correctness (MBC) metric defined by Ulicna *et al.*,[30] calculated as follows:

$$\text{MBC} = \frac{\text{correct divisions}}{\text{correct divisions} + \text{missed divisions} + \text{incorrect divisions}}$$

2.5.6 Deployment

We previously described the DeepCell Kiosk,[53] our scalable cloud-based deployment for deep learning models. The Kiosk provides a drag-and-drop interface for model predictions currently deployed at `www.deepcell.org/predict`. To provide a seamless pipeline for nuclear segmentation and tracking, we deployed a new consumer for tracking jobs. First, each movie is split into single frames, which are distributed for nuclear segmentation. This step takes advantage of the Kiosk's ability to parallelize and scale resources to match demand. Once nuclear segmentation is complete on all frames, the masks are concatenated, and tracking is performed. The user receives a final output that contains the raw data, labeled masks, and lineages.

2.5.7 Benchmarking

We compared the performance of our model against four other algorithms: Baxter,[49] CellTrackerGNN,[50] EmbedTrack,[51] and Trac^{x}.[36] Using the test split of our dataset, we evaluated the tracking performance of each algorithm on ground truth segmentation and predicted segmentations generated by either the algorithm or Caliban. We evaluated the resulting tracking predictions using our division evaluation pipeline and evaluation software from the Cell Tracking Challenge.[45] The notebooks used

to generate benchmarks are available at `https://github.com/vanvalenlab/Caliban-2023_Schwartz_et_al`.

We evaluated Caliban's inference speed using a single GPU (NVIDIA RTX A6000) and eight CPUs (AMD EPYC 7763 64-Core Processor). The inference time was split into four sections: segmentation inference, neighborhood encoder inference, tracking inference, and linear assignment. Inference was repeated three times for each movie in the test data split.

2.5.8 Python packages

The following Python packages (listed in no particular order) were used in the course of this work: TensorFlow,[78] NumPy,[79] SciPy,[80] NetworkX,[81] scikit-learn,[82] scikit-image,[83] pandas,[84] Spektral[85] and Matplotlib.[86]

2.6 Supplemental materials

Tracking	Segmentation	DET	SEG	TRA	Div R	Div P	Div F1	MBC	AA	TE
Caliban	Caliban	**0.991**	**0.924**	**0.990**	**0.90**	**0.94**	**0.92**	**0.85**	0.97	0.97
	GT	*1.000*	*1.000*	*1.000*	*0.95*	*0.98*	*0.97*	*0.94*	0.99	*0.99*
Baxter	Caliban	0.988	0.920	0.987	0.50	0.74	0.60	0.43	**0.98**	**0.98**
	GT	0.997	0.996	0.997	0.60	0.89	0.72	0.56	*1.00*	*0.99*
Trac^{x}	Caliban	**0.991**	**0.924**	0.989	0.32	0.26	0.28	0.17	0.95	0.95
	GT	*1.000*	*1.000*	0.999	0.34	0.54	0.42	0.27	0.98	0.98
CellTrackerGNN	CellTrackerGNN	0.815	0.682	0.812	0.43	0.05	0.08	0.04	0.76	0.76
	GT	*1.000*	0.999	0.996	0.65	0.10	0.18	0.10	0.93	0.93
EmbedTrack	EmbedTrack	0.816	0.642	0.815	0.64	0.17	0.27	0.15	0.82	0.82

Table S2.1: **Benchmarking the performance of different tracking methods on the test split of DynamicNuclearNet.**
Bold font indicates the best scores on predicted segmentations. Italic font denotes the best scores on ground truth (GT) segmentations. CTC: Cell Tracking Challenge, DET: CTC detection accuracy,[45] SEG: CTC segmentation accuracy,[87] TRA: CTC tracking accuracy,[87] Div R: division recall, Div P: division precision, Div F1: division F1 score, MBC: mitotic branching correctness,[30] AA: association accuracy,[46,47] TE: target efficiency.[46,47]

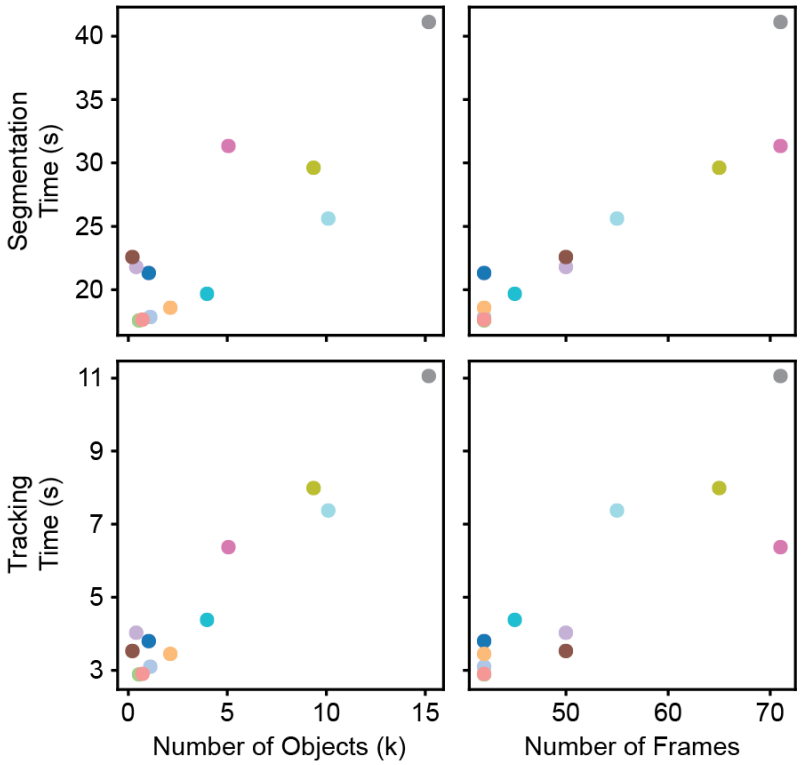


Figure S2.1: **Runtime for segmentation and tracking with Caliban.**
The total runtime for segmentation and tracking is plotted as a function of the number of objects and frames in the sample. Each point represents a movie in the test data split, with a unique color assigned to each movie.

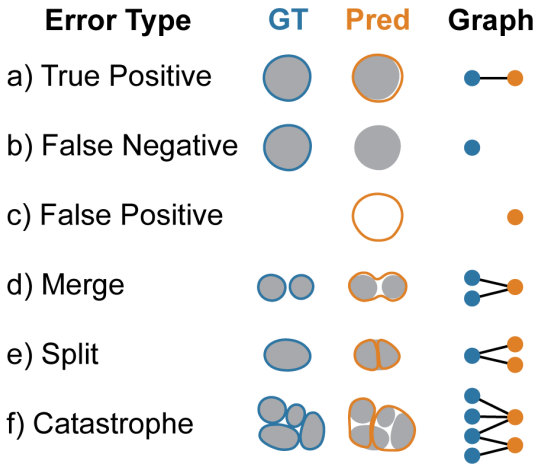


Figure S2.2: **Object-based evaluation of segmentation performance.** Segmentation predictions were evaluated based on object-level accuracy by first constructing a graph in which edges indicate an overlap between two objects. Each subgraph is then isolated and analyzed to identify the type of segmentation error present. (a) Subgraphs with one GT and one predicted node represent a true positive segmentation. Subgraphs containing only one node represent (b) a false negative if the node is GT or (c) a false positive if the node is predicted. Subgraphs with three nodes indicate (d) a merge if two GT nodes are associated with one predicted node or (e) a split if two predicted nodes are associated with one GT node. (f) Finally, all subgraphs containing more than three nodes are assigned to the catastrophe error class.

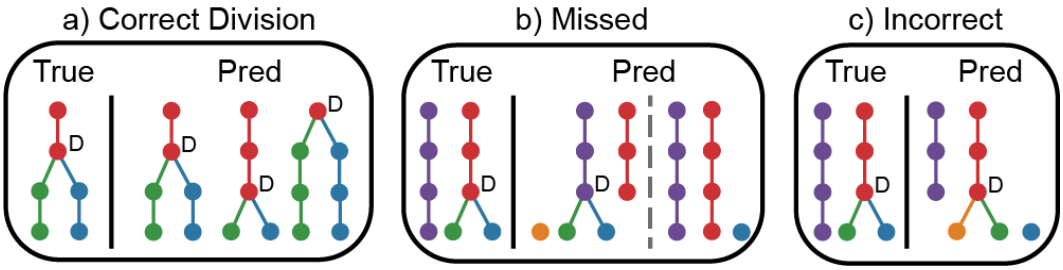


Figure S2.3: **Division-based evaluation of tracking performance.** Division events are classified as correct, missed, or incorrect based on a comparison of the true and predicted tracking graphs. (a) A division is considered correct if the prediction links the parent to the correct daughters within one frame of the GT division event. We allow divisions to shift in time because segmentation predictions can change when the cell is identified as one or two objects. (b) Divisions are identified as missed if the daughter cells are assigned to the incorrect parent or if no parent is identified. (c) A division is incorrect if the parent is assigned to only one of the correct daughter cells.

Declarations

Code availability

The software used for data labeling is available at `https://github.com/vanvalenlab/deepcell-label`. The code used for model development, cell tracking, and model deployment are available at `https://github.com/vanvalenlab/deepcell-tf`, `https://github.com/vanvalenlab/deepcell-tracking`, and `https://github.com/vanvalenlab/kiosk-console`, respectively. Finally, the code for reproducing all models and figures included in this paper is available at `https://github.com/vanvalenlab/Caliban-2023_Schwartz_et_al`. All code is released under a modified Apache license and is free for non-commercial use.

Data availability

The DynamicNuclearNet dataset is available through `deepcell.datasets` (`https://deepcell.readthedocs.io/en/master/data-gallery/dynamicnuclearnet.html#sphx-glr-data-gallery-dynamicnuclearnet-py`) for non-commercial use.

References

1. Lahav, G. *et al.* Dynamics of the p53-Mdm2 feedback loop in individual cells. *Nature Genetics* **36,** 147–150 (2004).
2. Geva-Zatorsky, N. *et al.* Oscillations and variability in the p53 system. *Molecular Systems Biology* **2** (2006).
3. Regot, S., Hughey, J. J., Bajar, B. T., Carrasco, S. & Covert, M. W. High-sensitivity measurements of multiple kinase activities in live single cells. *Cell* **157,** 1724–1734 (2014).
4. Hughey, J. J., Gutschow, M. V., Bajar, B. T. & Covert, M. W. Single-cell variation leads to population invariance in NF-κB signaling dynamics. *Molecular Biology of the Cell* **26,** 583–590 (2015).
5. Neumann, B. *et al.* High-throughput RNAi screening by time-lapse imaging of live human cells. *Nature Methods* **3,** 385–390 (2006).
6. Neumann, B. *et al.* Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes. *Nature* **464,** 721–727 (2010).
7. Huang, K. C., Mukhopadhyay, R., Wen, B., Gitai, Z. & Wingreen, N. S. Cell shape and cell-wall organization in Gram-negative bacteria. *Proceedings of the National Academy of Sciences* **105,** 19282–19287 (2008).
8. Keren, K. *et al.* Mechanism of shape determination in motile cells. *Nature* **453,** 475–480 (2008).

9. Chan, C. K., Hadjitheodorou, A., Tsai, T. Y.-C. & Theriot, J. A. Quantitative comparison of principal component analysis and unsupervised deep learning using variational autoencoders for shape analysis of motile cells. *bioRxiv* (2020).

10. Zaritsky, A. *et al.* Interpretable deep learning uncovers cellular properties in label-free live cell images that are predictive of highly metastatic melanoma. *Cell Systems* **12,** 733–747 (2021).

11. Wu, Z. *et al.* DynaMorph: self-supervised learning of morphodynamic states of live cells. *Molecular Biology of the Cell* **33** (2022).

12. Purvis, J. E. *et al.* p53 dynamics control cell fate. *Science* **336,** 1440–1444 (2012).

13. Albeck, J. G., Mills, G. B. & Brugge, J. S. Frequency-modulated pulses of ERK activity transmit quantitative proliferation signals. *Molecular Cell* **49,** 249–261 (2013).

14. Purvis, J. E. & Lahav, G. Encoding and decoding cellular information through signaling dynamics. *Cell* **152,** 945–956 (2013).

15. Lane, K. *et al.* Measuring signaling and RNA-seq in the same cell links gene expression to dynamic patterns of NF-κB activation. *Cell Systems* **4,** 458–469 (2017).

16. Feldman, D. *et al.* Optical pooled screens in human cells. *Cell* **179,** 787–799 (2019).

17. Reicher, A., Koren, A. & Kubicek, S. Pooled protein tagging, cellular imaging, and in situ sequencing for monitoring drug action in real time. *Genome Research* **30,** 1846–1855 (2020).

18. Funk, L. *et al.* The phenotypic landscape of essential human genes. *Cell* **185,** 4634–4653 (2022).

19. Walton, R. T., Singh, A. & Blainey, P. C. Pooled genetic screens with image-based profiling. *Molecular Systems Biology* **18,** e10768 (2022).

20. Carlson, R. J., Leiken, M. D., Guna, A., Hacohen, N. & Blainey, P. C. A genome-wide optical pooled screen reveals regulators of cellular antiviral responses. *Proceedings of the National Academy of Sciences* **120,** e2210623120 (2023).

21. Pachitariu, M. & Stringer, C. Cellpose 2.0: how to train your own model. *Nature Methods,* 1–8 (2022).

22. Greenwald, N. F. *et al.* Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning. *Nature Biotechnology* **40,** 555–565 (2022).

23. Kirillov, A. *et al.* Segment anything. *arXiv.* eprint: `2304.02643` (2023).

24. Ulman, V. *et al.* An objective comparison of cell-tracking algorithms. *Nature Methods* **14,** 1141–1152 (2017).

25. Ker, D. F. E. *et al.* Phase contrast time-lapse microscopy datasets with automated and manual cell tracking annotations. *Scientific Data* **5,** 1–12 (2018).

26. Tsai, H.-F., Gajda, J., Sloan, T. F., Rares, A. & Shen, A. Q. Usiigaci: Instance-aware cell tracking in stain-free phase contrast microscopy enabled by machine learning. *SoftwareX* **9,** 230–237 (2019).

27. Anjum, S. & Gurari, D. *CTMC: Cell tracking with mitosis detection dataset challenge* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (2020), 982–983.

28. Lugagne, J.-B., Lin, H. & Dunlop, M. J. DeLTA: Automated cell segmentation, tracking, and lineage reconstruction using deep learning. *PLoS Computational Biology* **16,** e1007673 (2020).

29. Su, Y.-T., Lu, Y., Liu, J., Chen, M. & Liu, A.-A. Spatio-temporal mitosis detection in time-lapse phase-contrast microscopy image sequences: A benchmark. *IEEE Transactions on Medical Imaging* **40,** 1319–1328 (2021).

30. Ulicna, K., Vallardi, G., Charras, G. & Lowe, A. R. Automated deep lineage tree analysis using a Bayesian single cell tracking approach. *Frontiers in Computer Science* **3,** 734559 (2021).

31. Zargari, A. *et al.* DeepSea: An efficient deep learning model for single-cell segmentation and tracking of time-lapse microscopy images. *bioRxiv,* 2021–03 (2021).

32. Liu, Z. *et al.* A survey on applications of deep learning in microscopy image analysis. *Computers in Biology and Medicine* **134,** 104523 (2021).

33. Wen, C. *et al.* 3DeeCellTracker, a deep learning-based pipeline for segmenting and tracking cells in 3D time lapse images. *eLife* **10,** e59187 (2021).

34. Sugawara, K., Çevrim, Ç. & Averof, M. Tracking cell lineages in 3D by incremental deep learning. *eLife* **11,** e69380 (2022).

35. Kuprieiev, R. *et al. DVC: Data Version Control - Git for Data & Models* version 2.45.1. 2023. `https://doi.org/10.5281/zenodo.7646429`.

36. Cuny, A. P., Ponti, A., Kündig, T., Rudolf, F. & Stelling, J. Cell region fingerprints enable highly precise single-cell tracking and lineage reconstruction. *Nature Methods* **19,** 1276–1285 (2022).

37. *Cell Tracking Challenge (2D)* `http://celltrackingchallenge.net/2d-datasets/`.

38. Veličković, P. *et al. Graph Attention Networks* in *International Conference on Learning Representations* (2018). `https://openreview.net/forum?id=rJXMpikCZ`.

39. Brody, S., Alon, U. & Yahav, E. How attentive are graph attention networks? *arXiv*. eprint: 2105.14491 (2021).

40. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Computation* **9,** 1735–1780 (1997).

41. Sadeghian, A., Alahi, A. & Savarese, S. *Tracking the untrackable: Learning to track multiple cues with long-term dependencies* in *Proceedings of the IEEE international conference on computer vision* (2017), 300–311.

42. Van Valen, D. A. *et al.* Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. *PLoS Computational Biology* **12,** e1005177 (2016).

43. Munkres, J. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics* **5,** 32–38 (1957).

44. Jaqaman, K. *et al.* Robust single-particle tracking in live-cell time-lapse sequences. *Nature Methods* **5,** 695–702 (2008).

45. Matula, P. *et al.* Cell tracking accuracy measurement based on comparison of acyclic oriented graphs. *PloS One* **10,** e0144959 (2015).

46. Hayashida, J., Nishimura, K. & Bise, R. *MPM: Joint representation of motion and position map for cell tracking* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), 3823–3832.

47. Nishimura, K., Hayashida, J., Wang, C., Ker, D. F. E. & Bise, R. *Weakly-supervised cell tracking via backward-and-forward propagation* in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16* (2020), 104–121.

48. Maška, M. *et al.* The Cell Tracking Challenge: 10 years of objective benchmarking. *Nature Methods,* 1–11 (2023).

49. Magnusson, K. E., Jaldén, J., Gilbert, P. M. & Blau, H. M. Global linking of cell tracks using the Viterbi algorithm. *IEEE Transactions on Medical Imaging* **34,** 911–929 (2014).

50. Ben-Haim, T. & Raviv, T. R. *Graph neural network for cell tracking in microscopy videos* in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXI* (2022), 610–626.

51. Löffler, K. & Mikut, R. EmbedTrack—Simultaneous Cell Segmentation and Tracking Through Learning Offsets and Clustering Bandwidths. *IEEE Access* **10,** 77147–77157 (2022).

52. Wolf, S., Wan, Y. & McDole, K. Current approaches to fate mapping and lineage tracing using image data. *Development* **148,** dev198994 (2021).

53. Bannon, D. *et al.* DeepCell Kiosk: scaling deep learning–enabled cellular image analysis with Kubernetes. *Nature Methods* **18,** 43–45 (2021).

54. Hughes, A. J. *et al.* Quanti. us: a tool for rapid, flexible, crowd-based annotation of images. *Nature Methods* **15,** 587–590 (2018).

55. Berg, S. *et al.* Ilastik: interactive machine learning for (bio) image analysis. *Nature Methods* **16,** 1226–1232 (2019).

56. Wolny, A. *et al.* Accurate and versatile 3D segmentation of plant tissues at cellular resolution. *eLife* **9,** e57613 (2020).

57. Ouyang, W., Le, T., Xu, H. & Lundberg, E. Interactive biomedical segmentation tool powered by deep learning and ImJoy. *F1000Research* **10,** 142 (2021).

58. Cheng, B., Misra, I., Schwing, A. G., Kirillov, A. & Girdhar, R. *Masked-attention mask transformer for universal image segmentation* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 1290–1299.

59. Hollandi, R. *et al.* Nucleus segmentation: towards automated solutions. *Trends in Cell Biology* (2022).

60. Kaufman, T. *et al.* Visual barcodes for clonal-multiplexing of live microscopy-based assays. *Nature Communications* **13,** 2725 (2022).

61. Kudo, T., Lane, K. & Covert, M. W. A multiplexed epitope barcoding strategy that enables dynamic cellular phenotypic screens. *Cell Systems* **13,** 376–387 (2022).

62. Malin-Mayor, C. *et al.* Automated reconstruction of whole-embryo cell lineages by learning from sparse annotations. *Nature Biotechnology* **41,** 44–49 (2023).

63. Ershov, D. *et al.* TrackMate 7: integrating state-of-the-art segmentation algorithms into tracking pipelines. *Nature Methods* **19,** 829–832 (2022).

64. Lin, T.-Y. *et al. Feature pyramid networks for object detection* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 2117–2125.

65. Kirillov, A., Girshick, R., He, K. & Dollár, P. *Panoptic feature pyramid networks* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), 6399–6408.

66. Tan, M. & Le, Q. *Efficientnetv2: Smaller models and faster training* in *International conference on machine learning* (2021), 10096–10106.

67. Fu, C.-Y., Shvets, M. & Berg, A. C. RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free. *arXiv.* eprint: 1901.03353 (2019).

68. Koyuncu, C. F., Gunesli, G. N., Cetin-Atalay, R. & Gunduz-Demir, C. DeepDistance: a multi-task deep regression model for cell detection in inverted microscopy images. *Medical Image Analysis* **63,** 101720 (2020).

69. *deepcell-tf* https://github.com/vanvalenlab/deepcell-tf.

70. Heckbert, P. S. *Graphics Gems* chap. 'VIII.5' (Academic Press, 2013).

71. Meyer, F. & Beucher, S. Morphological segmentation. *Journal of Visual Communication and Image Representation* **1,** 21–46 (1990).

72. Van der Walt, S. *et al.* scikit-image: image processing in Python. *PeerJ* **2,** e453 (2014).

73. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv.* eprint: 1412.6980 (2014).

74. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learningfor image recognition. *CoRR, abs/1512* **3385,** 2 (2015).

75. Tan, M. & Le, Q. *Efficientnet: Rethinking model scaling for convolutional neural networks* in *International conference on machine learning* (2019), 6105–6114.

76. Kuhn, H. W. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* **2,** 83–97 (1955).

77. Liu, L. *et al.* On the variance of the adaptive learning rate and beyond. *arXiv.* eprint: 1908.03265 (2019).

78. Martín Abadi *et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* Software available from tensorflow.org. 2015. https://www.tensorflow.org/.

79. Harris, C. R. *et al.* Array programming with NumPy. *Nature* **585,** 357–362. https://doi.org/10.1038/s41586-020-2649-2 (Sept. 2020).

80. Virtanen, P. *et al.* SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* **17,** 261–272 (2020).

81. Hagberg, A., Swart, P. & S Chult, D. *Exploring network structure, dynamics, and function using NetworkX* tech. rep. (Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008).

82. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011).

83. Van der Walt, S. *et al.* scikit-image: image processing in Python. *PeerJ* **2,** e453. ISSN: 2167-8359. https://doi.org/10.7717/peerj.453 (June 2014).

84. Pandas development team, T. *pandas-dev/pandas: Pandas* version latest. Feb. 2020. https://doi.org/10.5281/zenodo.3509134.

85. Grattarola, D. & Alippi, C. Graph Neural Networks in Tensor-Flow and Keras with Spektral. *arXiv.* eprint: 2006.12138 (2020).

86. Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* **9,** 90–95 (2007).

87. Maška, M. *et al.* A benchmark for comparison of cell tracking algorithms. *Bioinformatics* **30,** 1609–1617 (2014).

Chapter 3

DEEP-LEARNING ENABLED SPATIAL OPTICAL BARCODES

3.1 Introduction

As described in Chapter 1, pooled microscopy screens have emerged as a powerful method for merging the rich phenotyping capabilities of microscopy with high-throughput perturbation screening. However, imaging pooled populations of cells introduces an additional challenge: each cell must be linked to the perturbation that it received. Optical barcodes are a set of methods that solve this problem by encoding a unique identifier in each cell that can be read on the microscope and used to link each cell with its associated perturbation. Broadly, optical barcodes fall into two classes: (1) sequential barcodes that rely on iterations of staining and imaging to read, and (2) spatial barcodes that create a pattern within the cell (Figure 3.1).

