POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

FasterCellpose: Knowledge Distillation for Efficient Fluorescence Nuclear Segmentation

Supervisors

Prof. Santa DI CATALDO

Prof. Francesco PONZIO

Prof. Xavier DESCOMBES

Candidate

Ivan MAGISTRO CONTENTA

07 2025

Abstract

Endocrine-disrupting chemicals (EDCs) are pollutants that interfere with hormone-receptor binding. In particular, agricultural pesticides have been linked to prostate cancer. To study their effects and the potential treatments, biologists use 3D in vitro organoids, captured by confocal microscopy with nuclear biomarker.
Cell segmentation models are employed by biologists to produce a map that identifies the objects inside the organoid. The label masks simplify experts' analysis. Although effective, the current models are computationally costly. Many laboratories can not afford high-performance GPUs. Moreover, labeled datasets are scarce and manual annotation is time-consuming.
This thesis aims to achieve accurate cell 3D segmentation with a compact neural network on CPU-only hardware. For this purpose, a lightweight version of the well-known Cellpose is presented: FasterCellpose. The baseline architecture was simplified by using more efficient convolutions, removing unnecessary layers and halving the number of channels per depth scale. The resulting CNN model is 56× smaller and 8.4× faster than Cellpose for 3D segmentation on CPU.
To preserve baseline performance, knowledge-distillation was employed to transfer pre-trained Cellpose "teacher" knowledge to the compact "student".
This work highlights the potential of model compression for both 2D and 3D cell segmentation. Furthermore, FasterCellpose can generalize like the baseline on external datasets.
These results pave the way for resource-efficient, high-quality segmentation on 3D confocal images of organoids.

Summary

Cancer can be caused not only by natural factors, but also by industrial compounds, called Endocrine Disrupting Chemicals (EDCs). These pollutants can interfere with the action of natural hormones by binding to their receptors. Agricultural pesticides are a class of EDCs linked to prostate cancer.
To investigate EDC effects on prostate and the potential treatments, biologists need a model that faithfully reflects human tissue. Conventional animal models and human cell lines fail to recapitulate EDC toxicity in human prostate tissue. To overcome these limitations, the researchers have turned to a cutting-edge technology, called organoid—3D structure derived from stem cells that closely mimics the behaviour of real organs.
Confocal microscopy acquires multi-channel fluorescence images of organoids, with each channel detecting a different biomarker. This study focuses on the nuclear biomarker: nuclei appear as ellipsoidal objects, and the whole set and their spatial arrangement distinguish specific phenotypes.

AI models are used to speed up the analysis of confocal images of organoids. For instance, they can produce segmentation masks that identify nuclei inside the sample. One well-known state-of-the-art segmentation model is Cellpose. Segmentation masks can be useful for further analysis, like phenotype classification, growth supervision, etc.
Although effective, state-of-the-art segmentation models are usually computationally onerous. They are time-consuming, especially if users can not afford high-performance GPUs.
Another problem is the lack of labeled 3D datasets; in fact, annotation of images is time-consuming, and online there are few labeled datasets for 2D acquisitions. Moreover, existing methods rely solely on fully supervised training.

This thesis aims to develop a lightweight version of Cellpose that can perform 3D segmentation efficiently on CPU, while maintaining baseline accuracy.

Cellpose is a state-of-the-art model based on a U-Net like architecture, designed for biological cell segmentation: it segments well both 2D and 3D images by computing cell probability scores and flow vectors. Cellpose offers a collection of pre-trained models on different cell types.

FastCellpose is a compact variant with a halved number of feature map channels per depth scale; it is specialized in whole kidney glomeruli segmentation.
In this thesis, we present FasterCellpose, a lightweight model obtained by re-designing the baseline architecture:

1. We reduced the number of convolutional layers in both the encoder and decoder
2. We introduced dilated convolutions in the encoder and transposed convolutions in the decoder
3. We halved the number of channels per depth scale (as in FastCellpose)
4. We replaced standard convolutions with depthwise separable ones

After streamlining the backbone, we applied offline knowledge distillation: this approach enables the transfer of expertise from the pre-trained Cellpose model (teacher) to FasterCellpose (student). The objective is to minimize the distillation loss function by comparing teacher and student's predictions. We worked on an unlabeled dataset of 2D patches, extracted from 3D fluorescence images of prostate organoids at different magnifications (20× and 40×). This process enabled us to obtain effective weights for each compact model.

As a result, FasterCellpose is 56× smaller than the baseline, in terms of the number of parameters and of Multiply-Accumulate Computations. On the CPU, it is 8.4× faster than the baseline for 3D segmentation, reducing network inference time on a 200 slices 1024 × 1024 image stack from about 90 minutes to around 10 minutes.
Furthermore, this model matches baseline performance on our datasets and it is able to generalize on 2D benchmarks, such as DAPI and BitDepth datasets, containing fluorescence images with nuclear biomarker (Table 1).