Sequential barcodes can easily scale to large library size, but the repeated imaging rounds required to capture the barcode can make the collection of other types of imaging data challenging. Additionally, the process of staining and imaging a sequential barcode can take days, which limits overall sample throughput. Sequential optical barcodes have been implemented with several different read-out methods. The Zhuang lab adapted MERFISH to a barcode strategy in which an N-bit barcode is read through 2N rounds of sequential FISH staining.[1,2] In this scheme, each bit of the barcode is encoded by one of two fluorescent read-out probes. The read-out sequence is highly expressed as part of the perturbation construct, so no additional signal amplification is needed to read the barcode using MERFISH. The Blainey lab implemented an alternative approach to sequential barcodes using *in situ* sequencing.[3] In this scheme, the barcode is a twelve-nucleotide sequence embedded in the 3' UTR of an antibiotic resistance gene of the lentiviral construct used to deliver the library. The barcode is flanked by standard sequences that serve as targets for rolling circle amplification to increase the number of copies of the barcode in the cell before readout with twelve rounds of in situ sequencing. The epitope barcoding strategy, ProCode, which originally used CyTOF, has since been applied in microscopy modalities, including multiplexed ion beam imaging (MIBI) and multiplex immunohistochemistry consecutive staining on a single slide (MICSSS).[4–6]

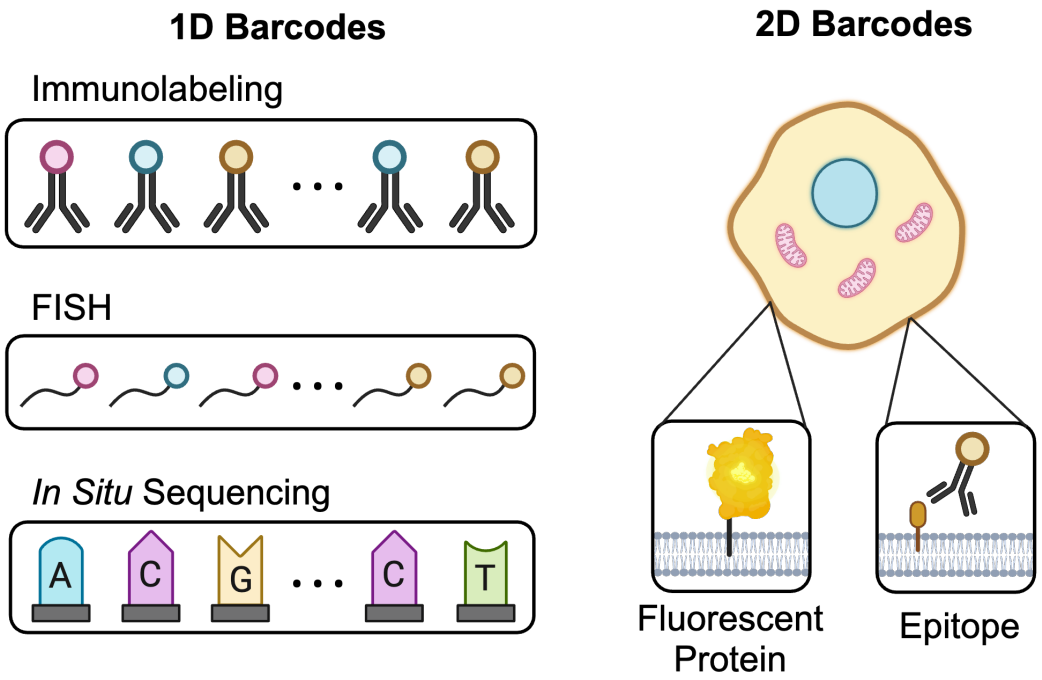


Figure 3.1: **Optical barcodes for pooled microscopy screens.**
Optical barcodes are sequences expressed in a cell that can be labeled and identified on a microscope as a unique identifier. 1D barcodes are read out through repeated rounds of staining and imaging. They can be encoded using epitopes targeted by antibodies, target sequences labeled by FISH probes or short nucleotide sequences read out using in situ sequencing. In contrast, 2D or spatial barcodes are typically labeled with a single round of imaging and combine spatial patterns in the cell with different colors using either fluorescent proteins or epitopes.

Although sequential barcode methods have been applied to screening large libraries, compatibility with other highly multiplexed downstream phenotyping methods, such as transcriptome-scale RNA FISH or cyclical immunofluorescence, has been challenging.[7] A preliminary multiomics dataset was collected using CRISPRmap, which paired barcoded gRNAs for read out using FISH with probes against 12 mRNA transcripts and antibodies against thirteen proteins imaged with IBEX.[8] This work clearly demonstrates the power of multi-omics investigations, but additionally highlights the need for future development to extend multi-omics to the transcriptome and proteome scale. Given that most experiments are limited in the amount of time dedicated to staining and imaging, any time spent reading a sequential barcode directly takes time away from phenotypic data collection.

In contrast to current implementations of sequential barcodes, spatial barcodes encode information by separating labels into subcellular compartments, which can

reduce the number of rounds of imaging required in readout. To date, implementations of spatial barcodes are based on labeling organelles. In yeast, Chen *et al.* constructed barcoded yeast populations by labeling four organelles (nucleus, vacuolar membrane, plasma membrane, and actin) with one of four fluorescent proteins, which yielded a barcode library that can encode tens of thousands of variants.[9] However, the use of four fluorescent proteins for the optical barcode precluded the use of this barcoding strategy for any reporter-based measurements. In an effort to overcome this limitation, two different methods have been published that reserve a portion of the visible spectrum for reporters and biosensors while using the remainder of the spectrum for barcodes based on fluorescent proteins conjugated to organelle localization sequences.[10,11] Finally, to allow the full spectrum of reporters to be used in barcoded experiments in mammalian cells, Kudo *et al.* introduced barcodes based on protein epitopes localized in either the nucleus or mitochondria.[12] This method, EPICode, uses three rounds of multiplexed immunochemistry to image eighteen epitopes. In any spatial barcoding strategy, the total number of variants that can be encoded in the library depends on a combinatorial relationship between the number of patterns and the number of colors. In a library design where all combinations of labels and organelles are allowed, the total library size is given by $\frac{n!}{(n-r)!}$ where n is the number of organelles and r is the number of patterns. Organelle-based barcoding methods have been successfully implemented so far only with a maximum of four organelles. As a result, any organelle-based barcode is limited in the library scale without dramatically increasing the number of colors.

In this work, we present a new spatial optical barcoding method that uses repetitive genomic sequences to substantially increase the number of spatial patterns that can be reliably created inside of cells. We pair these patterns with an endpoint, FISH-based readout that enables us to preserve the full color spectrum for live cell imaging reporters prior to reading out barcodes. Finally, we demonstrate that a deep learning classifier can accurately identify cells using this barcoding strategy. Our new spatial optical barcoding method can encode libraries encompassing 6840 variants while only requiring a single round of 3 color FISH to read the barcode identity. With the inclusion of an additional round of FISH labeling, the library scales to more than 40 million variants. This streamlined read-out approach enables our libraries to be paired with other multiplexed phenotype measurements such as seqFISH (transcriptome) or multiplexed immunofluorescence (proteome).[13–15]

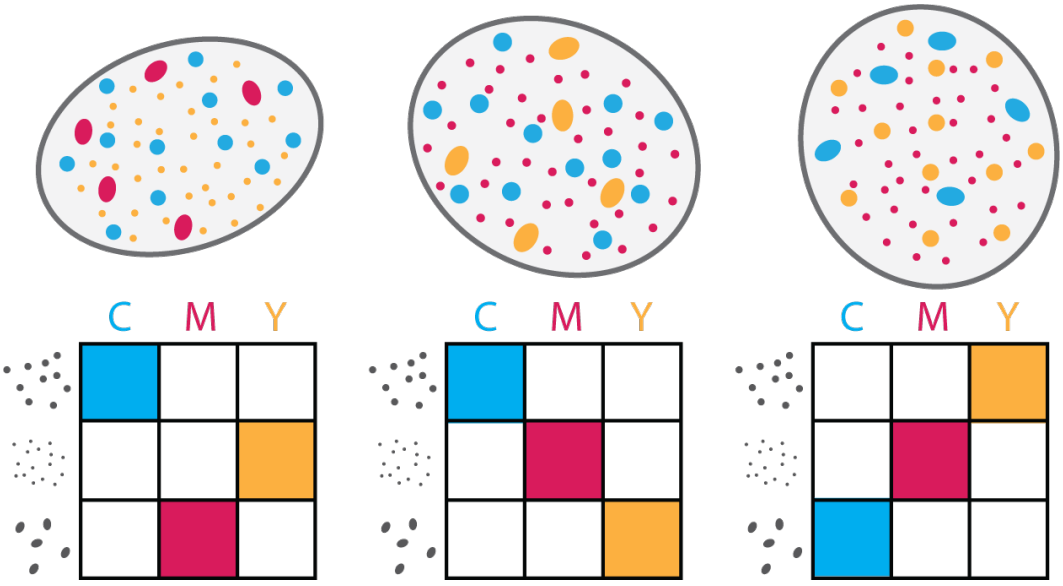


Figure 3.2: **CRISPR-dCas9 enables a scalable strategy for spatial optical barcodes.**
In each of these example nuclei, an optical barcode is constructed by pairing three distinct spatial patterns with three color assignments. By varying the combination of colors with patterns, a set of unique barcodes are created.

3.2 Method development

The development of CRISPR-Cas9 and its related variants has transformed the field's ability to introduce genetic perturbations into cells. A key variant has been deactivated Cas9 (dCas9), which binds to the genome as programmed by its associated gRNA. This method has been used to inactivate (CRISPRi) and activate (CRISPRa) gene expression. Further developments include CRISPR-imaging in which the targeting mechanism of CRISPR-dCas9 is used to label genomic sequences for live imaging.[16–18] The mechanism enabling this imaging is quite simple: repetitive sequences recruit many copies of the dCas9-gRNA complexes to the same location, creating bright and distinct spot patterns inside the nucleus.

In this work, we propose a new method for creating spatial patterns within cells using CRISPR-dCas9 to bring fluorescent tags to different repetitive sequences in the genome, creating distinct and recognizable spatial patterns. In our method, each spatial optical barcode consists of three spatial patterns labeled in three different colors (Figure 3.2). With this design, the total number of possible barcodes is $\frac{p!}{(p-c)!}$ where p is the number of distinct spatial patterns and c is the number of colors. To accommodate most standard microscopy setups, we use three colors (green, red, and far red), while reserving blue for nuclear labeling. With only 20 gRNA patterns, this

barcode strategy can create a library encoding 6840 variants that can be decoded in a single round of 3-color imaging. If we allow for two imaging rounds, we can achieve a library size of more than 40 million. To associate specific gRNAs with different readout colors, we designed a protocol for labeling gRNAs with FISH. In contrast to alternative optical barcoding strategies, our method can be read in a single round of imaging and can scale to genome-scale libraries. Ultimately, the only limit on our library size is the number of repetitive sequences in the genome that can produce distinguishable spatial patterns.

In the following sections, we describe our computational strategy for identifying gRNA sequences that produce distinct spatial patterns, as well as the corresponding protocols for testing and validating candidate sequences. We show that deep learning can be used to accurately identify spatial patterns and interpret spatial barcodes. In addition, we describe how we engineered binding sites for multicolor FISH labeling into the gRNA scaffold sequence.

3.3 Identifying and screening candidate gRNA sequences

3.3.1 Computational method for identifying potential sequences

The first challenge for developing our approach to spatial optical barcodes was identifying a set of gRNA sequences that produce distinct and visible patterns in the nucleus. We used existing gRNA sequences targeting the telomere and the α and β satellites of the centromeres as guidelines for our design criteria when selecting new gRNAs. Preliminary data collected using the CASFISH labeling method demonstrated that all three sequences could produce spatial patterns that could be visualized with standard epifluorescent microscopy (Figure 3.3). To perform a computational search for new gRNAs that bind to repetitive sequences, we considered two possible reference genomes. The first was GRCh38, which is derived from the Human Genome Project and has seen substantial improvements over the past two decades.[19] However, the methods used to assemble this genome struggle with repetitive sequences and thus fail to fully capture the satellite sequences of the centromeres, along with other highly repetitive regions. In contrast, the second reference genome we considered, T2T-CHM13, used new long-read sequencing technology to prioritize complete coverage of repetitive regions across the genome.[20] This reference genome used a complete hydatiform mole (CHM) cell line, which can be effectively considered homozygous with the exception of a few thousand heterozygous variants and a single megabase deletion. Given the importance of accurately identifying repetitive sequences that could be targeted with gRNAs, we chose to work with the

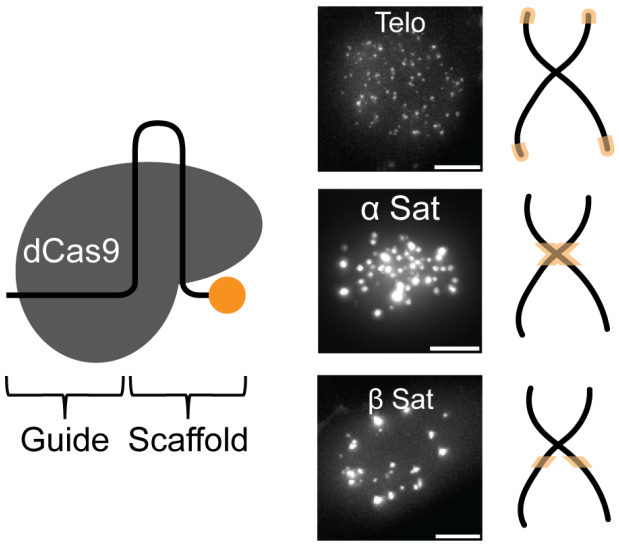


Figure 3.3: **Labeling repetitive genomic sequences using CRISPR-dCas9.** Shown here are sample images of nuclei labeled with dCas9 using gRNAs targeted against α satellite,[21] β satellite, and telomeres.[22] The β satellite gRNA sequence was designed based on the sequence published in [23]. Sample gRNA images were collected from HeLa cells stained using CASFISH, which utilizes a fluorescently labeled tracrRNA to visualize CRISPR-dCas9 binding sites. Scale bars, 10 μm.

reference genome T2T-CHM13. We expect that any cell line used in our experiments (or future experiments) will have genetic variation different from that of the CHM13 cell line. It follows that experimental validation must be performed for each cell line for all targets identified in the T2T-CHM13 genome.

A key design constraint for our barcoding method is compatibility with standard fluorescent microscopes (confocal or epifluorescent) and lower magnifications (10-40X) to facilitate rapid imaging of large numbers of cells. Although individual fluorescent molecules cannot be seen under these imaging conditions, clusters of molecules can. To mimic this expectation computationally, we first identified all binding sites within the genome for a given gRNA. We then developed a set of criteria to approximate when a set of adjacent gRNA binding sites would lead to a visible cluster on the microscope (Figure 3.4). Through a combination of hyperparameter optimization and experimental validation, we determined that a minimum of eighteen binding sites within eighteen kilobase pairs (kbp) are required to form a visible cluster. Additional binding sites that occur within 54 kbp of the first cluster were assigned to that first cluster. The parameters for this criteria were calibrated by comparing the known number of clusters experimentally observed for α satellite and telomere gRNAs with the number of clusters predicted by our

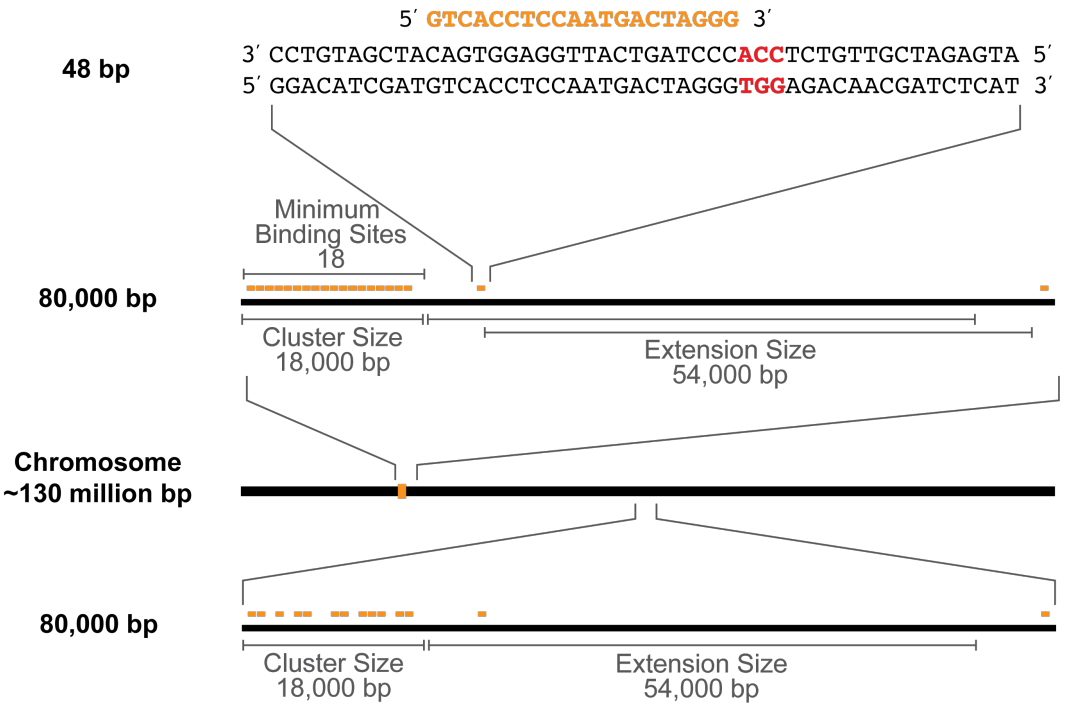


Figure 3.4: **Predicting visible gRNA binding clusters.**
Each possible gRNA sequence was identified based on its position 5' of a PAM site (NGG). A visible cluster was predicted if there are a minimum of eighteen gRNA binding sites within 18 kbp. Additional binding sites within 54 kbp of the first cluster were associated with that cluster. If these minimum criteria were not met, no visible cluster was predicted.

algorithm. Additional comparisons of experimental microscopy data with predicted cluster patterns led us to exclude any gRNAs with fewer than 8,000 binding sites across the entire genome, as these gRNAs were unlikely to produce a visible pattern.

For the purpose of identifying distinguishable patterns for our spatial barcoding strategy, we wanted to increase the likelihood that the collection of gRNAs that we would test could be distinguished from each other. We reasoned that distinguishable sequences likely had repeats on distinct locations on the chromosomes. To filter for these sequences, each sequence's predicted binding pattern was encoded as a single vector by breaking the genome into 100,000 bp sections. Sections with a predicted visible cluster received the value of 1; remaining sections were set to 0. We then performed dimensionality reduction with UMAP on the vectorized representation of the binding patterns.[24] This procedure allowed us to sample gRNAs for testing that were more likely to produce different patterns (Figure 3.5).

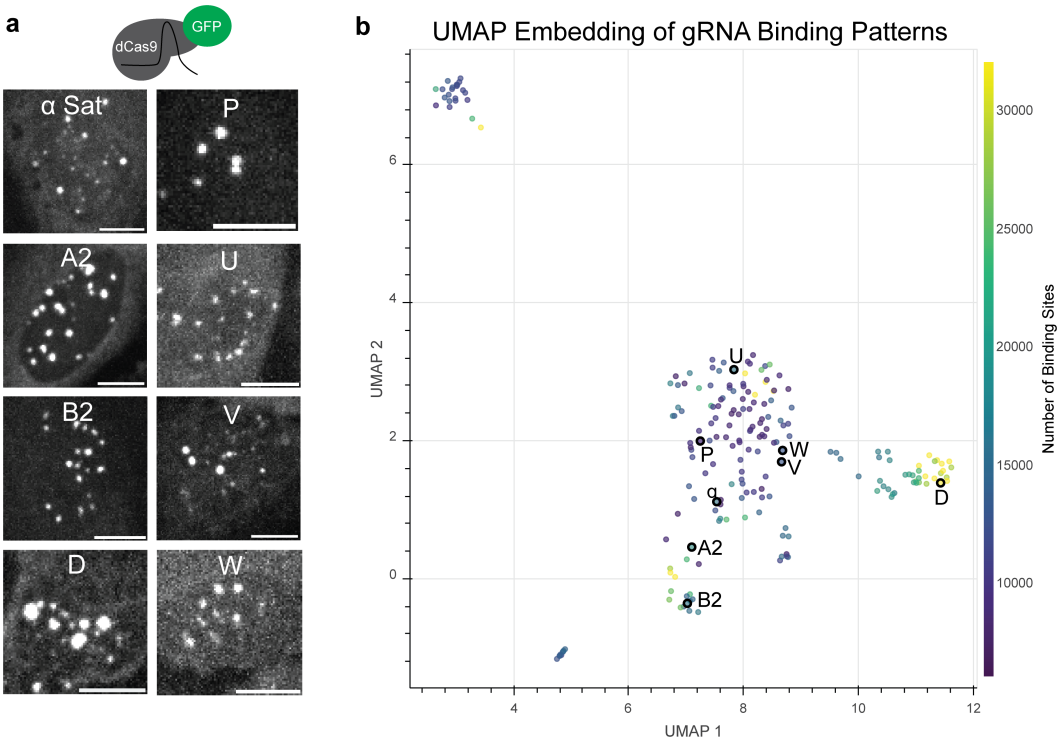


Figure 3.5: **Identifying distinct binding patterns using dimensionality reduction.** A set of gRNA binding patterns in cells (a) was sampled from an embedding space (b) constructed based on the predicted binding patterns of each gRNA. Sample gRNA images were collected from HeLa cells stably expressing dCas9-GFP and transfected with a gRNA expression plasmid. Scale bar, 10 μm.

3.3.2 Screening gRNAs for patterns with CASFISH

After selecting candidate gRNAs with our computational strategy, we then sought to test each gRNA quickly in a cellular context to verify that it produced a visible pattern. To do this, we used CASFISH, a fixed cell staining method that uses an RNA-protein complex consisting of a dCas9 protein complexed with fluorescently labeled gRNA.[25] Together, this complex enables visualization of the labeling pattern resulting from binding of dCas9-gRNA to matching genomic regions in a sequence-specific fashion (Figure 3.6a). This approach has several advantages, which lend itself to high-throughput screening. First, it can utilize any adherent cell line of interest without the need for additional cell line engineering (e.g., introducing dCas9). Second, gRNAs can be ordered from vendors such as IDT with a fast turnaround time and low cost. These gRNAs differ in their target sequence, but contain a sequence that binds to the scaffold contained in the universal fluorescently labeled tracRNA. Finally, since CASFISH is a staining method, it does not rely on exogenous or native expression in the stained cell, eliminating the need to optimize

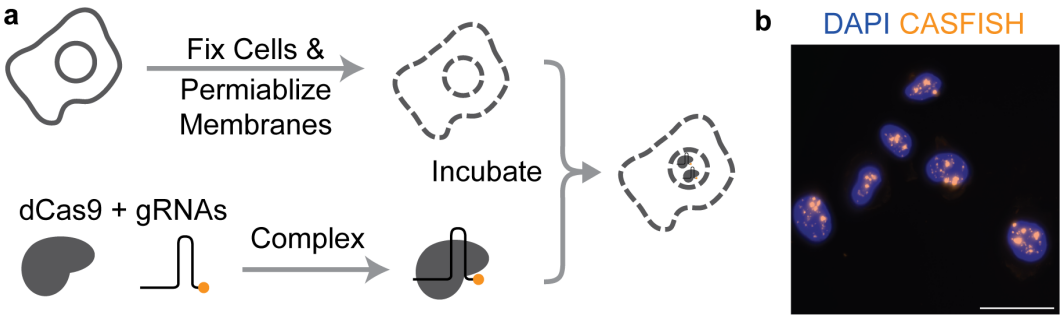


Figure 3.6: **CASFISH enables rapid screening of candidate gRNA sequences.** (a) Cells were fixed and permeabilized prior to incubation with complexes of dCas9 protein with a gRNA. Either dCas9 or the gRNA can be prepared with a fluorescent label in order to facilitate visualization. (b) A sample image of CASFISH staining of HeLa cells with gRNA P demonstrating reliable and consistent staining of every cell. See Section 3.8.5 for details on sample preparation. Scale bar, 50 μm.

the expression of dCas9 and gRNA. Thus, CASFISH can reliably generate labeling in every cell included in the experiment as shown in Figure 3.6b. With this pipeline, we tested 26 candidate gRNA sequences selected with the algorithm described above and identified 21 gRNAs that generated visible labeling patterns (Figure 3.7, Table S3.1).