Model	# Parameters (K)	net. inference time (s)	F1 score (%) (our)	F1 score (%) (DAPI)	F1 score (%) (BitDepth)
Cellpose	6,600	14.12	92.10	93.07	83.81
Distilled FastCellpose	564	4.47	93.01	81.27	87.38
FasterCellpose v3	115	3.30	92.07	81.60	87.17

Table 1: Comparison of Cellpose and lightweight variants in terms of performance metrics on 2D slices

In conclusion, this work demonstrates the potential of neural network compression techniques, such as lightweight architecture design and knowledge distillation, for efficient and accurate nuclei segmentation on 2D and 3D organoid images.
These results contribute to making advanced image analysis tools more accessible to biological researchers working with limited computational resources.

Acknowledgements

I would like to express my gratitude to all who contributed to the completion of my thesis. First and foremost, I thank my supervisors for their guidance and support throughout the thesis, namely Prof. Santa Di Cataldo, Prof. Francesco Ponzio, and Prof. Xavier Descombes. I also extend my thanks to the entire MORPHEME team at Inria Sophia-Antipolis for their collaboration and encouragement. I am especially grateful to have played a role in the Morpheus project, "Organoid Phenotypes Mapping and Modeling: Toward an Endocrine Disruptors Classification", which formed the backbone of this work, and I thank ANR for their financial support (ANR-21-CE45-0029).

A special thanks goes to my family, whose support made it possible for me to study at Politecnico di Torino and to have an important internship in France. They were, and continue to be, by my side. I hope to have made them proud.

Last but not least, I would like to thank my friends and my colleagues. They helped me through this challenging journey, and we shared many enjoyable moments.

Grazie Torino, Merci Sophia-Antipolis

Table of Contents

List of Tables

List of Figures

Acronyms

AI	Artificial Intelligence
BCE	Binary Cross-Entropy
CNN	Convolutional Neural Network
CP	Cellpose
CPU	Central Processing Unit
EDC	Endocrine Disrupting Chemical
FLOP	FLoating Point Operation
GNN	Graph Neural Network
GPU	Graphics Processing Unit
GT	Ground Truth
MAC	Multiply-Accumulate Computation
MAE	Mean Absolute Error
MSE	Mean Squared Error

Chapter 1

Introduction

In this chapter, we first review the Endocrine Disrupting Chemicals (EDCs) and their role in prostate cancer. Then, we introduce organoids, 3D in vitro models adopted by biologists to study human tissue: we present the motivations that led the biologists to use this model, and the effects of the pollutants on prostate organoids. We cover also the acquisition of organoids via specialized fluorescence microscopy. In particular, we underline the challenges with high-resolution 3D images. Finally, we define the main goals of this work.

1.1 Endocrine Disrupting Chemicals

Endocrine Disrupting Chemicals (EDCs) are a class of pollutants that pose various endocrine-disease risks. These contaminants block the interaction between natural hormones and their receptors [1]. They enter the body via inhalation or ingestion and interfere with the endocrine system. These industrial compounds can affect individuals throughout their lives and also future generations.
The main EDC classes are:

1. *Bisphenol A (BPA)*: used for manufacturing, toys, and food-contact materials (canned food and beverages) [2].
2. *Phthalates*: liquid plasticizers present in personal care products and medical tubing [2].
3. *Persistent Organic Pollutants (POPs)*: organic compounds that remain in the environment for extended periods. This category includes pesticides, synthetic chemicals, and industrial process residues [2, 3, 4].

All of these compounds are stored in body fat. While BPA and phthalates have a relatively low accumulation in adipose tissue, POPs have a crucial role in bioaccumulating in the food chain with significant health and environmental consequences

[2, 3]. Among POPs, there are insecticides, such as DDT and its metabolites, DDE and DDD. These organochlorine pesticides act as estrogens and may disrupt the normal development or function of reproductive organs [4].

1.1.1 EDCs on prostate cancer

Several epidemiological studies have highlighted the impact of EDCs on human health, such as the reproductive system. In particular, the POPs may cause diseases in the male reproductive gland, the prostate. In the USA, *Agricultural Health Study (AHS)* reported that certain organophosphate and organochlorine exposures in the agricultural population are associated with prostate cancer risk [2].