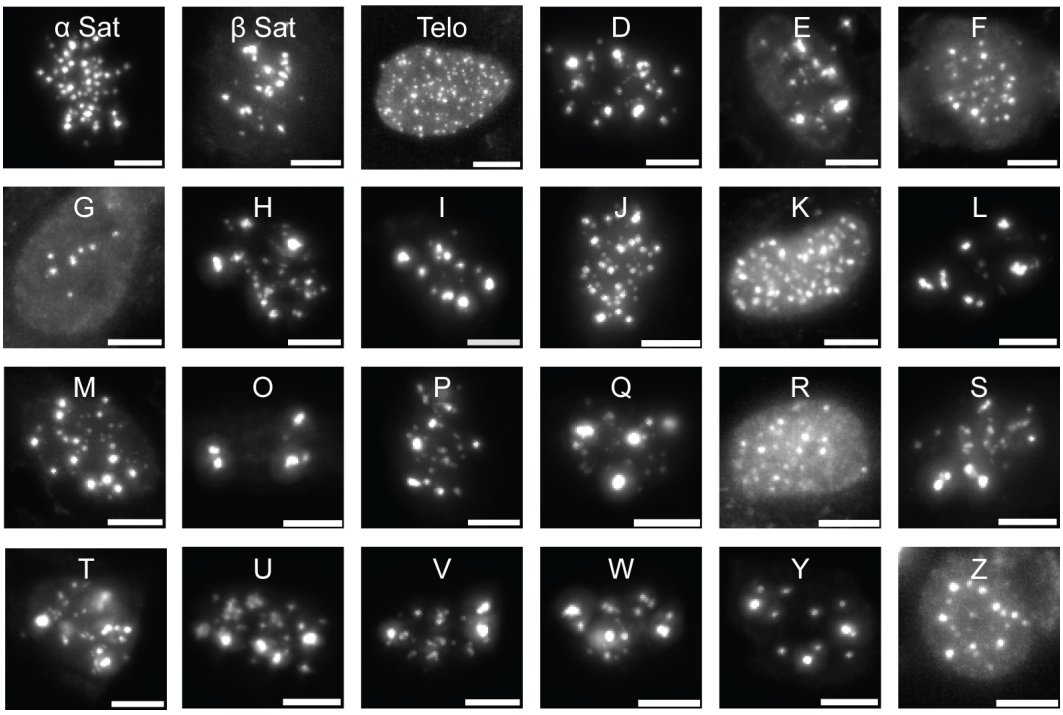


Figure 3.7: **gRNA candidates verified with CASFISH staining.**
21 additional gRNAs were identified and shown to generate visible patterns with CASFISH staining. See Section 3.8.5 for details on sample preparation. For gRNA sequences, see Table S3.1. Scale bar, 10 μm

3.4 Training a deep learning model for barcode identification

In our final barcoding experiments, each cell will be examined and identified on the basis of the gRNA patterns present in the nucleus. To facilitate this process, we will utilize a deep learning model trained to classify each gRNA pattern. To train a preliminary model, we collected a training dataset of CASFISH stained HeLa cells for each of the 24 gRNAs identified in the previous section. The data was processed such that each cell was cropped and resized to a standard size. A preliminary classifier was trained on all 24 gRNAs in the dataset. The performance of the preliminary classifier was highly variable depending on the specific gRNA (Figure S3.1). For example, almost all cells with the telomere gRNA were correctly identified by the model. However, the model frequently confuses gRNA T with gRNAs U and S. To train a model that improved performance for all gRNAs included in the model, we performed an iterative process to eliminate gRNAs that were prone to errors. As a result, we trained a final classifier on 12 gRNAs with the inclusion of a negative class without any gRNA and achieved at least 90% recall on all gRNAs (Figure 3.8). The negative class will allow the model to handle any cells that appear

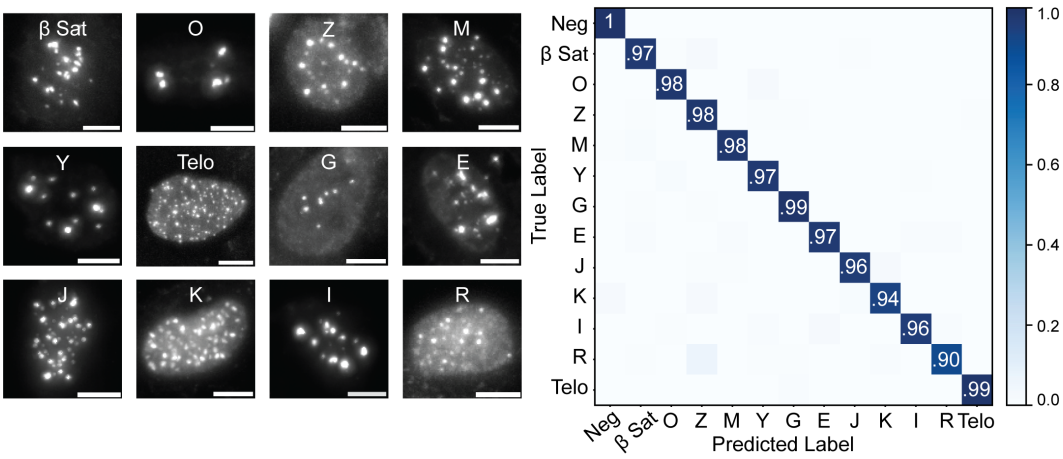


Figure 3.8: **A deep learning classifier can reliably identify gRNA patterns.** The version of the deep learning classifier was trained on 12 gRNA with the addition of a negative gRNA class for unstained cells. The model achieved over 90% recall on all classes. See Section 3.8.9 for details on dataset collection and model training. Scale bar, 10 μm.

in the final experiment without any gRNA present. This result demonstrates that a deep learning model can reliably identify and distinguish different gRNA patterns. Additionally, this success occurs throughout the full spectrum of the cell cycle as the cells used for training were not synchronized to a particular stage in the cell cycle.

In order to test the ability of gRNA patterns to be used as barcodes, and demonstrate our model's ability to correctly assign gRNA patterns to barcoded cells, we used two-color CASFISH to simulate barcoded cells. In this experiment, two gRNAs were selected and each duplexed to tracrRNA ATTO 550 or tracrRNA ATTO 647. To maximize signal sensitivity and subsequent detection of lower frequency binding events, we plan to use a spinning disk confocal microscope for the final implementation of this method, as opposed to the epifluorescent microscope used in Figure 3.8. As part of this eventual transition, we utilized this experiment as an opportunity to test the confocal imaging modality. We captured a pilot dataset of eight gRNAs using the new imaging modality and fine tuned the barcode classifier on this new dataset (Figure 3.9a). The classifier was then used to predict the gRNA pattern in each CASFISH channel in two color CASFISH samples. Cells were considered correctly identified if the model assigned the correct gRNA in both channels. On a test of two barcodes, U|G and O|β, the model correctly identified 91% and 89% of cells respectively (Figure 3.9b,c). These results demonstrate that gRNAs can be combined in a single cell and reliably identified from confocal images by the model as distinct and recognizable patterns.

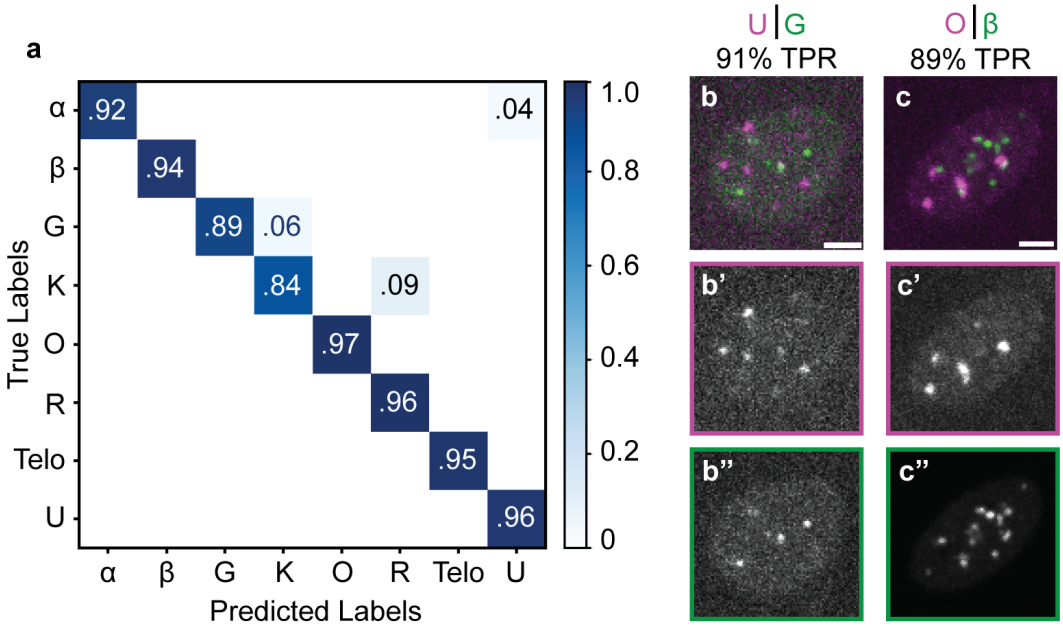


Figure 3.9: **Identifying barcoded cells.**
(a) A deep learning model was trained to identify eight gRNAs and achieved a minimum of 84% recall for each gRNA. The model was tested on cells that had been stained using two color CASFISH where each of the two gRNAs was conjugated to a different fluorescent marker. (b) In cells labeled with gRNAs U and G, the model correctly assigned the U|G barcode to 91% of cells. (c) In cells labeled with gRNAs O and β satellite, the model correctly assigned the O|β barcode to 89% of cells. See Section 3.8.10 for method details. Scale bars, 20 μm.

3.5 Imaging gRNA-induced spatial patterns with FISH

The next challenge in developing the spatial barcoding method was compatibility with live cell imaging. This imposed two constraints: (1) the gRNAs must be expressed in living cells and (2) the gRNAs must contain an accessible, stable, and modular binding site for multicolor RNA FISH. The first constraint is a natural consequence of our desire to perform live-cell imaging on barcoded cells: the barcodes must be present in the cells prior to imaging. The second constraint arises from the need to preserve the fluorescence channels for fluorescent biosensors. Although spatial patterns can be imaged with dCas9 conjugated to a fluorescent protein (Figure 3.5a), this experimental design hinders the use of fluorescent biosensors during live cell imaging. The readout of barcodes with multicolor RNA FISH is a better approach, as the full palette of fluorescent biosensors can be used for cell phenotyping. After live cell imaging, cells can be photobleached, fixed, and stained with multicolor RNA FISH for spatial barcode readout. Note that this approach is also compatible with multiple rounds of multicolor RNA FISH staining and imaging,

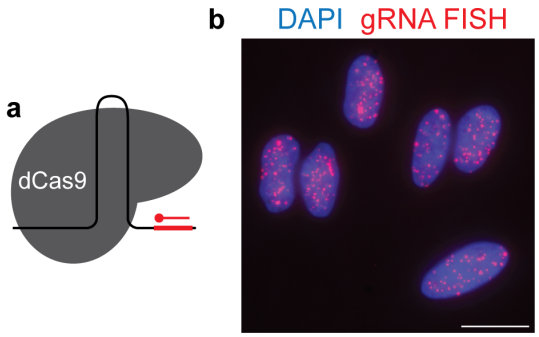


Figure 3.10: **Spatial patterns produced by gRNAs can be labeled using gRNA FISH.**
(a) We used a modified gRNA scaffold that introduced a binding site for a fluorescent oligo probe into the 3' end of the gRNA. (b) CASFISH was performed on HeLa cells with a modified α satellite gRNA prior to FISH staining. Each DAPI-labeled nucleus (blue) has the expected α satellite patterns visible through gRNA FISH (red). See Section 3.8.6 for details on sample preparation. Scale bar, 20 μm.

which can substantially increase our maximum possible barcoded library size: one round of 3-color imaging with 20 spatial patterns allows for a barcode size of 6840, while two rounds allows for a barcode size of more than 40 million.

Our first experiments to solve this challenge focused on developing methodology to image our spatial barcode gRNAs with RNA FISH. Our first proof-of-concept experiment used a modified version of CASFISH[25] to verify that our spatial patterns could be imaged with RNA FISH. For this experiment, we designed a modified gRNA scaffold with a binding site for a fluorescent oligo at the 3' end of the scaffold (Figure 3.10a). This modified gRNA sequence was synthesized by in vitro transcription and complexed with dCas9 proteins according to the standard CASFISH protocol. Following CASFISH staining, we performed secondary FISH labeling with a fluorescent oligo corresponding to the target sequence introduced into the scaffold. Each cell stained with the modified α satellite gRNA displayed patterns of FISH labeling consistent with the pattern previously observed with standard CASFISH (Figure 3.10b). Furthermore, control samples including a nonspecific gRNA with the same target FISH sequence and a negative sample without CASFISH staining did not present any specific FISH staining (data not shown). These experiments demonstrated that gRNA binding patterns can be reliably labeled with fluorescent oligo probes.

Next, we tested our gRNA scaffold design with expression in live cells. Although our final experimental design excludes the use of dCas9-GFP, during this stage of

method development, we found that it served as a useful positive control to verify that the spatial pattern produced by FISH staining corresponds to the pattern induced by gRNAs targeting repetitive sequences. We prepared a plasmid that expresses α satellite FISH gRNA under the U6 promoter and transfected it into HEK293T cells that stably express dCas9-GFP. Although we observed the successful formation of α satellite spots with dCas9-GFP (Figure 3.11b"), we did not observe the corresponding patterns in the images produced by gRNA FISH labeling (Figure 3.11b'). Given the position of the FISH binding site at the exposed 3' end of the gRNA, we suspected that the gRNA may experience 3' degradation when expressed in cells, an event that would lead to the ablation of the FISH binding site. To overcome this possible problem, we explored alternative locations for our FISH binding site, inspired by previous work that tested the efficacy of introducing long noncoding RNAs at various locations in the gRNA scaffold.[16] Based on their success using a modification of the internal hairpin of the gRNA, we designed a new internal FISH scaffold that expanded the hairpin loop to include the binding site of the FISH probe (Figure 3.11d). With this new design, we observed specific FISH labeling that corresponded directly to the spots seen with dCas9-GFP (Figure 3.11e).

Next, we sought to optimize our gRNA FISH protocols to enable high-throughput imaging. Two constraints imposed by the desire for high throughput are (1) low magnification and (2) imaging with an air objective. During our development of the gRNA FISH scaffold design, all imaging was performed with a 100X oil objective to maximize signal sensitivity, violating our two constraints. Decreasing from a 100X to a 40X objective would increase the number of cells per FOV by 7-fold. Similarly, imaging with an air objective would increase sample throughput and ease the difficulty of image acquisition. The trade-off of these two experimental choices is a reduction in the signal. To mitigate this issue, we explored several methods for amplification of the FISH signal (data not shown). These experiments led us to conclude that SABERFISH[26] was the optimal signal amplification strategy for our use case. SABERFISH uses primary probes that contain repeated binding sites for the secondary fluorescent probe (Figure 3.13a) and allows signal amplification with only one additional stage in the staining protocol. Our primary probes consist of a 50 bp recognition sequence that binds to a corresponding target sequence in the gRNA internal FISH scaffold and six binding sites for fluorescent secondary probes. In addition, these primary probes can be ordered as IDT ultramers, eliminating the need for time-intensive probe synthesis protocols. We tested three different primary binding sequences in the internal hairpin site and found that all three produced

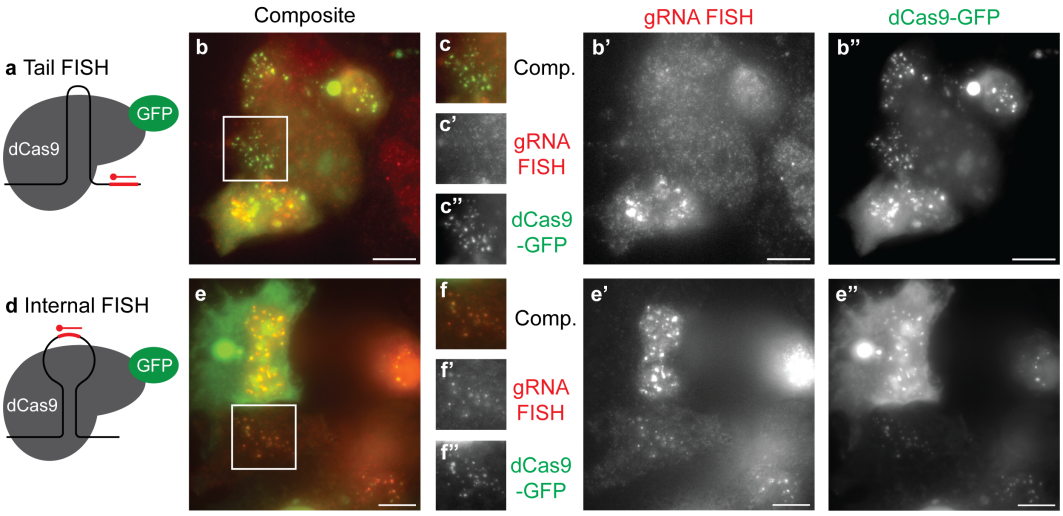


Figure 3.11: **gRNA FISH scaffold designs.**
α satellite gRNA expression plasmids were transfected into HEK293T/17 cells stably expressing dCas9-GFP. (a) The tail scaffold design includes a FISH probe binding site at the 3' end of the gRNA scaffold. Although this gRNA produced α satellite spots that could be visualized with dCas9-GFP (b"), there was no corresponding specific gRNA FISH labeling (b'). For further detail, see inset (c). (d) In contrast, the internal scaffold includes the FISH binding site within the hairpin loop of the gRNA scaffold. This design produced spots visible with both the dCas9-GFP (d") and the corresponding gRNA FISH labeling (d'). See inset (f) for further details. See Section 3.8.7 for details on sample preparation. LUTs are set for signal visibility and should not be compared between images. Scale bar, 10 μm.

successful FISH labeling (Figure 3.12).

To test the SABER gRNA FISH protocol in a context that mirrors the final experimental system, we designed a lentiviral construct for the expression of gRNAs. In addition to the gRNA expressed under the U6 promoter, the construct also expresses H2B-mOrange as a tranduction marker and puromycin resistance for selection. HeLa cells that stably expressed dCas9-GFP were also transduced with the gRNA virus. We performed gRNA SABERFISH on these barcoded cells (Figure 3.13b). Preliminary inspection of the data revealed that every cell expressing H2B-mOrange with visible levels of dCas9-GFP displayed patterns that were also labeled with FISH (Figure 3.13d). The dCas9-GFP virus also included Blasticidin resistance and cells were maintained under Blasticidin selection. As such cells expressing H2B-mOrange contain gRNA FISH patterns even in the absence of visible dCas9-GFP, suggesting that the expression of dCas9-GFP in these cells is too low to visualize, but high enough to effectively create gRNA patterns (Figure 3.13c"'). All imaging was performed with a 40X air objective, indicating that signal amplification achieved

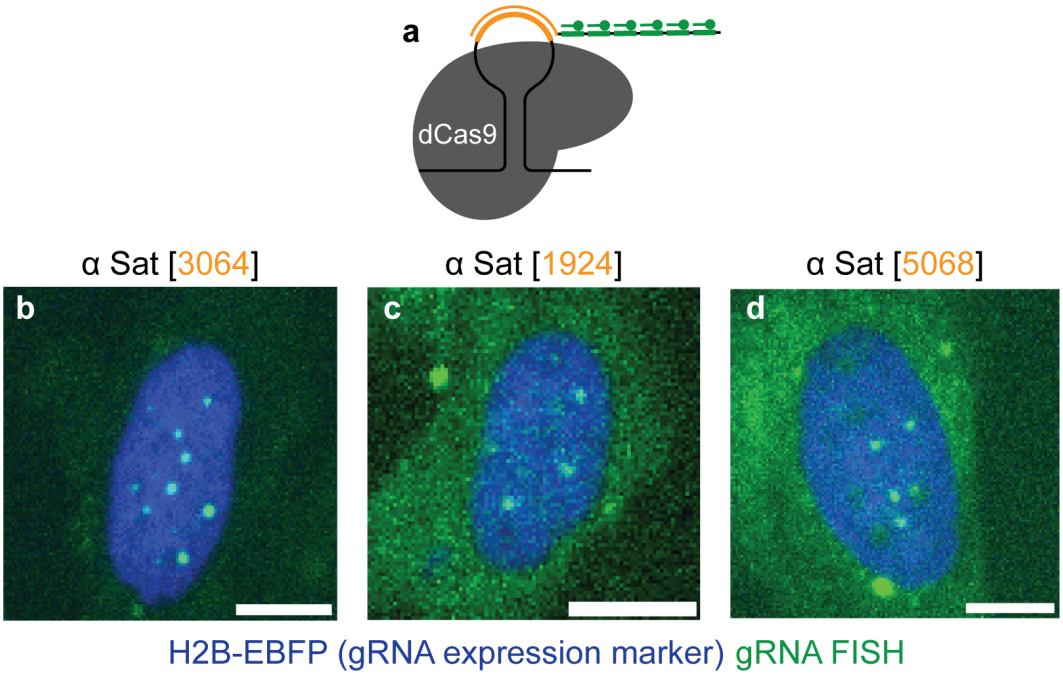


Figure 3.12: **gRNA FISH with different primary binding sequences.**
HeLa cells stable expressing dCas9 were transfected with a plasmid that expresses H2B-EBFP and the α satellite gRNA with an internal FISH binding site. (a) A primary probe was designed to recognize a binding site in the gRNA scaffold (labeled in orange) and contain six binding sites for a fluorescent secondary probe conjugated to Alexa Fluor 488. Three different primary probe binding sequences were tested in the internal hairpin position: (b) 3064, (c) 1924, and (d) 5068 (Table 3.1). All three primary probes enabled FISH labeling of the α satellite pattern. Staining was performed according to the protocol described in Section 3.8.8. Scale bar, 10 μm.

using the SABER design is sufficient for higher-throughput imaging.

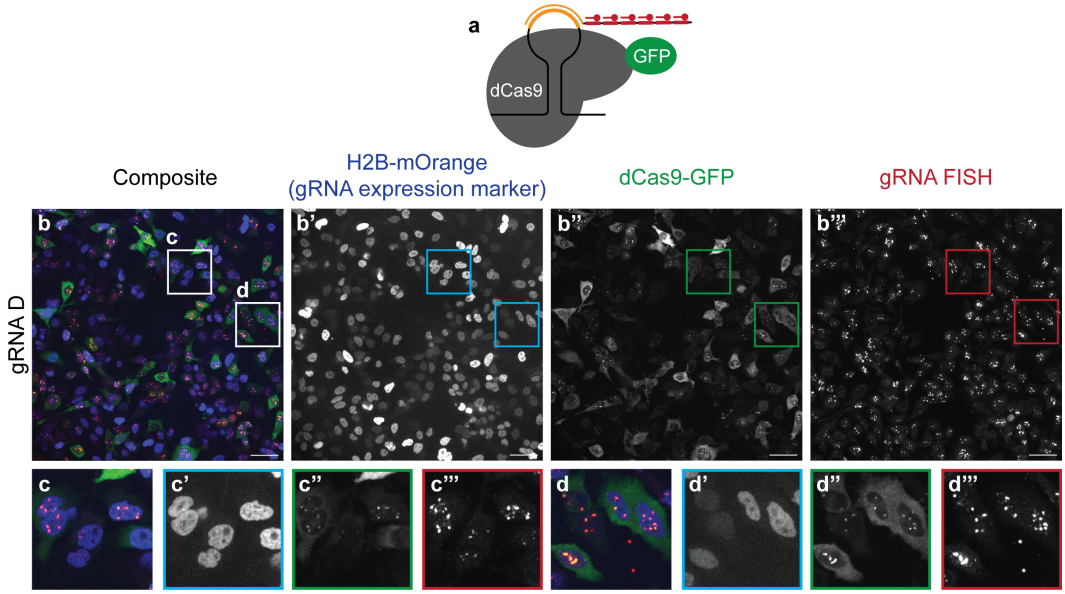


Figure 3.13: **SABER gRNA FISH in stable cells.**
(a) In order to amplify gRNA FISH signal so that we could image using a 40X air objective in contrast to the 100X oil objective used in preliminary experiments, we modified the design of our FISH probe to include an intermediate primary probe with binding sites for six secondary probes conjugated to Alexa Fluor 647. (b) We developed a stable HeLa barcoded cell line using lentiviral transduction. First, cells were transduced with a construct expressing dCas9-GFP. Second, they were transduced with a construct expressing H2B-mOrange as a transduction marker and gRNA D as a barcode. Insets (c) and (d) show that in cells expressing gRNA as indicated by the presence of H2B-mOrange, gRNA FISH staining recapitulates the labeling of dCas9-GFP and additionally labels the gRNA signal that is not visible with the limited signal amplification of dCas9-GFP. See Section 3.8.8 for details on sample preparation. Scale bar, 50 μm.

3.6 Discussion

In this work, we sought to develop a method for performing microscopy-based pooled perturbation screens that is compatible with other multiplexed phenotyping methods. This methodology will enable an extensive interrogation of cell state by collecting a comprehensive microscopy dataset that links multiple phenotypic measurements to perturbations. To this end, we have adapted CRISPR-imaging methodology to create distinct spatial patterns in cells and trained a deep learning model to identify these patterns. During the course of developing this method, several lessons have been learned that may be relevant to the continued development and future applications of this technology.

3.6.1 Method development lessons

gRNA scaffold design

To label gRNAs with several different colors after fixing cells, we modified the gRNA scaffold to include a binding site for a FISH probe. We began by placing the FISH binding site at the 3' end of the gRNA. In our preliminary tests using CASFISH, which introduces the dCas9-gRNA complex after fixation, FISH labeling was successful (Figure 3.10). However, when we expressed a gRNA with the same design in cells expressing dCas9-GFP, we observed the expected spots labeled by dCas9-GFP, but did not observe any corresponding FISH staining of the gRNA (Figure 3.11a–c). These results suggested that, while the gRNA was expressed and functional, the FISH binding site was not available for labeling. Although there are several possible explanations for why this might occur, we first pursued the hypothesis that the FISH binding site was being degraded by 3' RNase activity. As a result, we tested design modifications to protect the FISH binding site from RNase activity. Multiple gRNA designs were tested to physically protect the binding site, either by burying it in the Cas9 protein or by modifying its structure to render it non-targetable by nucleases. Of the two primary methods we tried, we found success in locating the binding site within the central hairpin of the gRNA, and it is this design that we ultimately used in our final design. However, we also tested an alternative design that extended the 3' end of the gRNA following the FISH binding site to include a 3' hairpin, which we reasoned would protect the binding site from degradation. Although we observed the successful formation of spots labeled with dCas9-GFP, FISH staining revealed spots that were not colocalized with dCas9-GFP. Furthermore, in a non-targeting gRNA control, we observed FISH labeled spots in the absence of any dCas9-GFP spots (data not shown). Our experience in designing the modified gRNA scaffold highlights the importance of appropriate controls when designing, testing, and validating pattern-generating methods for labeling cells. In this specific case, we found that the use of a dCas9-based labeling method as a control for the correct binding pattern was essential to ensure that the selected patterns were specific and intended. The second control method we identified was to test non-targeting (e.g., scramble) gRNAs to verify that the modifications are not leading to unintended labeling or spurious detection events.

gRNA delivery method

Following initial proof-of-concept testing using CASFISH in fixed cells, we began to perform experiments in live cells that stably express dCas9-GFP. Our initial experiments utilized plasmid transfection, as this allowed us to rapidly iterate on different designs of gRNA scaffolds. Our first gRNA plasmids expressed only gRNA. Although a successful transfection would be indicated by the formation of spots labeled with dCas9-GFP, experiments that did not generate spots were uninformative, as this negative result could be due to inefficient transfection or gRNA modifications that interfered with its function. To improve the interpretability of our experiments, we modified the design of the gRNA-expressing plasmids to include a second expression cassette for a transfection marker such as H2B-mCherry. With this design, we could use the presence of H2B-mCherry in a cell as a marker of gRNA expression, allowing us to separate the efficiency of transfection from the impact of gRNA design. To collect training data, we optimized our transfection protocol to maximize the number of cells within a FOV that expressed gRNA. However, transfection protocols that would allow a reasonable data collection throughput negatively impacted cell health. Although this was not readily apparent when cells were examined by phase microscopy, when we began FISH staining, we encountered high levels of background signal, likely due to staining of cellular debris (data not shown). In particular, high background levels were not observed in untransfected cells. Ultimately, we elected to use lentiviral vectors to generate stable cell lines expressing dCas9-GFP and gRNA. Although this change added additional overhead to our experiments due to the need to package each new gRNA construct, the overhead was offset by the decreased imaging time as a result of a significant increase in the number of cells expressing the gRNA. In addition, it eliminated the need to perform additional optimization of the FISH protocol to achieve clean staining. Additionally, since most pooled screens are delivered utilizing lentivirus, performing development utilizing lentiviral-based stable lines allows us to more closely mirror the final experimental context in which our barcodes will be used.

3.6.2 Next steps

Recent advances and technologies employing dCas9 for gene manipulation suggest that our barcoding system naturally complements dCas9-based perturbation screens. Multiplexed gRNA arrays have been shown to maintain knockdown and gene editing efficacy compared to single gRNA expression units.[27] Furthermore, multiplexed CRISPR imaging experiments have demonstrated that multiple repetitive sequences

can be targeted simultaneously.[28,29] This prior work suggests that our gRNA barcodes can be directly integrated into CRISPR knockdown screens without impacting the efficacy of the screen. To combine our barcodes with Cas9-based screens that require gene editing, Cas9 orthologs could be used to separate the targeting of barcode gRNAs from knockout gRNAs. Previously, three dCas9 orthologs have been utilized to generate three color CRISPR imaging of three different genomic loci, which supports the feasibility of this approach.[30] Although not yet tested, our method should generalize to the introduction of a barcoded population of cells into tissue. The gRNA FISH staining protocol we developed was derived from SABERFISH[26] which has been used in tissues.

We utilized two-color CASFISH to perform a proof-of-concept experiment that demonstrated our ability to generate multiple discrete patterns within a single cell, image these patterns using confocal microscopy and identify both patterns using a barcode classifier. We tested two sets of gRNA patterns using gRNAs conjugated to ATTO 550 and 647. We achieved 91% and 89% true positive rate on two barcodes, U|G and O|β respectively, in this proof of concept test of our labeling and model performance (Figure 3.9). These results establish that reasonable accuracy of identification, $\tilde{9}0\%$, is possible with our current model backbone and small trial datasets. While a proof of concept accuracy of $\tilde{9}0\%$ is sufficient to generate pooled data sets for analysis, further optimization will enable the collection of data sets with lower false-positive noise. Furthermore, this demonstration of successful identification of patterns by our model when combined with our ability generate stable barcoded cells suggest that the next step of constructing a training dataset in stable cell lines will enable full implementation and usability of the method. However, even after full implementation, I propose that increases in model accuracy, beyond the current $\tilde{9}0\%$ are possible, and could play a key role in increasing the available number of distinguishable gRNAs to further increase library size, or to otherwise reduce the amount of data needed for robust event detection.

In any pooled screen, a certain amount of noise is expected in the data as a result of cells being assigned to the incorrect perturbation. In order to maximize the strength of any screen this type of error needs to be minimized. For our method, we are using a deep learning model to assign cells to their perturbation. A typical deep learning model is trained on a given set of classes and will always assign a prediction to one of its known classes. A drawback of this approach is that even if a model is uncertain about an assignment it will always predict one of the known classes. In

order to overcome this potential limitation, investigations in the deep learning space have been made into methods to quantify model confidence in order to create an option for the model to assign a sample to an unknown class.[31,32] For the purpose of this spatial optical barcoding method, we will apply the Spectral-normalized Neural Gaussian Process (SNGP) technique to the barcode classifier model in order to reduce the likelihood of incorrectly assigning a cell to the wrong perturbation.[33]

Although our preliminary barcode classifier was trained on data collected using CASFISH staining, we have since refined our method to express gRNA in live cells and developed a protocol for gRNA FISH staining. As such, we will collect a new training data set that uses stable populations of cells that express each barcode gRNA and stained with gRNA FISH. This data set will be used to train a new deep learning classifier that can be used to identify barcodes in a unified experimental context. To test its performance on a pool of cells, we will develop two stable cell lines, each expressing H2B conjugated to a different fluorescent protein and a different barcode gRNA. The model's prediction of the gRNA barcode identity can then be validated using the color of the fluorescent protein expressed in that cell.

In order fully demonstrate that our barcodes are an effective method for performing pooled screens, additional validation experiments are required. First, we will verify that the expression of barcode gRNAs does not alter the state of the cell in the absence of any perturbation. Although barcode gRNAs are designed to target noncoding repetitive sequences, there is a theoretical possibility that binding of many dCas9-gRNA complexes to the genome could still impact normal cellular function. To investigate this possibility, we will perform bulk RNA sequencing experiments on cells expressing each of the barcode gRNAs along with a nontargeting gRNA as a negative control. By comparing the gene expression of barcoded cells to cells with a nontargeting control, we will detect any possible perturbations in cell state that may be caused by the binding of barcode gRNA. If some particular barcode gRNAs disrupt the cell state, these gRNAs can be eliminated and replaced with other candidates.

Next, as we would like to use this technology to barcode CRISPR knockdown screens, we will verify that knockdowns can be performed with the same efficacy in the presence and absence of barcode gRNAs. For this experiment, we will design three gRNA expression constructs using validated knockdown gRNA. The constructs will express (1) only the knockdown gRNA, (2) the knockdown gRNA and three barcode gRNAs, and (3) the knockdown gRNA and three nontargeting gRNAs.

Once stable cell lines are generated in a dCas9 background from each construct, we will measure the expression of the gene targeted for knockdown. Previous investigations of multiplexed CRISPR have shown that multiple gRNAs directed to different targets can be effective, although an observed reduction in efficacy between one and many gRNAs is often deemed acceptable.[27] As such, we expect a minor reduction in knockdown efficiency in both conditions that express four gRNAs that is due to the number of gRNAs expressed and not the presence of barcode gRNAs. However, despite potential reductions in knockdown efficiency, existing methods for detecting the number of RNA transcripts are sufficiently precise that even modest knockdown, which might more closely represent physiological perturbation, would be detectable and quantifiable.

Although our initial development of this method has been performed in HEK293 and HeLa cells, which has allowed us to optimize our transfection and labeling protocols given their prevalence in the field, this method should translate to other human cell types. However, since immortalized cell lines are known to have significant genetic abnormalities, we expect that new training data will need to be collected for each new cell line to account for these differences. However, since we have developed this platform using lentiviral delivery, new cell types require only an initial optimization of the transduction protocol before a new library of gRNA-expressing cells can be banked. Finally, while the barcode gRNAs presented here are developed to target the human genome, a similar set of gRNAs could be rapidly developed for other species using the algorithm and testing framework developed here.

3.7 Conclusion

In this chapter, I present the development of key components of an optical barcoding method for use in pooled microscopy screens. This method leverages noncoding repetitive sequences in the genome to generate spatial patterns using CRISPR imaging as a labeling method. In Figure 3.5, we demonstrate that we can computationally identify new gRNAs that produce patterns and have shown that these patterns can be distinguished using a deep learning model to identify the gRNA that generated a particular pattern (Figure 3.8). We have developed stable cell lines expressing dCas9 and a barcode gRNA and demonstrated that barcode gRNAs can be labeled using FISH after introducing a probe-binding site into the scaffold of the gRNA (Figure 3.13). Furthermore, we have tested three different probe-binding sites to ensure that we can label gRNAs with different readout sequences (Figure 3.12). Using these tools, we are now ready to create a pool of barcoded cells by introducing

a combination of three gRNAs, each with a FISH binding site corresponding to a probe of different colors.

We will first generate a library of stable dCas9 cell lines expressing the suite of gRNAs presented in Figure 3.7 and use this library to collect an updated training dataset that utilizes FISH staining of these gRNAs. Our preliminary deep learning model trained on CASFISH data has shown that it can reliably distinguish between individual gRNA patterns Figure 3.8. As such, the deep learning model trained on the FISH-labeled dataset should be able to identify each of the three gRNAs in each cell captured in three different channels and match the combination of patterns and colors to a perturbation. While fine-tuning of a deep learning model to identify patterns across a range of cell types and collection conditions presents some challenges, they are primarily in the form of collecting sufficiently varied training data with which to ensure that the model generalizes well. I have developed pipelines for data ingress and annotation which have supported both the development of the barcode classifier and the work presented in Chapter 2. This infrastructure enables rapid processing of the range training data that may be needed to train a generalizable model that is capable of reliably identifying barcode patterns under an array of different imaging and treatment conditions. With the support of the cell segmentation tools presented in Chapter 2, data collected from a single well in a 96-well plate can provide thousands of single cell training examples.

Using this suite of tools, we propose that with only 20 gRNA patterns, we can encode a library of 6,840 barcodes, sufficient to support a perturbation screen affecting all protein-coding genes, with one round of three color imaging or over 40 million barcodes, more than sufficient to cover the entire transcriptome with a second round of imaging. Further, by requiring only one round, or at most two rounds for extensive coverage, of imaging to read our barcodes, our method prioritizes compatibility with optical phenotyping methods including multiplexed methods that require many rounds of imaging. As such, this methodology supports the collection of a multimodal dataset described in Chapter 4 by pairing combinations of barcoded live cell reporters with endpoint measurement of gene expression.

3.8 Materials and methods

3.8.1 Cell culture

HEK293T/17 cells (ATCC CRL-11268) were cultured in Dulbecco's modified Eagle's medium (DMEM; Caisson DML10) supplemented with 10% fetal bovine

serum (Thermo Fisher #26140079) and 100 U/mL penicillin / streptomycin (Caisson PSL01). HeLa cells (ATCC CCL-2) were cultured in Minimal Essential Media (EMEM; Cytiva SH30024.FS) supplemented with 10% fetal bovine serum (Thermo Fisher #26140079) with 100 U/mL penicillin/streptomycin (Caisson PSL01).

For experiments performed in a dCas9 background, HeLa cells (ATCC CCL-2) were transduced with either pGH125_dCas9-Blast (Addgene #61425[34]) or pLenti-EF1a-SPdCas9-EGFP-2A-Blast (Addgene #71215[35]). Details of the transduction protocal are described in the following section.

3.8.2 Lentiviral packaging and transduction

Lentivirus was packaged using third generation transfer plasmids and second generation helper plasmids. Transfer plasmids were prepared in the pLex_305 (Addgene #41390) or pTwist Lenti SFFV (Twist Biosciences) backbone. HEK293T/17 cells (ATCC CRL-11268) were seeded in 10 cm plates (3.8×10^6 in 10 ml of complete DMEM). After 24 hours, cells were transfected with psPax2 (9 μg; Addgene #12260), pMD2.G (0.9 μg; Addgene #12259) and transfer plasmid (9 μg) using TransIT-LT1 (54 μl; Mirus #2300) in a total volume of 315 μl of OptiMem (Invitrogen, #31985-070). 18 hours after transfection, the medium was replaced with 15 ml of high-BSA (1 mg/ml; Caisson B0005) complete DMEM. The virus was harvested 24 hours after the media change and any remaining cells were removed with a 0.45 μm PVDF syringe filter (Celltreat #229745). The virus was concentrated using the Lenti-X Concentrator Kit (Takara #631232).

Additional virus was packaged by the viral tools team at Janelia Research Campus.

Cells were transduced by adding concentrated virus and polybrene (10 μg/ml; Santa Cruz Biotechnology sc-134220) and centrifuging at 800 g for 1 hour. After two days of recovery, cells were transferred to medium containing the appropriate antibiotic selection agent at the following concentrations: 16 μg/ml Blasticidin (Santa Cruz Biotechnology sc-204655A), 2 μg/ml puromycin (PeproTech #5855822).

3.8.3 gRNA sequence design

The human genome was analyzed using the sequencing data from the Telomere-to-Telomere (T2T) Consortium.[20] First, the genome was searched for "NGG" PAM sites by looking for the sequence "GG" or its reverse complement "CC". The 20 upstream base pairs were then compared with the sequences of existing gRNA sequences to count the number of binding sites for each gRNA. Biopython was used

to extract sequence data from genome files and to create reverse complements of PAM site sequences, as well as gRNA probe sequences.[36]

To determine which types of binding patterns would produce fluorescent spots, we generated a set of binding parameters that could correlate our computational results with our CASFISH data. Using data from the alpha satellite, beta satellite, and telomere gRNA sequences, we found a set of parameters that best linked the experimental data for fluorescent spots for CASFISH experiments imaged at 100x to the number of spots that the computational data predicted. The threshold was that there must be at least eighteen binding sites within a region of 18,000 bp, with an extension size of 54,000 bp, which meant that any binding site after the first eighteen that was found within the 54,000 bp region of the last binding site would be included in the same fluorescent spot as the first eighteen binding sites.

New gRNA sequences were identified using these clustering parameters (eighteen binding sites, 18,000 bp region, 54,000 bp extension size). We used the same algorithm for identifying new PAM sites, except that now for each PAM site, the 20 basepair region upstream was either added to the binding sites for that gRNA if it already existed or stored as a new gRNA sequence. Then, at the end of each chromosome, all gRNA sequences that did not meet the thresholds for creating a fluorescent spot/cluster were dropped from the list. This process was repeated for each chromosome.

To try to computationally predict differences in gRNA binding patterns, each chromosome was divided into 100,000 bp chunks and the above process was repeated for filtering gRNA sequences. Then, for these new gRNA sequences, an array was created in which rows represented probe sequences, and columns represented individual chunks. A value of 0 indicated that that probe did not have a fluorescent cluster in that fragment, while a value of 1 indicated the existence of a fluorescent cluster in that fragment for that gRNA probe. Based on previous experimental observations, we dropped all probes that had fewer than 8000 binding sites. We then used UMAP to cluster the vectorized binding data to try to distinguish between gRNA sequences that had unique binding patterns.[24]

For a complete set of all gRNA sequences used in this work, see Table S3.1.

3.8.4 gRNA transfections

Cells (10^4 cells in 100 μl) were seeded on 96-well glass plates coated with fibronectin and incubated overnight. For each well, plasmid DNA (100 ng) was diluted in 10

μl of Opti-MEM prior to adding TransIT-LT1 (Mirus, 0.2 μl) and incubated for 30 minutes at RT prior to adding to cells. Cells were imaged 24 hours after transfection.

3.8.5 CASFISH

Cells were seeded on 96-well glass plates coated with fibronectin and incubated overnight. The CASFISH staining protocol was adapted from [25]. Cells were fixed in a pre-chilled solution of methanol and acetic acid in a 1:1 ratio for 5 minutes at -20°C and then rinsed three times in 1×PBS. The cells were then incubated in reaction buffer [Hepes (20 mM, pH 7.5), KCl (100 mM), $MgCl_2$ (5 mM) with freshly added DTT (1 mM), glycerol (5% v/v), and TWEEN-20 (0.1% v/v)]. The gRNA duplex was prepared by combining fluorescently labeled tracRNA (IDT, 100 μM in IDT Duplex Buffer) and crRNA (IDT, 100 μM in IDT Duplex Buffer) in equimolar ratios, heating at 95°C for 5 minutes and then cooling to room temperature. CASFISH probes were prepared by combining dCas9 protein (IDT, 25 nM) and gRNA duplex (25 nM) in reaction buffer and incubating at RT for 10 minutes prior to storage on ice. Cells were stained with CASFISH probes for 5 minutes at 37°C, washed 3 times with 1×PBS, stained with DAPI (300 nM) for 5 minutes and finally washed twice with 1×PBS.

CASFISH samples were imaged with a Nikon Ti2-E fluorescence microscope controlled by Nikon Elements. Images were acquired with a Nikon SOLA SE II light source, a Photometrics Prime 95B CMOS camera, and a 40X oil objective.

3.8.6 CASFISH with FISH

gRNAs with a FISH binding site were synthesized using in vitro transcription as follows. The DNA template was prepared using Q5 polymerase with a forward primer including a T7 promoter and a reverse primer binding to the end of the gRNA scaffold. The resulting PCR product was purified using a Qiagen MinElute column. gRNAs were synthesized using a NEB HiScribe T7 kit and purified using the NEB Monarch RNA Cleanup kit. CASFISH probes were prepared by combining dCas9 protein (IDT, 25 nM) and IVT gRNA (25 nM) in reaction buffer and incubating at RT for 10 minutes before storage on ice. Cells were prepared for staining following the standard CASFISH protocol described above. Cells were stained with CASFISH probes for 5 minutes at 37°C, washed 3 times with 1×PBS, stained with DAPI (300 nM) for 5 minutes, and finally washed twice with 1×PBS. Before secondary staining, cells were fixed in 1×PBS with 4% (w/v) paraformaldehyde for 10 minutes, rinsed twice in 1×PBS, and rinsed twice in 2×SCC. The secondary probe sequences

and the subsequent staining protocol were adapted from [13]. Fluorescent probes (50 nM) were diluted in secondary staining buffer [6.67×SSC, ethylene carbonate (16% w/v), dextran sulfate (0.16 g/ml)] and then incubated with cells in the dark for 20 minutes at RT. Cells were rinsed twice and washed once (5 min) in wash buffer [2×SCC, formamide (10% v/v), Triton-X 100 (0.1% v/v)]. Before imaging, antibleaching buffer was added to each well [Tris-HCl (50 mM, pH 8.0), NaCl (300 mM), trolox (3 mM), D-glucose (8%), catalase (1:100 dilution), glucose oxidase (0.5 mg/ml)].

CASFISH samples with FISH staining were imaged with a Nikon Ti2-E fluorescence microscope controlled by Nikon Elements. Images were acquired with a Nikon SOLA SE II light source, a Photometrics Prime 95B CMOS camera, and a 100X oil objective.

3.8.7 Direct gRNA FISH

Cells were seeded on 96-well glass plates coated with fibronectin. Samples were rinsed in 1× PBS, fixed in 1×PBS with 4% (w/v) paraformaldehyde for 10 minutes and then rinsed twice in 1×PBS. Cells were permeabilized by washing three times for five minutes each in 1×PBS with 0.1% (v/v) Triton X-100. Before secondary staining, cells were rinsed once in 1×PBS and twice in 2×SCC. Secondary probe sequences and the subsequent staining protocol were adapted from [13]. Fluorescent probes (100 nM) were diluted in secondary staining buffer [6.67×SSC, ethylene carbonate (16% w/v), dextran sulfate (0.16 g/ml)] and then incubated with cells in the dark for 30 minutes at RT. Cells were rinsed twice and washed once (5 min) in wash buffer [2×SCC, formamide (10% v/v), Triton-X 100 (0.1% v/v)]. Before imaging, antibleaching buffer was added to each well [Tris-HCl (50 mM, pH 8.0), NaCl (300 mM), trolox (3 mM), D-glucose (8%), catalase (1:100 dilution), glucose oxidase (0.5 mg/ml)].

gRNA FISH samples were imaged with a Nikon Ti2-E fluorescence microscope controlled by Nikon Elements. Images were acquired with a Nikon SOLA SE II light source, a Photometrics Prime 95B CMOS camera, and a 100X oil objective.

3.8.8 SABER gRNA FISH

The target sequences for the primary probes were selected from a collection of 25-mer DNA barcode sequences that were designed to be orthogonal.[37] Each target sequence is a concatenation of two 25-mer barcode sequences (Table 3.1). The remainder of the primary probe consists of six repeated binding sites for the 22 bp

Table 3.1: **gRNA FISH primary probe sequences.**

Name	gRNA Target Sequence
3064	ATCGGGGTGGAAATTATGGCGAATAAACGCGCCTTAAGATACTGCGTAATCT
1924	AGCTCGATACATACTACTGGGTCGGTAGAATTGAGTACGTGCGCAAAGGAA
5658	tAAAAGTCGGCCAATGTGCTATGGGTGTCGTGGAGTTGAAATCGTTGGATA

Table 3.2: **Fluorescent secondary probe sequences.**

Name	Sequence
sf25*647	/5Alex647N/TTTATTATTGGTTATTATTGGT/3InvdT/
sf26*564	/5Alex546N/TTTAGGTTTATTTAGGTTTATT/3InvdT/
sf27*488	/5Alex488N/TTATGATGATGTATGATGATGT/3InvdT/

fluorescent secondary probe, each separated by an "A" nucleotide. Primary probes were ordered from IDT as single-stranded DNA ultramers. The sequences for the secondary probes were drawn from [26] (Table 3.2). Secondary probes were ordered from IDT as oligos conjugated to Alexa Fluor secondaries and purified with HPLC.

Glass coverslips were prepared with a CultureWell silicone gasket to create two 9mm circular chambers or an Ibidi silicone eight-well chamber. Chambers were coated with fibronectin prior to seeding cells and incubating overnight. Samples were rinsed in 1×PBS, fixed in 1×PBS with 4% (w/v) paraformaldehyde for 10 minutes and then rinsed twice in 1×PBS. The silicon gaskets were removed from the coverslip and cells were permeabilized overnight by fully immersing the coverslip in 70% EtOH at -20°C. All subsequent washes were performed in a 200 ml beaker with coverslips held in a slide wash rack. Cells were also permeabilized in 1×PBS with 0.5% (v/v) Triton X-100 for 15 minutes, rinsed once in 1×PBS, washed for 2 minutes in 1x PBS and washed for 2 minutes in 2×SSC with 0.1% (v/v) Tween-20 (2×SSCT). For primary staining, samples were incubated overnight at 37°C in 30% (v/v) formamide, 10% (w/v) dextran sulfate, 0.1% (v/v) Tween-20, 2×SSC and 100 μm primary probe. After overnight incubation, samples were washed twice for 10 minutes in prewarmed 2×SSCT (at 60°C) and washed once for 5 minutes in room temperature 2×SSCT. Before secondary staining, the samples were washed twice in 1× PBS for 2 minutes. Samples were incubated at 37°C for 1 hour with 1 μm fluorescent secondary probes in 1×PBS. After incubation, samples were washed twice for 5 minutes in prewarmed 1×PBS (at 37°C).

SABER gRNA FISH samples were imaged with a Nikon Ti2-E with Crest X-Light V3 spinning disk confocal microscope controlled by Nikon Elements. Images were acquired with a Lumencor Celesta light source, a Photometrics Prime 95B CMOS

camera, and a 40X air objective.

3.8.9 Deep learning barcode classifier

For the collection of training data, samples were stained according to the CASFISH protocol described above in Section 3.8.5. Prior to imaging, the bottom of the glass plate was coated with a thin layer of immersion oil. Imaging was performed with a Nikon Ti2-E fluorescence microscope controlled by Nikon Elements. Images were acquired with a Nikon SOLA SE II light source, a Photometrics Prime 95B CMOS camera, and a 40X oil objective. Automated jobs were configured to collect 24 fields of view from random positions within each well.

Nuclear segmentation was performed on the DAPI channel using the model published in [38]. Using the CASFISH channel, each cell was cropped according to a bounding box around the nuclear segmentation label and resized to 128×128. Before training, the data were manually curated to remove any samples with abnormal segmentation results or other aberrations.

Model training

The barcode classifier model was constructed using a DenseNet121 backbone as a foundation with weights pre-trained on ImageNet.[39] The output of the backbone was fed into a classification head consisting of two additional dense layers to predict the final classification. Before training, images were normalized by subtracting the average pixel value and dividing by the standard deviation of pixel values. Training data were augmented with random rotations, crops, flips, and scaling to improve the diversity of the data. We used 70% of the data for training, 20% for validation, and 10% for testing. The model was trained using the Adam optimizer with a learning rate of 10^{-4}, a clipnorm of 10^{-3}, and a batch size of sixteen images; training was performed for fifty epochs.[40] After each epoch, the learning rate was adjusted using the function $\mathrm{lr} = \mathrm{lr} \times 0.99^{\mathrm{epoch}}$.

3.8.10 Two-color CASFISH

For the collection of training data, samples were stained according to the CASFISH protocol described above in Section 3.8.5. Two tracrRNAs were used for the collection of training data conjugated to either ATTO 550 or ATTO 647. Barcoded cell data was collected by first separately complexing two gRNAs duplexed to either tracrRNA ATTO 550 or ATTO 647 to dCas9. The two complexing reactions were combined prior to staining the sample. Samples were imaged with a Nikon Ti2-E

with Crest X-Light V3 spinning disk confocal microscope controlled by Nikon Elements. Images were acquired with a Lumencor Celesta light source, a Photometrics Prime 95B CMOS camera, and a 40X air objective.

Nuclear segmentation was performed on the DAPI channel using CellSAM.[41] Each cell was cropped according to a bounding box around the nuclear segmentation label and resized to 128×128. Both the CASFISH channel and the DAPI channel were used as inputs to the model.

Model training was performed as described in Section 3.8.9.

For barcoded cells, predictions were performed separately on each CASFISH channel. Cells were counted as correctly identified if both gRNA predictions were made correctly.

3.8.11 Python packages

The following Python packages were used in the course of this work: TensorFlow,[42] NumPy,[43] scikit-learn,[44] scikit-image,[45] pandas,[46] napari,[47] nd2,[48] Biopython,[36] CellSAM,[41] UMAP,[24] seaborn[49] and Matplotlib.[50]

3.9 Supplemental materials

Table S3.1: **Candidate gRNA sequences verified with CASFISH.**

Name	Sequence	Name	Sequence
α Satellite	GAATCTGCAAGTGGATATT	M	TGGATATTTGGACCTCTTTG
β Satellite	AGACAAGAGTTACATCACCT	O	AATGGAATCAACGCGAGTGC
Telomere	TAGGGTTAGGGTTAGGGTTA	P	CCGAGTGCAGGGGAATGGAA
D	TGGAATGGAGTGGAATGGAA	Q	TGGAATGGAATGCAATGGAA
E	CCGAGTGGAATGGCATGGAA	R	TTTGAGGATTTCGTTGGAAA
F	TTCAAACCTGCTCTATGAAA	S	TGGATTGGAATGGAATGGAA
G	TCAGAGGAATAGAAAGGGAC	T	TGGAATGGAATCGAATGGAA
H	TGGAATGGAATGGAATGAAA	U	TGGAATAGAATGGAATGGAA
I	CGAATGGAATCATCATCGAA	V	TGGAATGGAATGGAGTGGAA
J	TTGAGGCCTTCGTTGGAAAC	W	TGGAATGGAATGGAATGGAG
K	TTGAGGATTTCGTTGGAAAC	Y	TTTGAGGTCAATGGTAGAAT
L	AATCGAATGGAATCATCGAA	Z	GACTTGAAACACTCTTTTTG

The gRNA sequences for α satellite and telomere were drawn from the literature. The β satellite sequence was designed based on the satellite repeat sequence published in [23]. The remaining gRNAs were candidates selected by the algorithm presented in this work and were experimentally validated with CASFISH.

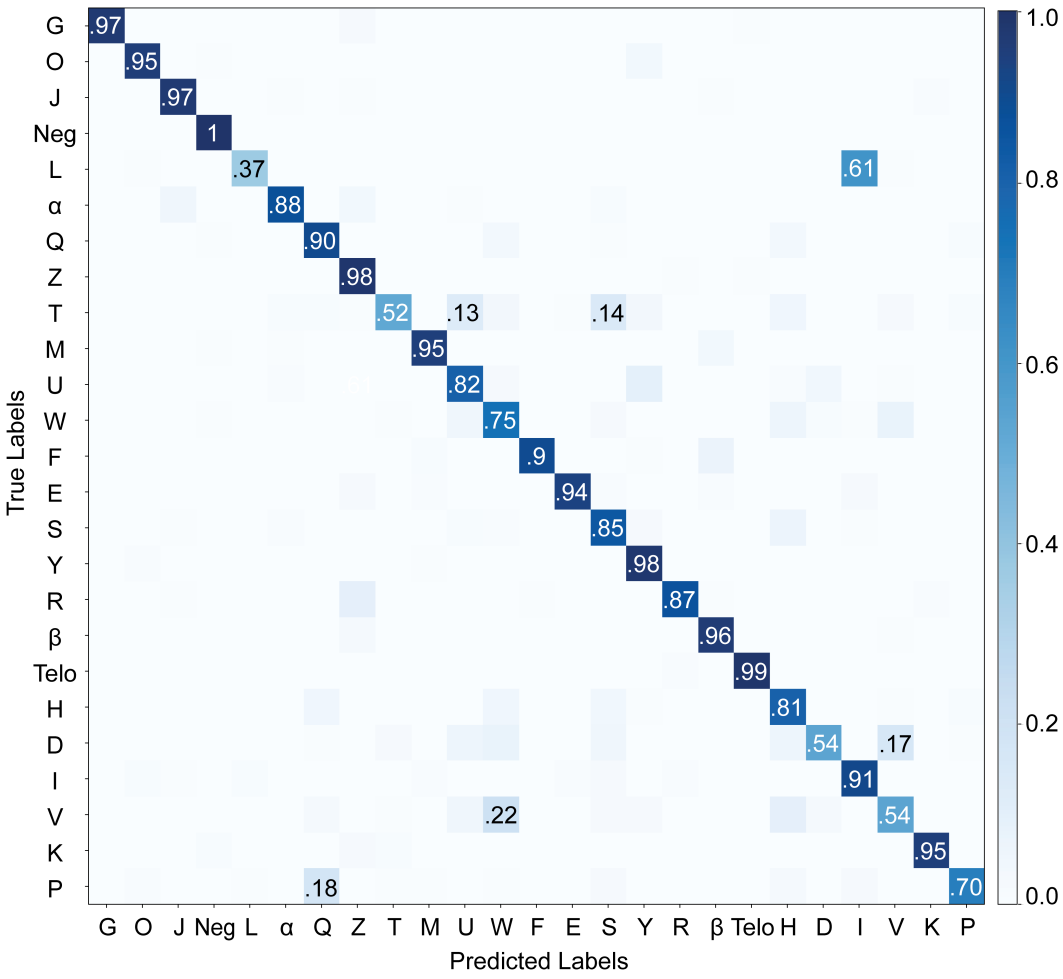


Figure S3.1: **A preliminary deep learning classifier trained on 24 gRNAs.** A preliminary deep learning model was trained on all 24 gRNAs identified using CASFISH screening. This model demonstrated highly variable performance across different gRNAs ranging from nearly perfect recall for the telomere gRNA to as low as 37% recall for gRNA L.

References

1. Emanuel, G., Moffitt, J. R. & Zhuang, X. High-throughput, image-based screening of pooled genetic-variant libraries. *Nature Methods* **14,** 1159–1162 (2017).

2. Wang, C., Lu, T., Emanuel, G., Babcock, H. P. & Zhuang, X. Imaging-based pooled CRISPR screening reveals regulators of lncRNA localization. *Proceedings of the National Academy of Sciences* **116,** 10842–10851 (2019).

3. Feldman, D. *et al.* Optical pooled screens in human cells. *Cell* **179,** 787–799 (2019).

4. Wroblewska, A. *et al.* Protein barcodes enable high-dimensional single-cell CRISPR screens. *Cell* **175,** 1141–1155 (2018).

5. Rovira-Clavé, X. *et al.* Spatial epitope barcoding reveals clonal tumor patch behaviors. *Cancer Cell* **40,** 1423–1439 (2022).

6. Dhainaut, M. *et al.* Spatial CRISPR genomics identifies regulators of the tumor microenvironment. *Cell* **185,** 1223–1239 (2022).

7. Lawson, M. & Elf, J. Imaging-based screens of pool-synthesized cell libraries. *Nature Methods* **18.** review, 358–365 (2021).

8. Gu, J. *et al.* CRISPRmap: Sequencing-free optical pooled screens mapping multi-omic phenotypes in cells and tissue. *bioRxiv,* 2023–12 (2023).

9. Chen, R. *et al.* A barcoding strategy enabling higher-throughput library screening by microscopy. *ACS Synthetic Biology* **4,** 1205–1216 (2015).

10. Yang, J.-M. *et al.* Deciphering cell signaling networks with massively multiplexed biosensor barcoding. *Cell* **184,** 6193–6206 (2021).

11. Kaufman, T. *et al.* Visual barcodes for clonal-multiplexing of live microscopy-based assays. *Nature Communications* **13,** 2725 (2022).

12. Kudo, T., Lane, K. & Covert, M. W. A multiplexed epitope barcoding strategy that enables dynamic cellular phenotypic screens. *Cell Systems* **13,** 376–387 (2022).

13. Eng, C.-H. L. *et al.* Transcriptome-scale super-resolved imaging in tissues by RNA seqFISH+. *Nature* **568,** 235–239 (2019).

14. Radtke, A. J. *et al.* IBEX: A versatile multiplex optical imaging approach for deep phenotyping and spatial analysis of cells in complex tissues. *Proceedings of the National Academy of Sciences* **117,** 33455–33465 (2020).

15. Black, S. *et al.* CODEX multiplexed tissue imaging with DNA-conjugated antibodies. *Nature Protocols* **16,** 3802–3835 (2021).

16. Shechner, D. M., Hacisuleyman, E., Younger, S. T. & Rinn, J. L. Multiplexable, locus-specific targeting of long RNAs with CRISPR-Display. *Nature Methods* **12,** 664–670 (2015).

17. Hong, Y., Lu, G., Duan, J., Liu, W. & Zhang, Y. Comparison and optimization of CRISPR/dCas9/gRNA genome-labeling systems for live cell imaging. *Genome Biology* **19,** 1–10 (2018).

18. Chen, B., Guan, J. & Huang, B. Imaging specific genomic DNA in living cells (2016).

19. Schneider, V. A. *et al.* Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome Research* **27,** 849–864 (2017).

20. Nurk, S. *et al.* The complete sequence of a human genome. *Science* **376,** 44–53 (2022).

21. Shao, S. *et al.* Long-term dual-color tracking of genomic loci by modified sgRNAs of the CRISPR/Cas9 system. *Nucleic Acids Research* **44,** e86–e86 (2016).

22. Chen, B. *et al.* Dynamic imaging of genomic loci in living human cells by an optimized CRISPR/Cas system. *Cell* **155,** 1479–1491 (2013).

23. Waye, J. S. & Willard, H. F. Human beta satellite DNA: genomic organization and sequence definition of a class of highly repetitive tandem DNA. *Proceedings of the National Academy of Sciences* **86,** 6250–6254 (1989).

24. McInnes, L., Healy, J. & Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv.* eprint: 1802.03426 (2018).

25. Deng, W., Shi, X., Tjian, R., Lionnet, T. & Singer, R. H. CASFISH: CRISPR/Cas9-mediated in situ labeling of genomic loci in fixed cells. *Proceedings of the National Academy of Sciences* **112,** 11870–11875 (2015).

26. Kishi, J. Y. *et al.* SABER amplifies FISH: enhanced multiplexed imaging of RNA and DNA in cells and tissues. *Nature Methods* **16,** 533–544 (2019).

27. McCarty, N. S., Graham, A. E., Studená, L. & Ledesma-Amaro, R. Multiplexed CRISPR technologies for gene editing and transcriptional regulation. *Nature Communications* **11,** 1281 (2020).

28. Ma, H. *et al.* Multiplexed labeling of genomic loci with dCas9 and engineered sgRNAs using CRISPRainbow. *Nature Biotechnology* **34,** 528–530 (2016).

29. Fu, Y. *et al.* CRISPR-dCas9 and sgRNA scaffolds enable dual-colour live imaging of satellite sequences and repeat-enriched individual loci. *Nature Communications* **7** (2016).

30. Ma, H. *et al.* Multicolor CRISPR labeling of chromosomal loci in human cells. *Proceedings of the National Academy of Sciences* **112,** 3002–3007 (2015).

31. Abdar, M. *et al.* A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion* **76,** 243–297. ISSN: 1566-2535. http://dx.doi.org/10.1016/j.inffus.2021.05.008 (Dec. 2021).

32. Gawlikowski, J. *et al. A Survey of Uncertainty in Deep Neural Networks* 2022. arXiv: 2107.03342 [cs.LG].

33. Liu, J. Z. *et al. Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness* 2020. arXiv: 2006.10108 [cs.LG].

34. Hess, G. T. *et al.* Directed evolution using dCas9-targeted somatic hypermutation in mammalian cells. *Nature Methods* **13,** 1036–1042 (2016).

35. Lin, L. *et al.* Genome-wide determination of on-target and off-target characteristics for RNA-guided DNA methylation by dCas9 methyltransferases. *Gigascience* **7,** giy011 (2018).

36. Cock, P. J. *et al.* Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics* **25,** 1422 (2009).

37. Xu, Q., Schlabach, M. R., Hannon, G. J. & Elledge, S. J. Design of 240,000 orthogonal 25mer DNA barcode probes. *Proceedings of the National Academy of Sciences* **106,** 2289–2294 (2009).

38. Schwartz, M. S. *et al.* Caliban: Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning. *bioRxiv* (2023).

39. Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. *Densely connected convolutional networks* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 4700–4708.

40. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv.* eprint: 1412.6980 (2014).

41. Israel, U. *et al.* A Foundation Model for Cell Segmentation. *bioRxiv* (2023).

42. Martín Abadi *et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* Software available from tensorflow.org. 2015. https://www.tensorflow.org/.

43. Harris, C. R. *et al.* Array programming with NumPy. *Nature* **585,** 357–362. https://doi.org/10.1038/s41586-020-2649-2 (Sept. 2020).

44. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011).

45. Van der Walt, S. *et al.* scikit-image: image processing in Python. *PeerJ* **2,** e453. ISSN: 2167-8359. https://doi.org/10.7717/peerj.453 (June 2014).

46. Pandas development team, T. *pandas-dev/pandas: Pandas* version latest. Feb. 2020. https://doi.org/10.5281/zenodo.3509134.

47. Ahlers, J. *et al. napari: a multi-dimensional image viewer for Python* https://github.com/napari/napari.

48. Lambert, Talley. *nd2: Full-featured nd2 (Nikon NIS Elements) file reader for python* https://github.com/tlambert03/nd2.

49. Waskom, M. L. seaborn: statistical data visualization. *Journal of Open Source Software* **6,** 3021. https://doi.org/10.21105/joss.03021 (2021).

50. Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* **9,** 90–95 (2007).

Chapter 4

EXPLORING NETWORK RELATIONSHIPS WITH A BARCODED COMBINATORIAL REPORTER LIBRARY

4.1 Introduction

As discussed in Section 1.1, the dynamics of signaling pathways are an important dimension that cells use to encode information about stimuli and trigger different downstream responses. In order to fully decode complex signaling networks, three types of measurements are required. The first is single-cell measurements as many pathways demonstrate heterogeneous responses when comparing between individual cells in a population.[1–5] The second is to combine measurements of multiple pathways in a single cell to explore the dependencies between different pathways.[6,7] The third is multiplexed measurements that pair signaling reporters with other measurements of cell state such as gene expression.[8] This type of dataset serves as a foundation for analysis of mutual information which estimates the amount of information a pathway's signaling dynamics encode about the nature of the stimulus.[9–11] In order to collect a dataset that includes the combination of many different signaling pathways, I propose building a combinatorial pooled reporter library using gRNA barcodes as described in Chapter 3 to precisely define and link the identity of each cell to its reporter . In this library, each cell will express reporters for three different pathways of interest. When used in a pooled context, systematic sampling of triplets of pathways will allow a comprehensive measurement of the entire network (Figure 4.1).

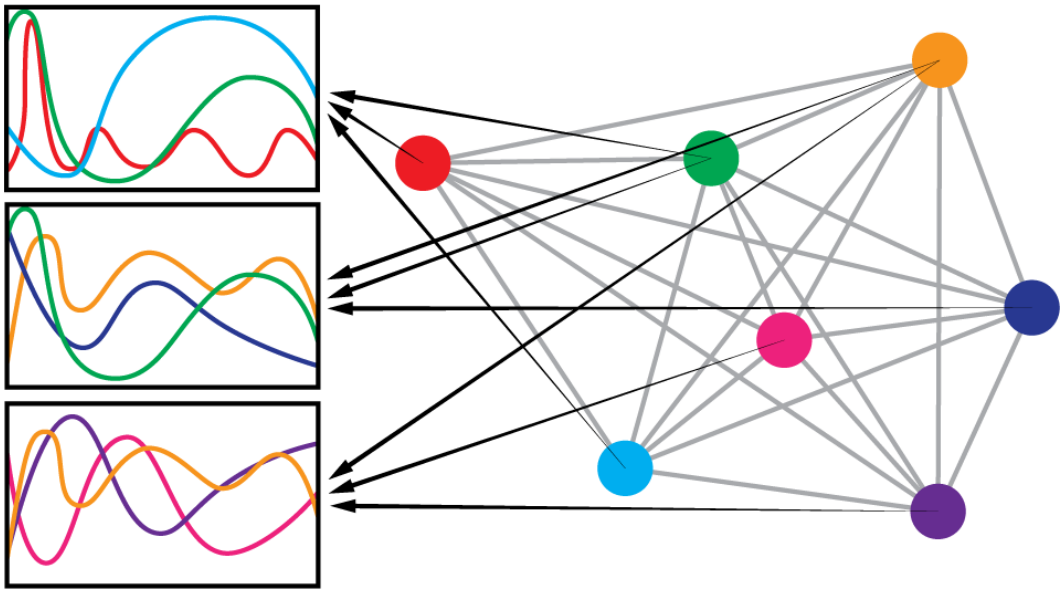


Figure 4.1: **Combinatorial measurements of signaling dynamics to capture network-level trends.**
In order to fully understand the inter-dependencies between nodes in a signaling network, sets of nodes need to be measured in individual cells. By systematically sampling sets of three nodes in individual cells in a pooled context, comprehensive measurements of an entire network can be captured in a single pool of cells.

4.2 Proposed experiment

In our combinatorial reporter library, each cell will express three different fluorescent proteins: EBFP, mOrange, and mClover. Each fluorescent protein will be conjugated to a signaling reporter or to a localization sequence for an organelle (Figure 4.2). Furthermore, each fluorescent protein construct will also express a barcode gRNA that can be labeled using FISH as described in Chapter 3. Using this method, each reporter and localization sequence will be assigned a different gRNA pattern. Additionally, the color of the FISH readout probes will correspond to the color of the fluorescent protein, which will enable readout of any given cells' set of reporters after live imaging collection. With this library, we can perform three color live imaging to observe the response of cells to various stimuli. To achieve labeling of the gRNA library after live imaging, cells will be fixed, bleached, and stained with FISH probes to identify each of the three fluorescent protein constructs expressed in the cell. Subsequently, additional end-point measurements will be collected, including SeqFISH to measure gene expression.

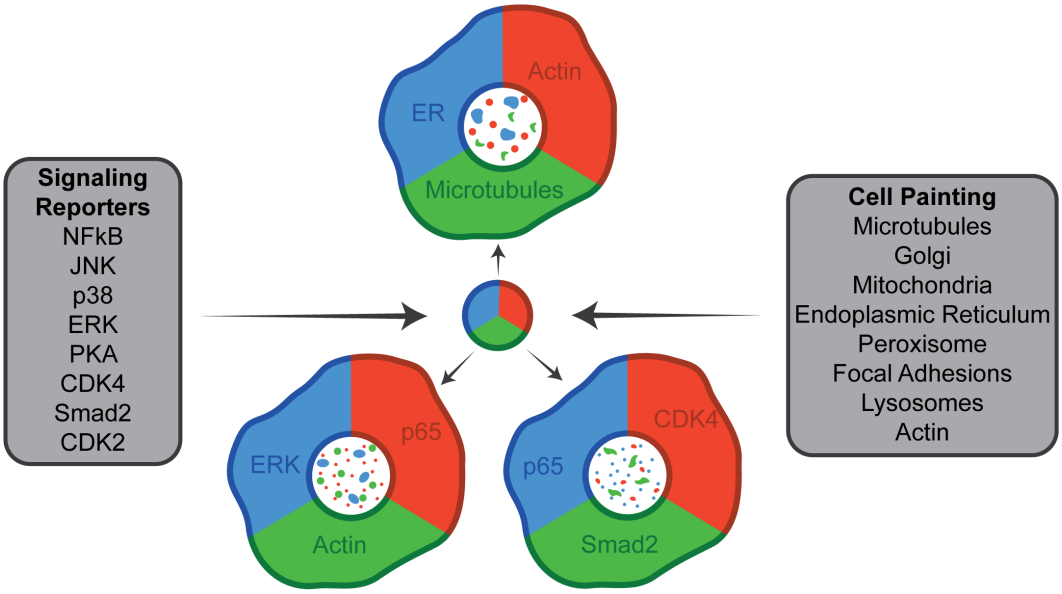


Figure 4.2: **A pooled combinatorial library of live cell reporters.**
Each cell in this pooled library will express three fluorescent proteins each conjugated to either a signaling reporter or an organelle localization sequence. Following live imaging, FISH staining will reveal gRNA barcode patterns in the nucleus that will identify which reporter was expressed in conjugation with each of the three fluorescent proteins.

4.3 Preliminary results

As a proof-of-concept experiment to demonstrate the power and utility of this method, we have assembled a set of live cell reporters that capture a variety of signaling pathways and organelles. We have selected a set of eight reporters that capture aspects of innate immune signaling, MAPK signaling, and cell cycle progression (Table 4.1). Each signaling reporter included here captures activity through translocation of the reporter between the cytoplasm and the nucleus. In addition to capturing signaling dynamics, we want to capture other features of cell state during live imaging. To this end, organelle labeling using Cell Painting has been shown to be an effective method to capture cell states and phenotypes in a variety of screening contexts.[12,13] Here we propose a live cell implementation of the Cell Painting assay that labels eight organelles using protein localization sequences conjugated to fluorescent proteins (Table 4.2).

In order to validate the performance of all reporters and labels in our experimental system, each of the selected reporters and organelle labels was cloned into a standard lentiviral backbone and transduced into HeLa cells. For signaling reporters, we verified that each reporter was responsive to an appropriate stimulus (as seen in the

Table 4.1: **Signaling pathway reporter constructs.**

Pathway	Reporter	Source
NFκB	p65-mClover	Addgene #127172[14]
JNK	JNK-KTR-mClover	Addgene #59151[6]
p38	p38-KTR-mClover	Addgene #59152[6]
ERK	ERK-KTR-EBFP	Addgene #59150[6]
PKA	PKA-KTR-mClover	Addgene #177199[15]
Smad2	mClover-Smad2	Addgene #118943[16]
CDK2	DHB-mClover	Addgene #136461[17]
CDK4	mClover-CDK4-KTR	Addgene #126680[18]

Table 4.2: **Organelle labeling constructs.**

Organelle	Reporter	Source
Actin	Lifeact-mOrange	Addgene #98877[19]
Endoplasmic Reticulum	ER-mClover	Addgene #137804[19]
Focal Adhesions	mOrange-PXN	Addgene #129604[19]
Golgi Apparatus	mClover-Giantin	Addgene #98880[19]
Lysosomes	LAMP1-mOrange	Addgene #98882[19]
Microtubule	mClover-MapTau	Addgene #137808[19]
Mitochondria	4xMTS-mClover	Addgene #98876[19]
Peroxisome	mOrange-peroxi	Addgene #137806[19]

example shown in Figure S4.1). For organelle labels, we verified that the pattern of the label corresponded to the correct organelle (Figure 4.3). Additionally, this process allowed us to verify that all sequences could be expressed by our chosen promoter, PGK, and successfully packaged as lentiviruses.

To assemble the final library of pooled combinations, we tested a strategy that would facilitate future split-pool viral transduction. In this approach illustrated in Figure 4.4a, subpopulations of cells can be transduced separately with EBFP-conjugated reporters. These subpopulations can then be recombined prior to resplitting for separate transduction with each of the reporters conjugated to mClover. Finally, the process of pooling and splitting can be repeated a third time for the mOrange reporter. The final population of cells will contain individual cells that represent all possible combinations of reporters. In order to perform sequential transduction rapidly with minimum cell passages, we adapted our standard lentiviral transduction protocol, which calls for typical adherent cells, to a variation performed with suspension cells. With this new suspension transduction protocol described in Section 4.6.3, all three stages of viral transduction can be performed within a few hours before allowing the cells to adhere normally. A preliminary version of this protocol was used to

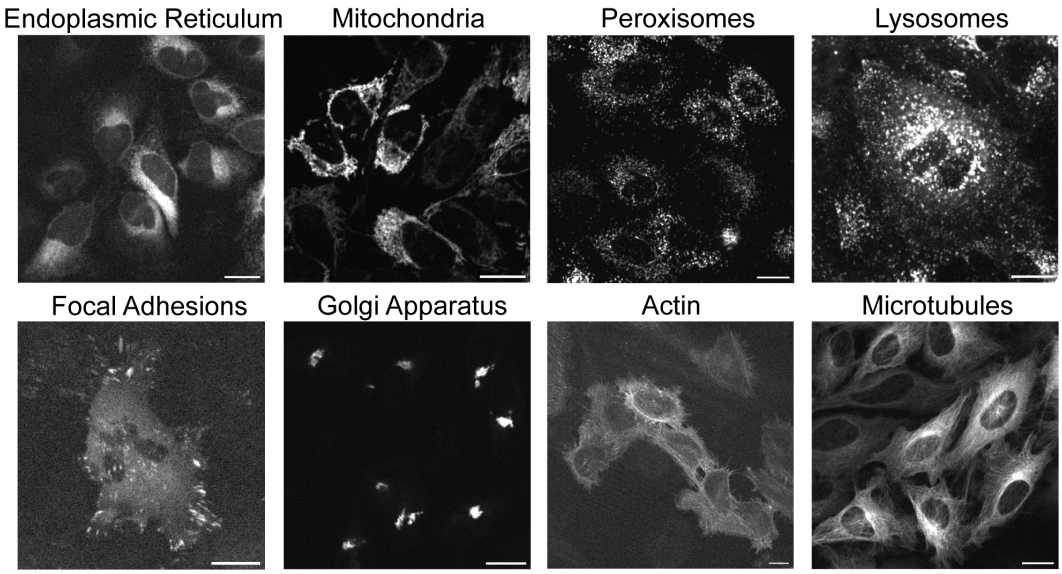


Figure 4.3: **Live cell painting.**
Sample images of cells transduced with each of the eight organelle reporter constructs. Sequence sources used for viral transduction are available in Table 4.2. Scale bars, 20 μm.

sequentially transduce a population of cells with three different viruses and we were able to successfully produce triple positive cells using this strategy (Figure 4.4b).

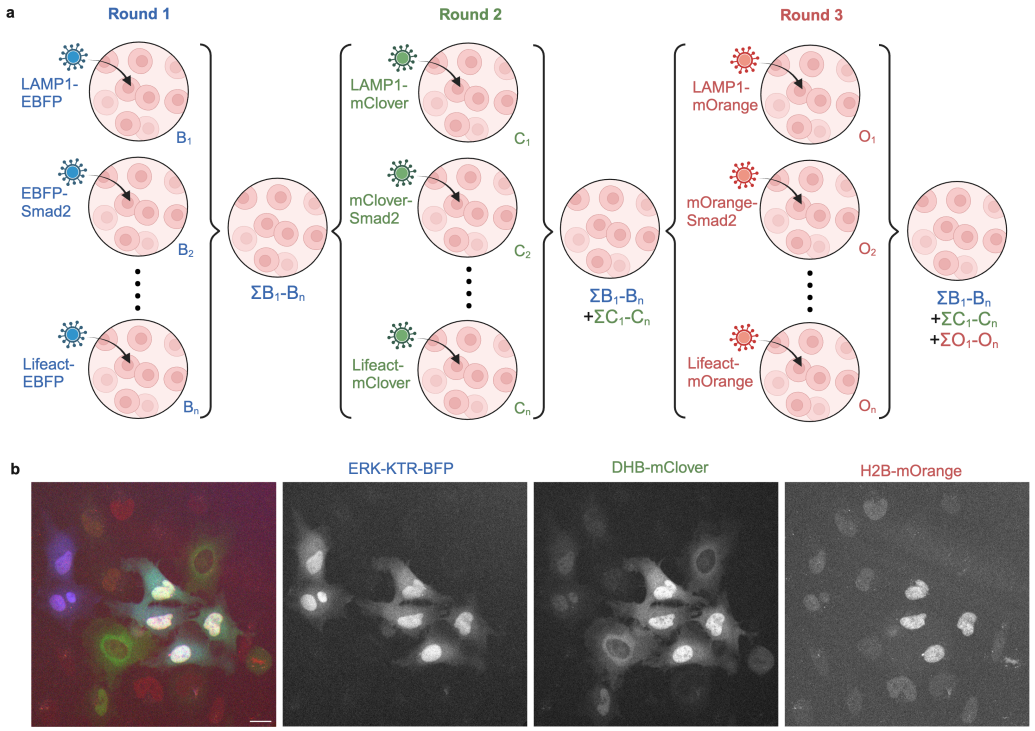


Figure 4.4: **Split-pool transduction for combinatorial library assembly.** (a) During round 1, separate populations of cells in suspension are each transduced with an EBFP reporter. The populations are then pooled together and immediately split into new subpopulations for transduction with a mClover reporter. Finally, the process of pooling and splitting is repeated a second time prior to transduction with a mOrange reporter. The final population of cells will contain all possible combinations of EBFP, mClover, and mOrange reporters. (b) A sample population of cells sequentially transduced in suspension with three reporters: ERK-KTR-EBFP, DHB-mClover and H2B-mOrange. This FOV contains multiple cells that express all three reporters. See Section 4.6.3 for additional experimental details. Scale bar, 20 μm. Created with Biorender.com.

4.4 Future directions

To date, I have established a panel of live cell reporters that have been validated for stable expression in HeLa cells. I have developed a preliminary protocol to enable the rapid generation of a combinatorial pooled population of cells. With these preliminary steps and validation experiments complete, I am prepared to order a set of constructs that pairs each reporter/fluorescent protein combination with the assigned gRNA. The datasets generated by these experiments will be able to pair live cell imaging collections with barcode gRNA FISH to identify each reporter and SeqFISH to measure end-point gene expression. For each stimulus in the panel, multiple dose concentrations will be captured above and below the optimized

dosage in order to facilitate a dose-response analysis when measuring the mutual information between a stimulus and a pathway's dynamic response.

4.5 Conclusion

When investigating the relationship between different components of a signaling network, arrayed measurements of reporters of individual pathways are generally ineffective, because of the acknowledged presence of variation between wells in a plate or replicates of an experiment. To overcome this limitation, two methods have been pursued in order to capture measurements of multiple signaling pathways. The first is to express and combine multiple reporters in a single cell.[6,20,21] This approach has allowed the measurement of up to four reporters in a single cell and revealed potential connections between the dynamic responses of different pathways. Alternatively, pooled libraries of reporter cell lines have been generated such that each cell expresses a different reporter whose identity can be revealed after the conclusion of the live imaging experiment.[22,23] Pooled reporter experiments have previously been developed that were able to span up to twelve different pathways and captured aspects of synchronicity between different pathways. Despite these advances, methods to capture both multiple reporters in a single cell and many reporters of different pathways in a single experiment have not yet been available. This has limited the precise analysis of the interactions of signaling pathways at a larger scale.

To overcome these existing limitations, I propose to combine our existing work using gRNA-based optical barcodes (Chapter 3) to create a combinatorial library of reporters for signaling activity and cell state (Figure 4.2). As shown in Figures 4.3 and S4.1, we have collected a panel of signaling reporters and organelle labels and demonstrated that they can be stably expressed in HeLa cells. In addition, we have developed a protocol for lentiviral transduction in suspension that will enable rapid generation of the combinatorial cell line using a split-pool approach (Figure 4.4). Using this pool of cells, we can capture measurements of multiple reporters in a single cell while sampling combinations from a panel of sixteen reporters of signaling pathways and cell state. This approach will enable the collection of a comprehensive dataset of cell signaling and state that would be infeasible to collect in an arrayed fashion.

Table 4.3: **Signaling pathway stimuli.**

Pathways	Stimulus	Concentration
JNK, p38	Anisomycin	50 ng/ml
ERK	EGF	100 ng/ml
PKA	IBMX	250 μM
NFκB	TGF-α	2 ng/ml
Smad2	TGF-β	10 nM

4.6 Methods

4.6.1 Lentiviral construct validation

Each reporter sequence was cloned into the pLex_305 (Addgene #41390) lentiviral backbone that drives expression under the PGK promoter. For preliminary testing, each reporter was tested with one of three fluorescent proteins: EBFP,[24] mOrange,[25] mClover.[26] The combinations of reporters and fluorescent proteins used in the initial tests are listed in Table 4.1 and Table 4.2. Lentiviral packaging was performed by the Janelia Viral Tools support team. HeLa cells (ATCC CCL-2) were cultured in Minimal Essential Media (EMEM; Cytiva SH30024.FS) supplemented with 10% fetal bovine serum (Thermo Fisher #26140079) with 100 U/mL penicillin/streptomycin (Caisson PSL01). Cells were transduced by adding virus (MOI 7) and polybrene (10 μg/ml) and centrifuging at 800 g for 1 hour. Cells were imaged two days after transduction to verify that the construct was correctly expressed and that the signaling reporters responded to stimulation.

4.6.2 Stimulus optimization

The combinations of reporters and stimuli tested are listed in Table 4.3. For each test case, cells (10^4 cells in 100 μl) were seeded on 96-well glass plates coated with fibronectin and incubated overnight. Reporter plasmid DNA (100 ng) was diluted in 10 μl of Opti-MEM prior to adding TransIT-LT1 (Mirus, 0.2 μl) and incubated for 30 minutes at RT prior to adding to cells. Cells were imaged 24 hours after transfection. Live cell imaging was performed using five minute intervals. Several FOVs were selected and imaged for at least one time point prior to dosing with the appropriate stimulus. Imaging continued for a minimum of two hours and ended after a response to the stimulus was observed. Initial stimulus concentrations were drawn from the literature and titrated as needed for HeLa cells.

4.6.3 Lentiviral suspension transduction

HeLa cell suspension (2 ml of 1.25×10^5 cells/ml) was added to a well in a 6-well ultra-low attachment plate. Cells were transduced by adding the first virus (MOI 5) and polybrene (10 μg/ml), centrifuging at 800g for 10 minutes and shaking at 250 rpm for 30 minutes. The cells were then pelleted and resuspended in 2 ml of fresh media along with the second virus (MOI 5) and polybrene (10 μg/ml) in an ultra-low attachment plate before repeating the centrifugation and shaking steps. Subsequently, the cells were again pelleted and resuspended in fresh media along with the third virus (MOI 5) and polybrene (10 μg/ml) before performing the final centrifugation and shaking steps. Finally, cells were transferred to a 6-well glass plate coated with fibronectin and allowed to recover for two days prior to imaging.

4.6.4 SeqFISH library preparation

In order to prepare a panel of target genes for SeqFISH labeling, bulk RNA sequencing must be performed on representative samples to establish the relative expression and expected fold change of genes of interest.

RNA sequencing

HeLa cells (1.5×10^5) were seeded in a 6-well plastic plate and allowed to adhere overnight. Cells were then dosed with the following drugs and incubated for four hours: anisomycin (50 ng/ml), EGF (100 ng/ml), IBMX (3-isobutyl-1-methylxanthine, 250 μM), TGF-α (2 ng/ml) and TGF-β (10 nM). An undosed sample was also collected in parallel. Subsequently, RNA was collected using the Qiagen RNeasy plus mini kit for RNA purification, and contaminating DNA was digested and removed using the DNA-free kit DNase treatment from Ambion. Finally, the RNA samples were sequencing using the Illumina HiSeq 2500 using a PE50, 20M paired end run. RNA sequencing data processed using the Galaxy web server.[27] The Trimmomatic tool was used to trim sequences, then sequences were aligned to the human genome using HISAT2. FastQC and MultiQC were used to ensure the quality of the trimming and alignment outputs. The featureCounts tool was then used to create a file that contains transcript counts for each human gene. The results of the feature counts for the undosed and dosed conditions for each drug were then analyzed and compared in Python.

4.7 Supplemental materials

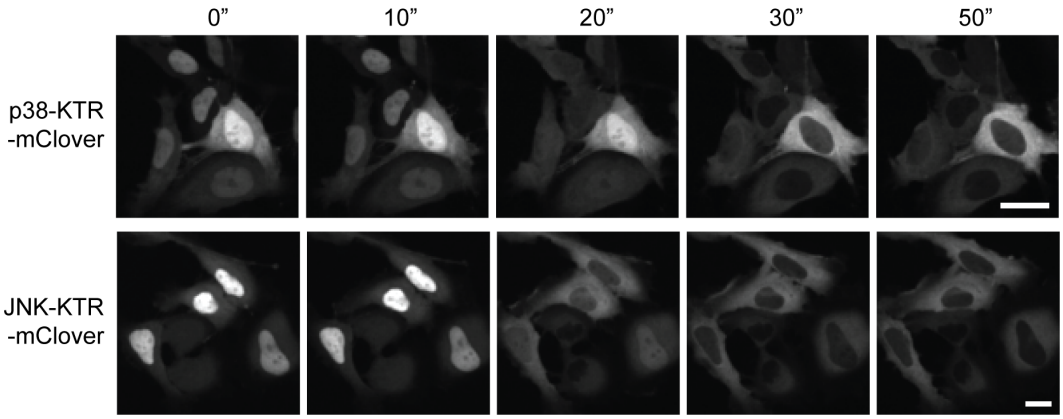


Figure S4.1: **HeLa cell response to anisomycin.**
HeLa cells stably expressing p38-KTR-mClover and JNK-KTR-mClover were dosed with anisomycin (50 ng/ml) following the collection of the first timepoint. Both reporters responded to stimulation as indicated by the translocation of the reporter from the nucleus into the cytoplasm. Scale bar, 30 μm.

References

1. Lee, T. K. *et al.* A noisy paracrine signal determines the cellular NF-κB response to lipopolysaccharide. *Science Signaling* **2,** ra65–ra65 (2009).

2. Cohen, A. A. *et al.* Dynamic proteomics of individual cancer cells in response to a drug. *Science* **322,** 1511–1516 (2008).

3. Albeck, J. G., Mills, G. B. & Brugge, J. S. Frequency-modulated pulses of ERK activity transmit quantitative proliferation signals. *Molecular Cell* **49,** 249–261 (2013).

4. Purvis, J. E. *et al.* p53 dynamics control cell fate. *Science* **336,** 1440–1444 (2012).

5. Purvis, J. E. & Lahav, G. Encoding and decoding cellular information through signaling dynamics. *Cell* **152,** 945–956 (2013).

6. Regot, S., Hughey, J. J., Bajar, B. T., Carrasco, S. & Covert, M. W. High-sensitivity measurements of multiple kinase activities in live single cells. *Cell* **157,** 1724–1734 (2014).

7. Pargett, M. & Albeck, J. G. Live-Cell Imaging and Analysis with Multiple Genetically Encoded Reporters. *Current Protocols in Cell Biology* **78,** 4–36 (2018).

8. Lane, K. *et al.* Measuring signaling and RNA-seq in the same cell links gene expression to dynamic patterns of NF-κB activation. *Cell Systems* **4,** 458–469 (2017).

9. Selimkhanov, J. *et al.* Accurate information transmission through dynamic biochemical signaling networks. *Science* **346,** 1370–1373 (2014).

10. Tang, Y. *et al.* Quantifying information accumulation encoded in the dynamics of biochemical signaling. *Nature Communications* **12,** 1272 (2021).

11. Cheong, R., Rhee, A., Wang, C. J., Nemenman, I. & Levchenko, A. Information transduction capacity of noisy biochemical signaling networks. *Science* **334,** 354–358 (2011).

12. Bray, M.-A. *et al.* Cell Painting, a high-content image-based assay for morphological profiling using multiplexed fluorescent dyes. *Nature Protocols* **11,** 1757–1774 (2016).

13. Tegtmeyer, M. *et al.* High-dimensional phenotyping to define the genetic basis of cellular morphology. *Nature Communications* **15,** 347 (2024).

14. Feldman, D. *et al.* Optical pooled screens in human cells. *Cell* **179,** 787–799 (2019).

15. Kudo, T., Lane, K. & Covert, M. W. A multiplexed epitope barcoding strategy that enables dynamic cellular phenotypic screens. *Cell Systems* **13,** 376–387 (2022).

16. Li, Y. *et al.* Spatiotemporal control of TGF-β signaling with light. *ACS Synthetic Biology* **7,** 443–451 (2018).

17. Spencer, S. L. *et al.* The proliferation-quiescence decision is controlled by a bifurcation in CDK2 activity at mitotic exit. *Cell* **155,** 369–383 (2013).

18. Yang, H. W. *et al.* Stress-mediated exit to quiescence restricted by increasing persistence in CDK4/6 activation. *eLife* **9,** e44571 (2020).

19. Chertkova, A. O. *et al.* Robust and Bright Genetically Encoded Fluorescent Markers for Highlighting Structures and Compartments in Mammalian Cells. *bioRxiv* (2020).

20. Mehta, S. *et al.* Single-fluorophore biosensors for sensitive and multiplexed detection of signalling activities. *Nature Cell Biology* **20,** 1215–1225 (2018).

21. Linghu, C. *et al.* Spatial multiplexing of fluorescent reporters for imaging signaling network dynamics. *Cell* **183,** 1682–1698 (2020).

22. Kaufman, T. *et al.* Visual barcodes for clonal-multiplexing of live microscopy-based assays. *Nature Communications* **13,** 2725 (2022).

23. Yang, J.-M. *et al.* Deciphering cell signaling networks with massively multiplexed biosensor barcoding. *Cell* **184,** 6193–6206 (2021).

24. Ai, H.-w., Shaner, N. C., Cheng, Z., Tsien, R. Y. & Campbell, R. E. Exploration of new chromophore structures leads to the identification of improved blue fluorescent proteins. *Biochemistry* **46,** 5904–5910 (2007).

25. Shaner, N. C. *et al.* Improving the photostability of bright monomeric orange and red fluorescent proteins. *Nature Methods* **5,** 545–551 (2008).

26. Bajar, B. T. *et al.* Improving brightness and photostability of green and red fluorescent proteins for live cell imaging and FRET reporting. *Scientific Reports* **6,** 20889 (2016).

27. Afgan, E. *et al.* The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. *Nucleic Acids Research* **44,** W3–W10 (2016).

Chapter 5

CONCLUSION

The exponential growth in the ability of machine learning to interpret and decipher the world around us has facilitated a gold rush across scientific disciplines. Behaviors and observations that were previously too complicated or convoluted for a human to decode or mathematically pull apart into principal components can now be solved with an increasing variety of machine learning models. With new model architectures being devised to handle new tasks, or improve the modeling of older tasks, it is now possible to ask questions in biology that were not possible even a few years ago. I sought to leverage these advances to enable the collection of multimodal measurements of pooled libraries, while taking advantage of CRISPR-Cas9 to generate traceable labels to link experimental modalities. However, pursuing this type of experiment first called for development of methods to process the vast amounts and varied types of data which result from pooled screens and multimodal measurements.

Fundamentally, major advances in deep learning come from two places, advances in model architecture, and increased quantity, quality, variety, or processing of the data fed into a model. The foundation of any deep learning model is an extensive training dataset that associates the input data with the desired output from the model. Without high-quality data, a deep learning model is starting off from a deficit. However, the work that goes into building these datasets is often an unacknowledged component of deep learning model development. This body of work sought to utilize deep learning to improve the design and analysis of biological microscopy experiments, to create a platform capable of augmenting the latter of these two deep learning modalities. To accomplish this feat, I utilized deep learning for two different objectives, which in tandem would provide additional data acquisition and quality improvements. The first objective is to robustly extract single cell features from microscopy data without needing extensive parameter tuning or oversight on new datasets. The second is to utilize deep learning as an integral tool in experimental design in order to reduce the technical difficulty of data collection.

A key focus of my work, connecting both Chapters 2 and 3, was the development of tools to support building new training datasets. In order to build DynamicNucle-

arNet, which was presented in Chapter 2, I developed a framework for versioning each piece of annotated data and tracking relevant metadata such as the microscopy collection conditions and the quality of the annotations. Typically the process of assembling a training dataset is highly manual and consists of smaller subunits of data organized in folders. Updates to the data may overwrite the previous version or be roughly versioned using additions to the file names. This process is time consuming and vulnerable to human error. Furthermore if an error is identified, it is often not possible to return to an earlier version of the dataset prior to the error. The data infrastructure that I developed replaces manual oversight with automated routines that build in versioning and create a reproducible process for interacting with data. Furthermore, this infrastructure was readily adapted to building several different training datasets described in Chapter 3. While the data infrastructure introduced in this work is currently fine-tuned towards our use cases, and would require investment by labs interested in its use to tailor to their use cases, the impact of this infrastructure on this thesis clearly indicates that there is significant value in developing better data management systems for the biological microscopy community. Minimally, it should be noted that investing time in establishing even a simple data management process for a single project is highly worthwhile, due to future time savings and increased resiliency to errors which can hamper deep learning model training. With a solid dataset development infrastructure in place, it becomes possible to rapidly build new datasets in order to prototype models for new tasks.

Leveraging the advances in data set acquisition and augmentation presented in this thesis, I was then able to assemble a new dataset to test the ability of deep learning models to classify the identify of a gRNA based on the nuclear labeling pattern that it generated (Chapter 3). The goal of this optical barcoding strategy is to be simple to execute, primarily by shifting the burden of complexity from the experimental data collection stage, which is limited by how quickly a human and microscope can acquire data, to the data analysis stage, which is primarily bounded by machine learning performance. In the pursuit of collecting multi-modal datasets, the time spent during each round of staining and imaging becomes a valuable commodity such that any steps that can be taken to eliminate the need for additional rounds of imaging are highly advantageous. With this aim in mind, we utilized the nuclear staining patterns generated by gRNAs to design an optical barcode that can be captured in a single round of imaging, and utilizes a deep learning model to read out each barcode.

The key challenge to overcome in developing this system, and the ML tools needed to decode the barcodes is found in collecting an appropriate dataset to use to train the model. Developing the foundation of this system required collecting new data for each new cell line, with additional overhead of the need to collect data from different perturbation conditions that will be represented in the final experimental dataset. Nevertheless, despite the time required to collect an appropriate dataset, this step represents a one-time investment that prepares for many future experiments. The optical barcoding method presented in Chapter 3 lays the foundation for an experimental platform that can be used to collect complex multimodal datasets such as those described in Chapter 4 that include live imaging of reporters, perturbation labeling with optical barcodes and measurement of gene expression with SeqFISH. However, the challenges, and the solutions identified in Chapter 3, highlight the importance of considering data collection and analysis as a unified process that can be jointly optimized to create an experimental design that maximizes the potential impact of the experiment.

With the tools and methods presented in this thesis, including a new framework for robust large data set acquisition and management, with reduced capacity for human error, and methods for generating large multimodal pooled library screens in single cells, it is easier than ever to apply the power of deep learning to the unique challenges presented by biology. Biology is composed of complex systems, with many deep and unseen interactions, with the results of those interactions taking up to days to fully manifest themselves. Unraveling these complex systems, one interaction at a time, is costly, time consuming, and error-prone. However, with the methods presented here, a scientist can step back from the minutiae and query the role of a protein in a system over time, or even the role of a system in how a cell changes over time, all while decoupling the speed of discovery from how fast a scientist can set up and collect individual experiments.

Appendix A

DEEPCELL LABEL USER MANUAL

How to Use DeepCell Label

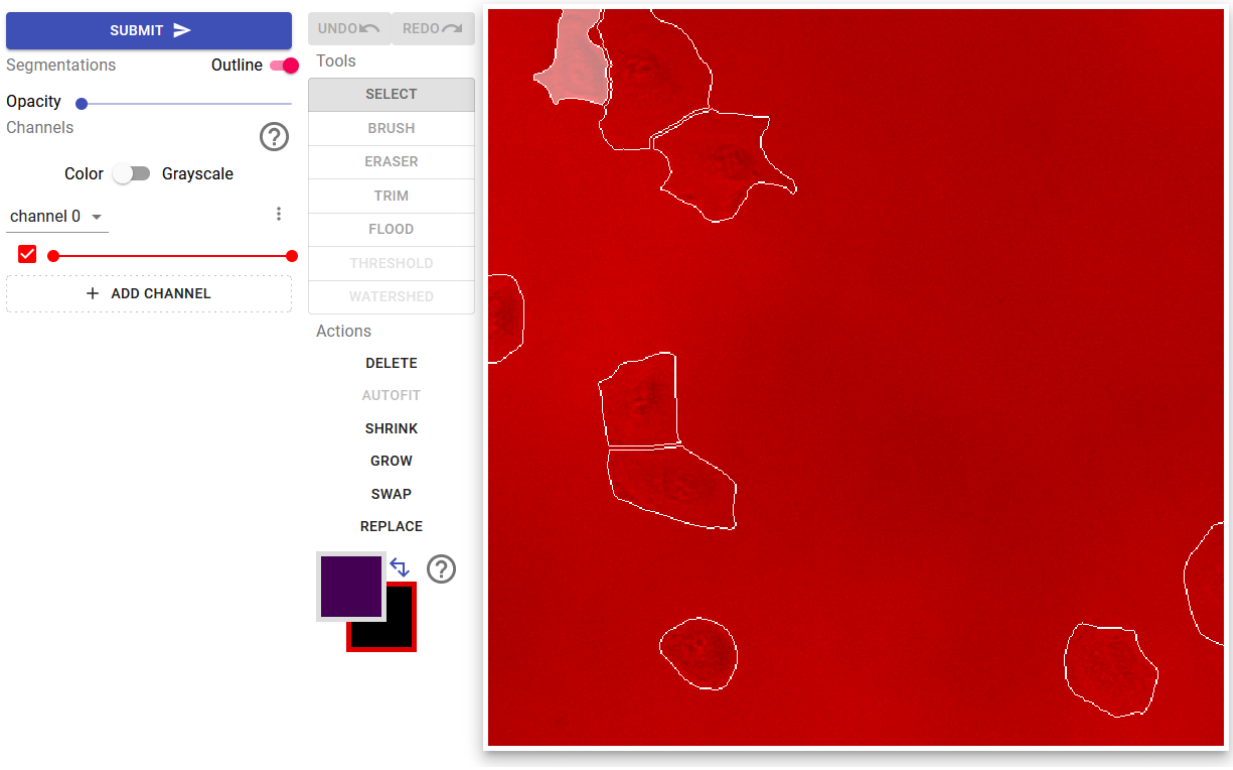
This guide shows how to use different annotation tools within DeepCell Label (DCL) to create or edit annotated biological images. Please refer to job instructions for specific corrections needed. Use the outline as a reference for particular tool descriptions.

Overview

DCL offers a suite of tools to interact with labeled or unlabeled images. Unique project URLs should have already been generated for your project. If you need access to these URLs or want to generate your own, please contact a member of DeepCell for further information.

Layout

When you click on a valid DCL URL, you should be taken to a screen similar to this:



The right side of the screen is the workspace, which contains the image and any annotations associated with the file that has been uploaded. The left hand side contains all of the tools available to manipulate the image or annotations.

The **Undo/Redo Buttons** allows you to undo (or redo) the changes you make on the canvas. Be aware that switching tools and changing annotation selections are considered “edits” and multiple clicks of the button may be required to fully restore the desired result.

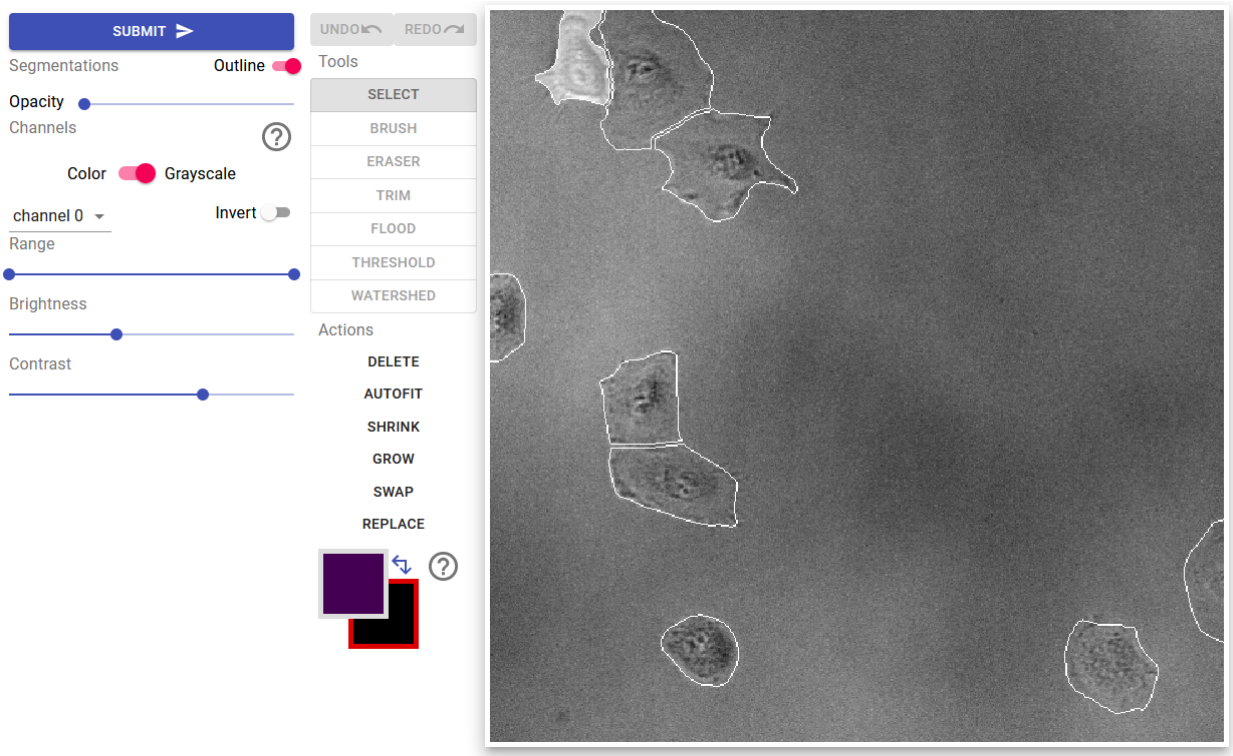
The **Palette Boxes** (located below the REPLACE button) allows you to visualize current label selections. The upper-left palette box shows the color of the currently selected label. Hovering over this box allows you to cycle through the different labels by clicking the **left and right arrows**, add a new label value by pressing the **“+” symbol**, or deselect the current label by pressing the **“x” symbol**. The lower-right palette box displays the label value of the canvas (black = no label value). The canvas is where edits can be made with tools and actions. The label value of the canvas can be changed to match any label (for more information, see the description of the **Select Tool**).

When you’ve finished your corrections, use the **Submit Button** to upload them.

Preparing the canvas:

When an image is loaded into DCL, the default view assumes your image contains multiple channels and will load your image with false coloring. Move the slider under each color to change the intensity of that color.

If you are working on phase images or single color images, you may want to switch the toggle to **Grayscale**. When in grayscale mode you are able to have more control over the brightness and contrast of the image. Move each slider to a position that allows you to see the most features within an image. These settings can be changed at any time.



The outlines of any existing annotations will be displayed by default. They can be turned on/off by toggling the **Outline** button. If the image being annotated contains too many objects, it will be beneficial to see the annotations filled in with different colors. This can be done using the **Opacity** slider. Pressing “z” will also cycle through three preset opacity values (0, 30%, and 100%)

Zooming and panning:

To zoom into the image, use the scroll wheel or the “+” and “-” keys. To move around once zoomed in, hold down the spacebar then click and drag the mouse.

Changing channels:

Change channels with the "c" key.

Changing frames:

Use “a” or the left-arrow key to go back one frame, and “d” or the right-arrow key to go forward one frame. Going forward from the last frame will loop back around to the start of the file. Some datasets will only have one frame per file.

Editing Tools

Select Tool:

Use the **select tool** to click on the annotation that you want to edit. Selected annotations will be highlighted in white and the annotation color shown in the upper-left palette box. Only selected cells can be interacted with through other tools or actions. Press “esc” to deselect the current annotation. If an object needs a new label, click the “+” symbols on the upper palette or press “n” to select an unused label. To select 2 labels at once (in the case of swapping or replacing), click on the label to replace twice (this will now be the background label in the lower-right palette box) and click on the label to paint with once (this will become the foreground label in the upper-left palette box).

Brush Tool:

Use the **brush tool** to make corrections to the current selected annotation. Click once to paint an area the size of the cursor. Click and drag to make larger edits. The brush size can be adjusted using the up- and down-arrow keys.

To adjust the border between 2 labels, first **select** the label you want to paint with, then hold down shift and click on the label you want to overwrite. Once you see the labels properly represented in the palette box (the top-left box will be the label you are painting with and the bottom-right box will be the label that is being brushed over), use the brush tool as you would normally.

Eraser Tool:

Use the **eraser tool** to remove portions of the current selected annotation. Click once to erase a portion of the label according to the size of the cursor. Click and drag to make larger edits. The eraser size can be adjusted using the up and down arrow keys. Press “z” to quickly switch to the **brush tool**.

Trim Tool:

Most of the time annotations should be connected. To quickly remove disconnected portions of

an annotation, select the **trim tool** and click on the part of the annotation you want to keep. If the annotation to be fixed was not originally selected, click again for the trim tool to be applied.

Flood Tool:

Occasionally, annotations will contain empty spaces. To quickly fill these spaces, select the **flood tool** and click on the empty space to be filled. Make sure the correct annotation has been selected otherwise another label could be applied to the space.

Actions

Actions have been designed to make certain common edits easier to perform. Make sure to use the **select tool** to click on the annotation that needs to be corrected.

Delete:

To delete a whole annotation, click on that label with the **select tool** and the click **delete**. This is useful when whole objects are labeled incorrectly.

Shrink/Grow:

Annotations that are uniformly too large or too small can be edited using the **shrink** or **grow** actions. Both shrink and grow add or subtract one pixel around the label respectively.

Replace:

Sometimes annotations of a single cell have been split in error and given two different labels. To correct this, double click one of the labels, click the second label and click **replace.** Repeat as necessary. Use the **brush tool** to make all parts of the label connect.

Swap two labels:

This action is only needed to make sure that labels match up with each other correctly across different frames of a file. Select two different labels by double clicking on one label and then clicking on the other. Click **swap** to swap values.

Appendix B

SEGMENTATION CORRECTION INSTRUCTIONS

DeepCell Label has been updated to improve the interface, so some functions and tools have changed. Please read the updated manual (DCL User Manual) for general instructions as to how to use this new tool. Once you are familiar with the new layout and features, refer to the following examples to check for specific work you need to do to complete the job. Thank you!

Overview

In this task you will work with a tool to create annotations of densely packed cells. This tool, DeepCell Label, has many features in it to draw and correct annotations. In this task, you will use a set of those features to create annotations of all the cells in the image.

Background

Our group is in the process of writing a computer program to automatically identify cells in culture. To help us do this, we're asking you to help create annotated datasets where single cells are manually identified. We can then feed this into the computer program to teach it how to accurately identify cells by itself. This program will be used in other research laboratories to study a range of topics, including viruses, cancer cells, and immune cells. **The software we create is only as good as the data used to create it, so accuracy in your annotations is extremely important.**

In this job, each file contains many cells. Many of the cells are already labeled correctly, but some cells in the file have labels that are too big, too small, or overlap adjacent cells. You will correct the labels with the provided tools.

Your Task

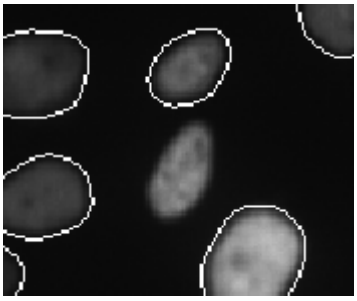
For this job, **your goal is to make sure that the border around each cell is in precisely the correct spot.**

Algorithm Accuracy: High

This means that **most** of the cells in this image are correct. **Some** of the cells in this job will need to be changed.

New Function: Autofit

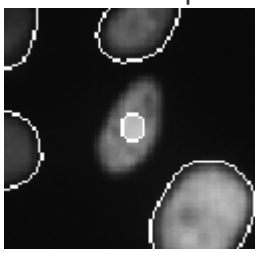
A new function has been added to the tool set that automatically fills in missing labels.



To use this function do the following:

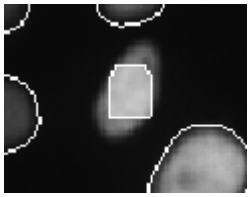
1. Click on the brush tool

2. Press “N” to select a new label
3. Fill in a small portion of the cell to be labeled

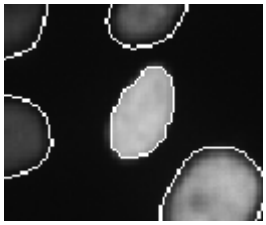


4. Click on Autofit

5. Wait until the label gets created

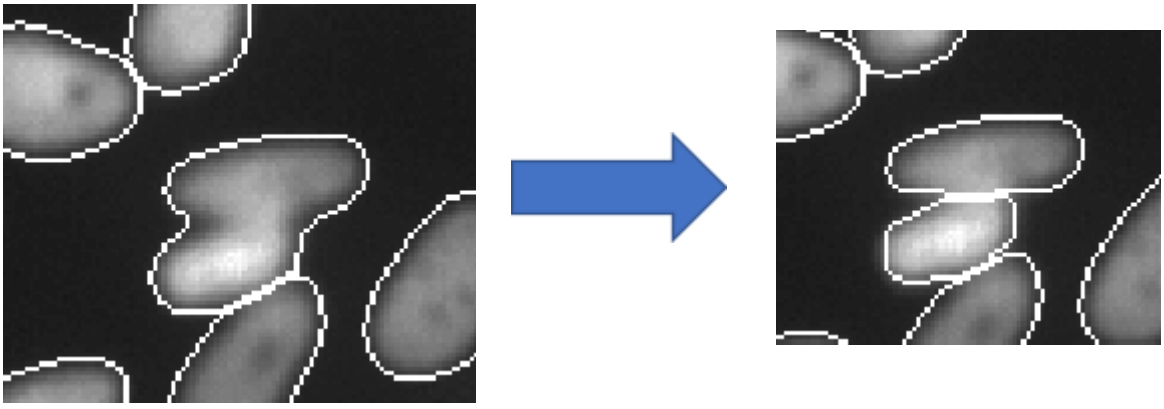


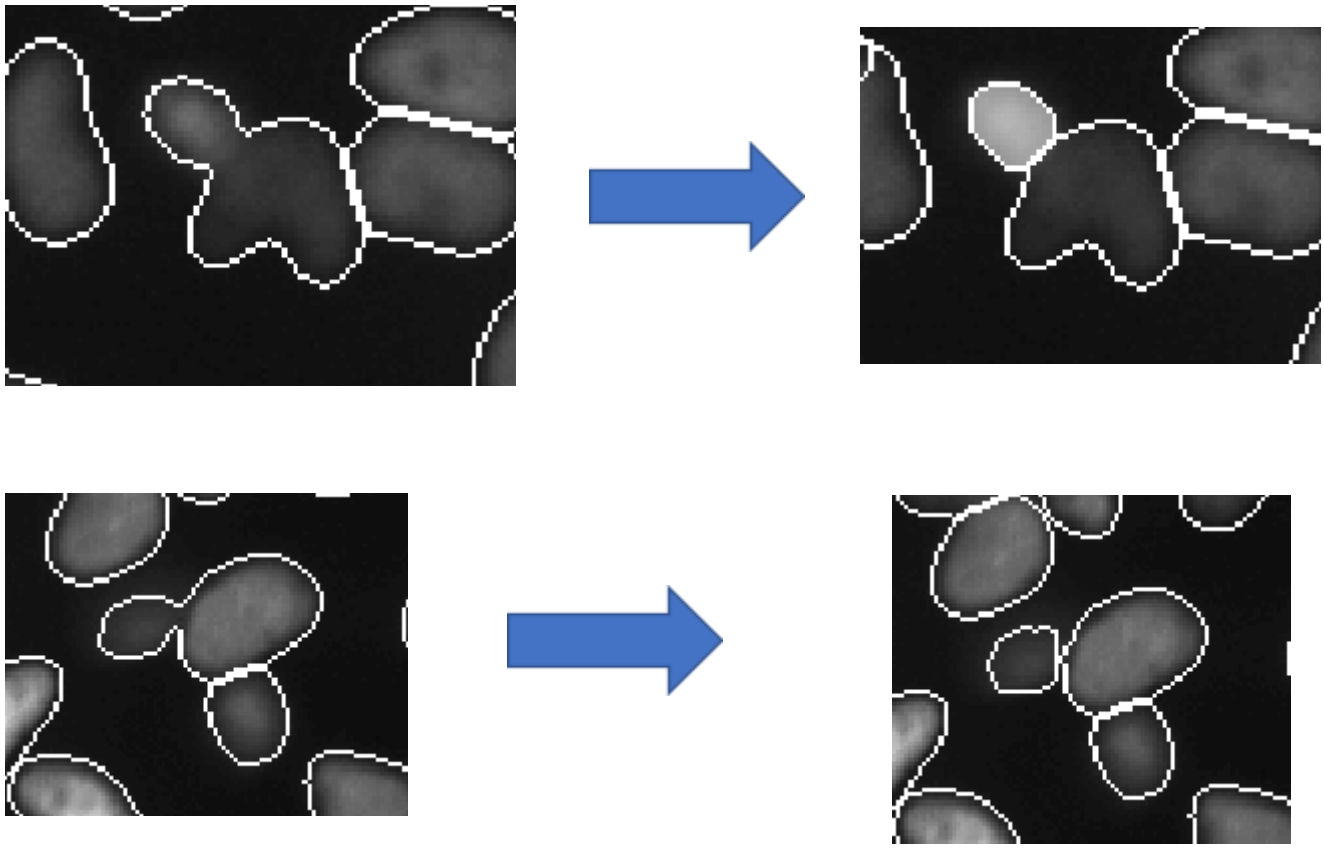
6. Evaluate the label and made edits as necessary



Specific Instructions for this job

The most common errors in this job are multiple cells with a single label. These labels need to be separated.



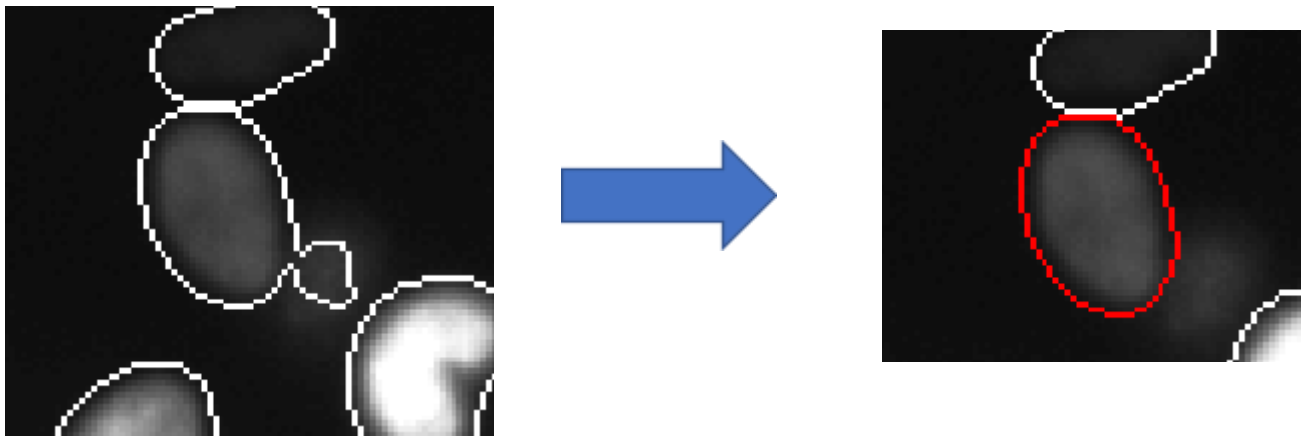


General Instructions

Less common errors may also be found in this job. Watch out for and correct the following errors.

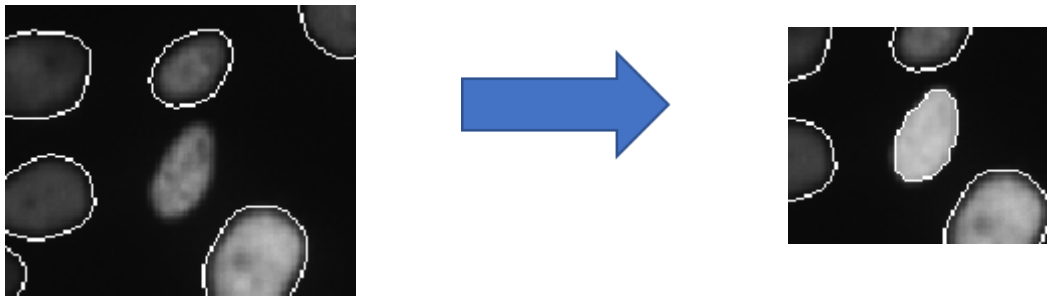
Label too large

When a cell has a label larger than the cell, fix the label to match the size of the cell.



Missing cell

If a cell doesn't have a label, you should add one



If DCL doesn't work properly

First, make sure you have clicked inside the viewing window, which may solve your issue. This is a new tool that, despite testing, may still have some bugs. The tool may also have been updated since the last time you accessed it. If it does not seem to load correctly, try refreshing the page without cached data. If this does not work and you encounter an error that prevents you from finishing correcting the file, please notify your manager and continue onto another file if possible.

Appendix C

TRACKING CORRECTION INSTRUCTIONS

Overview

In biology, cells naturally move over time. Occasionally, one cell divides into two cells. A cell that divides is a "parent cell" and the two new cells that result are "daughter cells." We want to track these movements and divisions. In this task, every cell in the movie has been labeled with a color and number. You will use DeepCell Label to correct the cell labels so that each cell has the same label across all frames. Additionally, you will find cell divisions and link the daughter cells to their parent cell.

Your Task

For this job, each different label value is represented by a color and number. You will follow each cell throughout the movie and check that each cell:

- Has the same label value in every frame
- Is linked to its daughter cells if it divides
- If there is a division, that the two daughter cells have different labels from each other and its parent

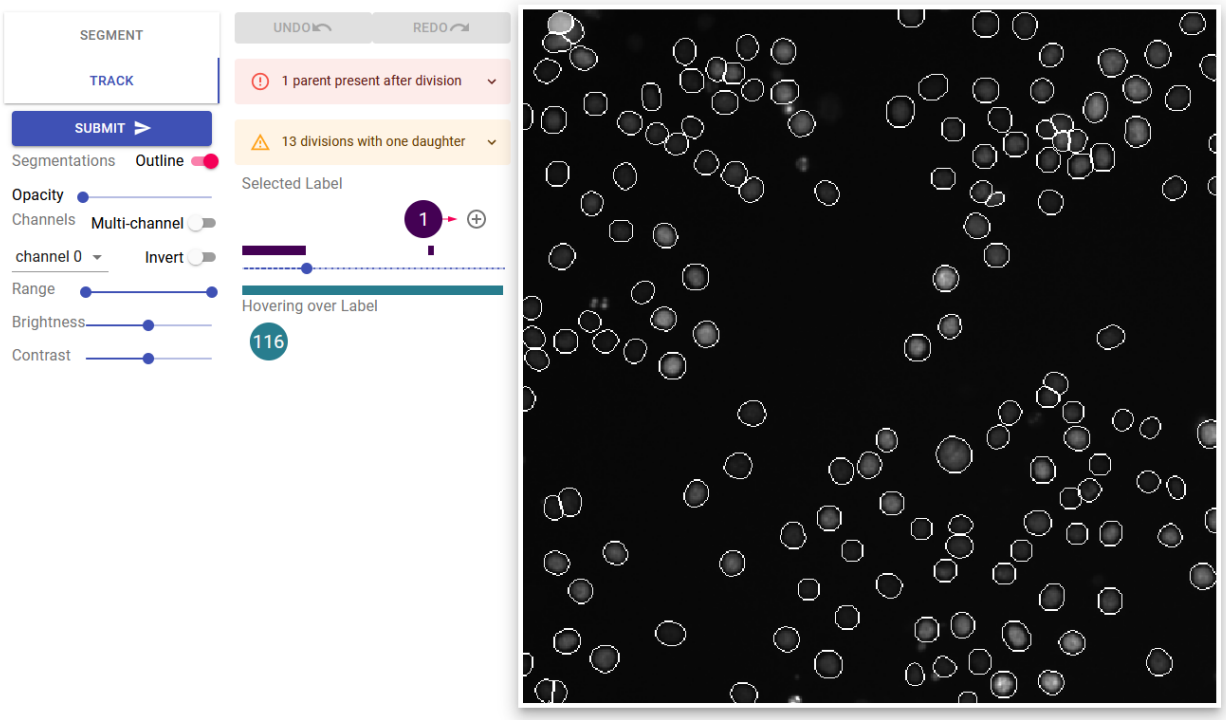
DCL has tools that will help with tracking these cells. Each tool for each use case is discussed below.

Some labels may disappear or migrate out of the field of view. This is ok. You should **not** make new labels or edit the shape of these labels.

Check **each label in each frame** throughout the whole movie. **You must look at every cell label in each frame before submitting the job**.

DCL tracking tools

When you click on the link you will see this (it may take a few moments to load):



If you have used DCL before, this will look familiar. This is the work area you will use in each job.

The work area is divided up into 3 areas:

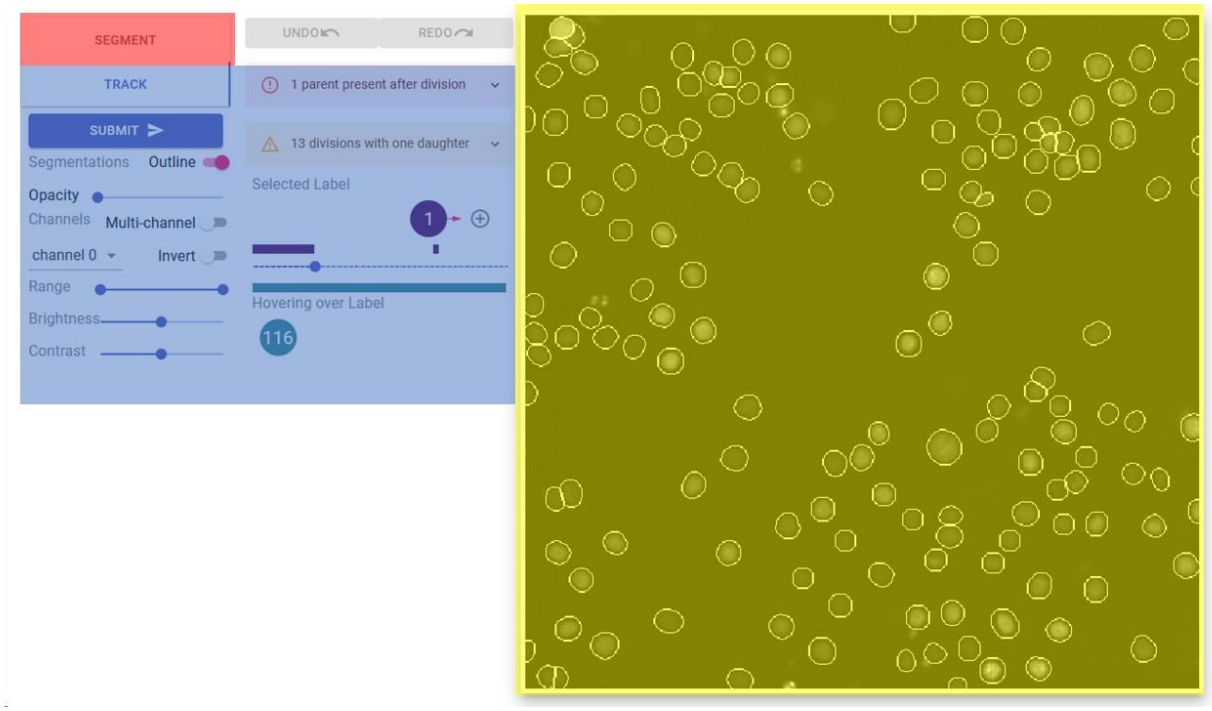
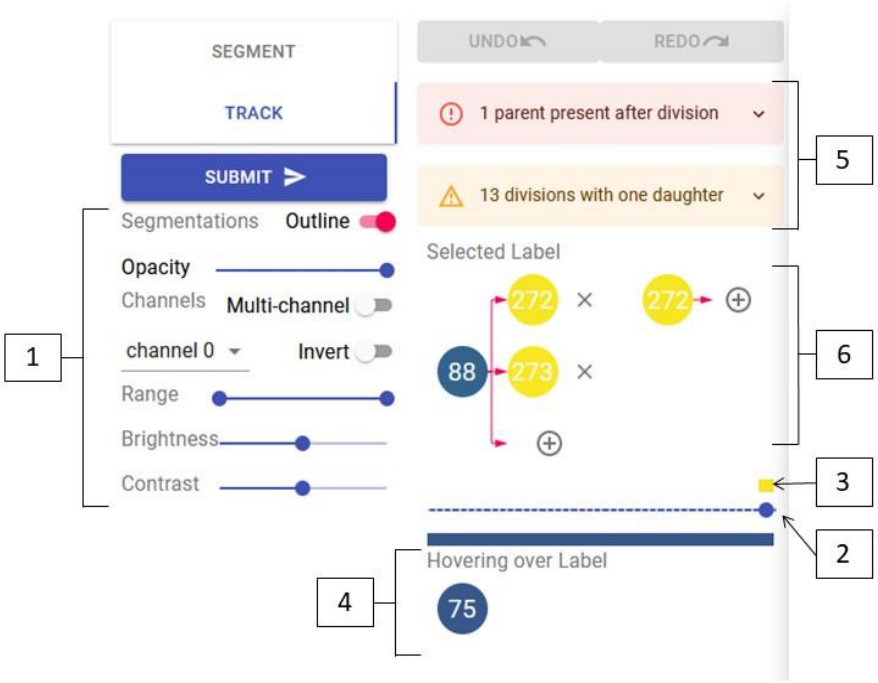


Image Canvas (Yellow) - This is where each frame of the movie and the cell labels will be displayed. The colors are there to help differentiate labels and to check if labels are staying the same throughout the movie.

Annotation Tools (Red) - These tools are for editing the label shapes or creating new ones. You should not need these tools for this task.

Tracking Tools (Blue) - These are the tools you will need to fix labels in the movie.



1. Image adjustments - These can help you see the movie better. For these jobs, increase the opacity to maximum so that only the labels are visible. You can press “Z” to change the opacity quickly.
2. Movie frame indicator - This shows which frame of the movie you are viewing. You can change frames by clicking on the bar or by pressing the “A” and “D” keys. “A” goes to the previous frame and “D” goes to the next frame.
3. Label frame indicator - This bar shows you in which frames the current selected label exists. (Note: the bar does not align perfectly to the movie frame indicator)
4. Cursor label indicator - This line shows the label value of the cell that the cursor is currently hovering over.
5. Error messages - The program checks for odd connections that exist in the movie. Pay close attention to these messages as they likely need to be corrected. Click on the messages to see labels that will need special attention. Red errors must be fixed. Yellow warnings need to be checked but may not require a change.
6. Lineage map - This map shows the currently selected cell and if that cell has a parent or daughters in the movie.

Interactions with the image canvas

In this job, you will correct incorrect label assignments throughout the movie. Use the following keybinds to interact with the image canvas:

“A” - moves one frame backwards in time

“D” - moves one frame forward in time

“[“ - Selects the previous label value (values will loop from end to beginning)

“]” - Selects the next label value (values will loop from end to beginning)

“Z” - Increases the opacity of the labels

General Instructions

The easiest way to interact with each job is to look at the label values to see when new cells appear and if labels change values in-between frames. New cells have a larger label number and will be yellow in color. These new cells are likely to be daughter cells as a result of cell division.

To set this up:

1. Press “Z” twice or increase the opacity slider to 100%
2. Press “[“ to select the highest numbered label
3. Click on the label number in the lineage map

This will take you to the last frame of this movie. You will then press “A” and “D” to cycle between frames in the movie. Follow the selected label through all the frames it is present in.

During this process, assess the following:

1. Does the label exist throughout every frame of the movie?
    a. YES: does the label stay constant throughout the movie?
        i. YES: move on to the next highest label by pressing “[“
        ii. NO: does the label switch places with another label?

1. YES: see Swapping labels section
2. NO: you likely have a situation where the parent label has been transferred to a daughter cell. See Changing parent cell to daughter cells section

b. NO: move on to #2

2. Does the label disappear because the cell merges with another cell?
   a. YES: Is the current label a daughter of the cell it merges into?
      i. YES: Is the other daughter cell also associated with the parent cell.
         1. YES: move on to the next highest label by pressing "["
         2. NO: See Changing parent cell to daughter cells or Adding daughter cells section
      ii. NO: see Adding daughter cells section
   b. NO: Has the label of that cell changed in a previous frame?
      i. YES see Combining labels section
      ii. NO: move on to #3
3. Does the cell disappear off the canvas?
   a. YES: scan all frames to see if the label stays the same if the cell reappears in the frame. Does the label stay the same?
      i. YES: move on to the next highest label by pressing "["
      ii. NO: see Combining labels section and Swapping labels section
   b. NO: the label has disappeared due to cell death. Move on to the next highest label by pressing "["

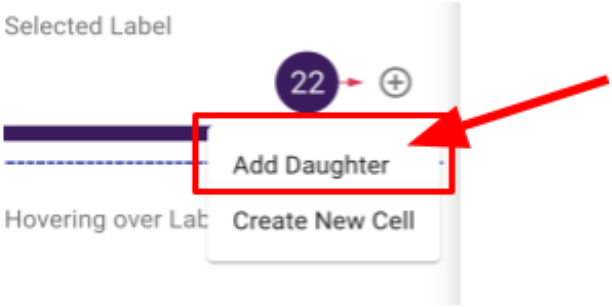
These general instructions should guide you through 90% of a job. Sometimes a cell needs multiple actions to correct it's label. Pay closer attention when correcting those cells. In the hardest cases, you may need to change a label to a new label value to correct all the cells. In that case, please see the Add new labels section

When you have finished reviewing all the labels, press the Submit button to complete your job.

Adding daughter cells

Use these instructions to add daughter cells to a parent cell when a cell divides.

1. Select the parent cell
2. Click the "+" symbol next to the parent cell label in the lineage map and select Add Daughter from the menu



3. Scan through the movie until the daughter cells appear (usually the next frame of the movie after the parent label ends)
4. Click on a daughter cell on the canvas

5. Wait for the lineage map to update

6. Repeat for the other daughter cell if necessary

When a parent label should be a daughter

Use these instructions when a parent cell divides but does not have a different label after dividing. A correct division will have different labels for the parent and daughters.

Frame 33 Frame 34

Example: Label 67 divides into two daughter cells in frame 34. Label 67 in frame 34 should be a new label instead of keeping the same label as before the division.

1. Select the parent cell

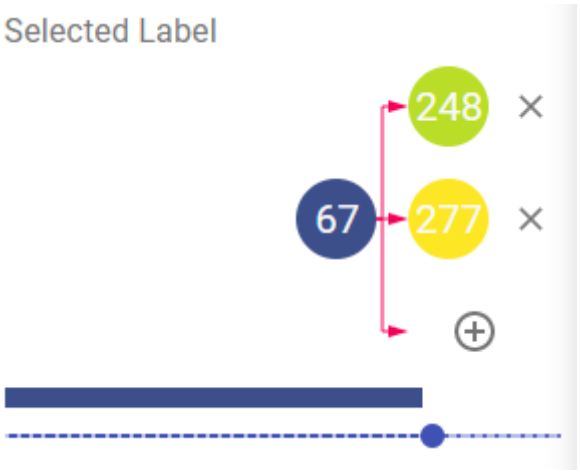
Frame 33

2. Click the "+" symbol next to the parent cell label in the lineage map and pick **Add Daughter** from the menu

3. Scan to the frame where the parent cell divides

Frame 34

4. Click on the parent label in the canvas in the first frame where it becomes a daughter
5. Wait for the lineage map to update



Frame 34

The daughter will have a new label from this frame on.

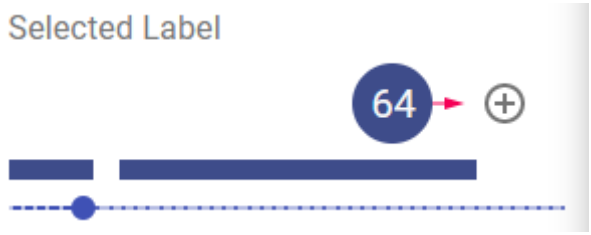
Combining labels across frames

Use these instructions when one cell has different labels in different frames but should have the same label. Pay attention to these instructions as they are currently not easy.

Frame 6 Frame 7 Frame 9

Example: Label 64 switches to 171 in frame 7.

1. Select one of the labels to be combined (usually the lower number)



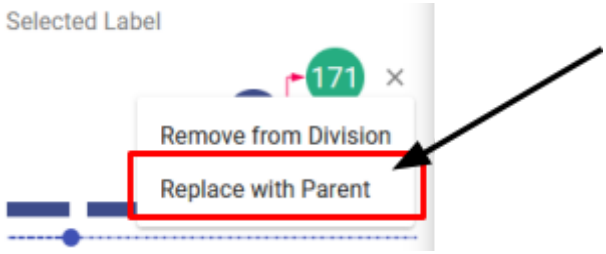
2. Click the "+" symbol next to the parent cell label in the lineage map and select "Add Daughter"

3. Scan to the frame where the cell changes to a different label without dividing (make sure the first label does not exist in this frame) and click on the new, different label you want to replace.

Frame 7

4. Wait for the lineage map to update

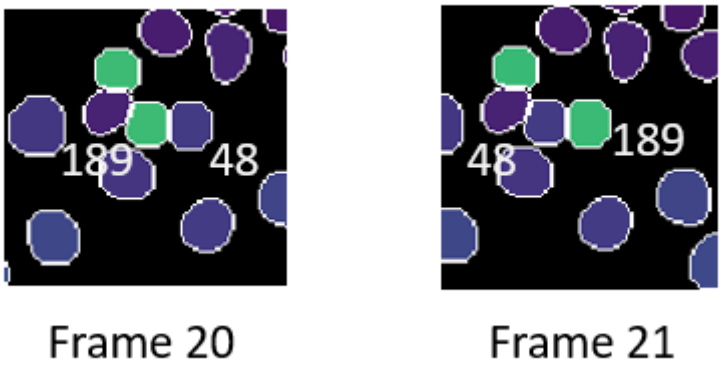
5. Click on the “x” symbol to the right of the daughter label and select “Replace with Parent” from the menu



6. Wait for the lineage map to update

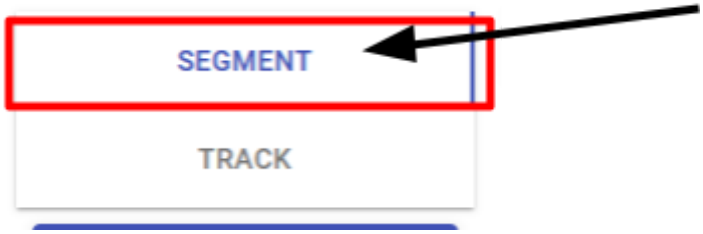
Swapping labels

Use these instructions to switch label values between two different labels.

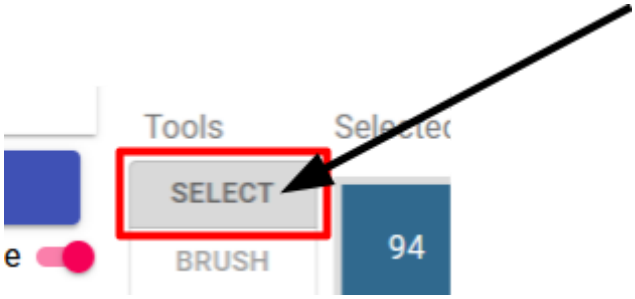


Example: Labels 189 and 48 switch places in frame 21.

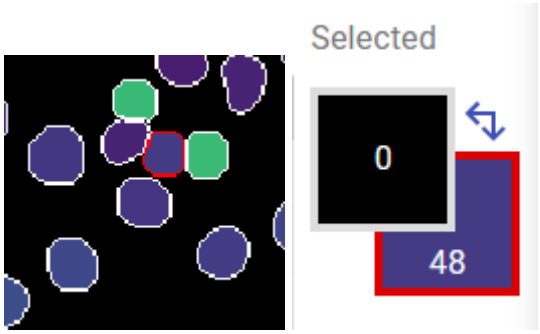
1. Click the Segment button to change tools.



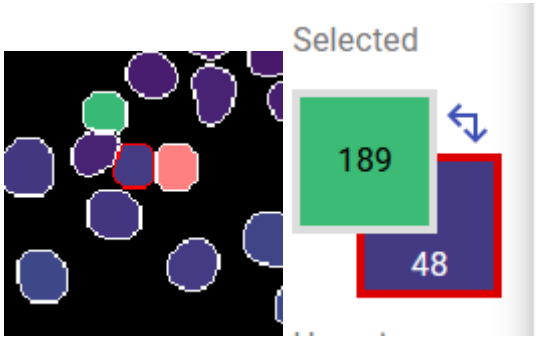
2. Make sure you are using the Select tool under the Tools buttons



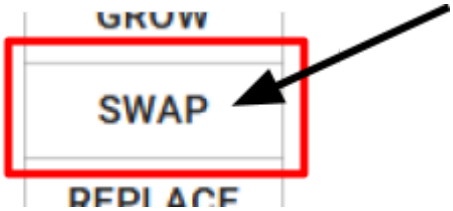
3. Double click on one of the labels to be changed so it is outlined in red



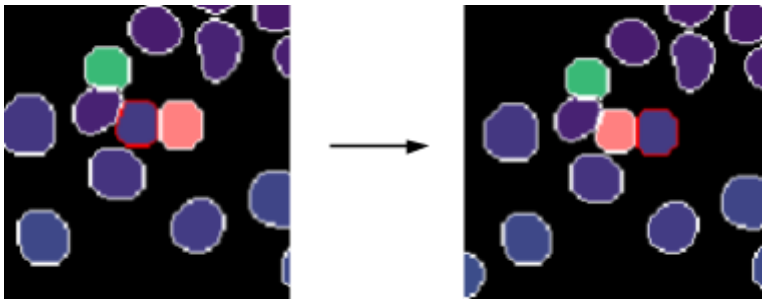
4. Click on the other label so it becomes red inside



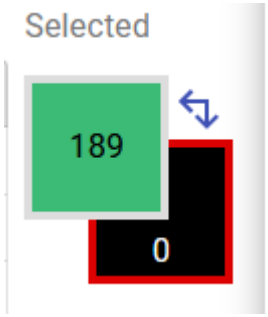
5. Click “Swap” under Actions



6. Wait for the highlighted cell to switch spots

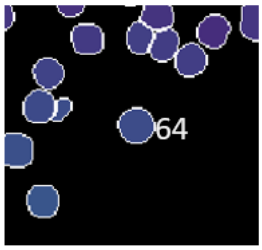


7. (Optional) If cells in multiple frames need to be changed, press “D” to go to the next frame and click “Swap” for each frame
8. Press “Esc” to unselect the 2nd label



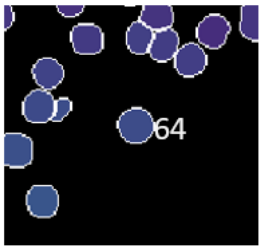
Add new cell value

Use these instructions to assign a new label value to cells that change labels multiple times between multiple cells during the course of the movie. **Use this function only when other functions will introduce a duplicate label in the same frame**.

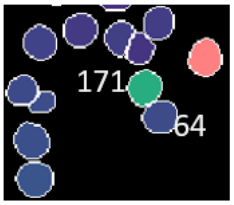


Frame 6

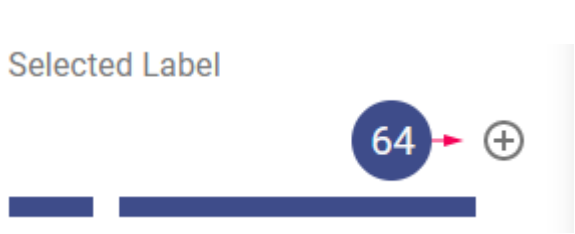
Frame 7

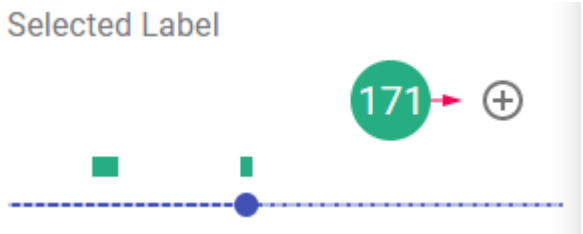


Frame 9



Frame 19

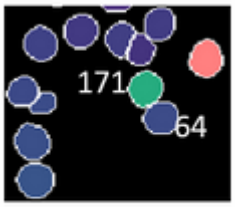




Example: Label 64 changes to 171 in frame 7 then changes back to 64 in frame 9. We cannot just combine label 64 with 171 because they both exist in frame 19. Label 171 in frame 19 will need to be changed to a new label before other fixes can occur.

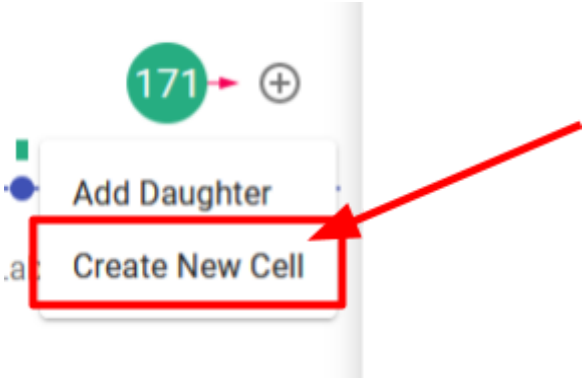
Select the label that needs to be fixed

1. Advance the movie to the frame where that label reappears in the movie

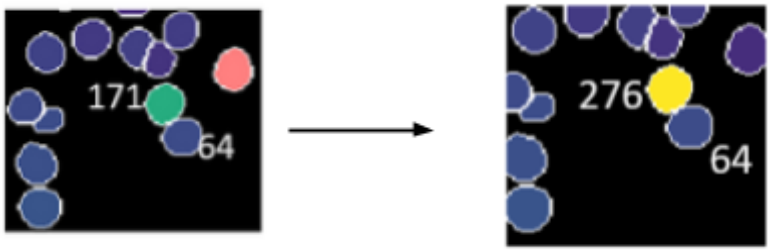


Frame 19

2. Click the “+” symbol next to the label in the lineage map and select “Create New Label”



3. Wait for the label to change



The new label replaces the original label from the current frame to the end of the movie. Then go back and make necessary corrections.

If DCL doesn't work properly

First, make sure you have clicked inside the viewing window, which may solve your issue. This is a new tool that, despite testing, may still have some bugs. The tool may also have been updated since the last time you accessed it. If it does not seem to load correctly, try clearing your browser cache and refreshing the page. If this does not work and you encounter an error that prevents you from finishing correcting the file, please notify your manager and continue onto another file if possible. If you have any suggestions or questions, please share them with us! We would love to know what will make this tool easier and more useful for you.