1.2 Organoids

Historically, the biomedical research has relied heavily on *animal models*. Although these in vivo models allow for a deep understanding of several biological processes, they lack human specificity and lead to a deficiency in knowledge of heterogeneous populations [5]. For instance, in both human and rodent models, the onset and development of prostate cancer are influenced by androgen and estrogen levels [2]. But, the experts should be cautious about mice models and in inferring causality in humans, since spontaneous prostate cancer is much rarer in mice than in men. Furthermore, *EU REACH regulation* promotes alternative non-animal methods to evaluate the hazards posed by chemical substances [2].
Consequently, researchers developed alternative models for these studies, such as *human cell lines*: these are a population of living cells growing in laboratories for extended periods, used for in-vitro research and cultivation of vaccines. Although convenient, they are often genetically unstable, ethically controversial, tumorigenic, and difficult to grow or control [6].
In recent decades, *organoids* have emerged, offering a compromise between 2D cell lines and living animals [7].

1.2.1 Definition

An *organoid* is a three-dimensional in vitro model that reproduces a human organ or tissue during growth, homeostasis, and disease, under controlled cultured conditions. Organoids are derived from primary tissue samples or stem cells (adult, embryonic, or induced pluripotent) [5]. These structures develop within a 3D extracellular matrix (e.g. matrigel or hydrogel), which provides growth-promoting factors [8]. The composition of the medium varies with the donor tissue [9]. This configuration preserves in vivo physiology and genetic diversity better than existing two-dimensional cell lines. However, organoids represent only an approximation of the donor organ or tissue, as they lack in vivo features, such as a defined body axis or a functional system. In that sense, they are considered complementary to animal models and human cell lines [5]. Another obstacle is the reliable growth of organoids, given to the limited availability of human tissue for the culture initiation [5]. The main applications of these models are tissue engineering, drug discovery, and regenerative medicine. Some organoids preserve characteristics of embryonic tissues that endow them with regenerative capacity. When implanted in host tissue, they can also undergo further maturation toward a more adult phenotype [9]. In addition, patient-derived organoids enable personalized drug screening to identify therapies and drugs that work uniquely on that subject [5].

1.2.2 EDC effects on prostate organoids growth

Androgen levels drive prostate organoids growth [7], such as testosterone and dihydrotestosterone (DHT). During the proliferation, luminal epithelial cells are responsible for the secretion of the prostatic fluid and have a crucial role. EDCs can alter organoid growth in terms of morphometric properties (shape and size) and fluid secretion. Disruptors have a significant impact on the androgen and estrogen signaling pathways. Agricultural pesticides are typically organochlorinated compounds, like DDT (Dichlorodiphenyltrichloroethane) and its metabolite DDE (DHT antagonist).
At elevated DHT concentrations, organoids first undergo massive cell proliferation. The final shape consists of two layers of nuclei, lumen, and basal layers, and prostatic fluid inside that puts pressure on these "barriers". This phenotype is called *cystic*. However, proliferation could be influenced by organochlorinate concentration.

- High DDE concentrations lead to the interruption of differentiation and hinders the formation of the cyst. *Compact* organoids are dense aggregates of nuclei without fluid inside.
- An overproliferation and uncontrolled cell growth produces organoids with irregular and nodular surfaces, whose phenotype is called *cauliflower*.

The detection of these morphological changes is not sufficient to characterize the biological events that cause them, making careful temporal sequences analysis of the organoid growth necessary.

1.3 Image acquisition and microscopes

The power of organoids is their three-dimensional structure. This architecture allows biologists to study structural complexity and its main features, such as the phenotype, morphological characteristics, and development. 3D imaging is superior

to traditional 2D tissue section imaging in organoid visualization [10].
The traditional microscopy is ideal for visualizing a thin section of a tissue [11]. Its lower image quality is due to the uniform illumination on the specimen and to the "noise" caused by the reflection of other parts of the sample.

Compared to light microscopy, *confocal microscopy* captures thick tissues. Confocal images accurately reflect the in vivo cellular composition of organoids [12]. This microscope consists of pinholes, objective lenses, and low-noise detectors [13]. The image quality is superior and provides more contrast than conventional microscopy. The acquisition technique allows virtual three-dimensional image reconstruction: each optical section is taken by moving axially through the focal region and is a few micrometers deep [14].
During acquisition, the excitation beam is focused through the objective into a small spot inside the tissue [11]. The excitation light is reflected off a dichroic mirror rather than directly illuminating the organoid. A laser tuned to a specific wavelength is used to induce fluorescence in a specific fluorophore, thereby revealing its marker. To overcome the reflected fluorescence light issue, the detection pinhole blocks any emission not originating from the focal plane.
As a concrete example, the *nuclear biomarker* [15] is tagged with fluorescent in situ hybridization. When the laser excites a specific fluorophore at a probe wavelength (i.e., the FISH probe), the resultant emission fluorescence will be at a longer wavelength. These fluorescence signals can be collected in sharp optical sections to map nuclear biomarkers in three dimensions.

There are confocal microscopes that are more suited for live imaging, called *spinning disk confocal microscopy.* Instead of having a single pinhole, they consist of a metal disk, called Nipkow disk, with hundreds of pinholes arranged in spirals [11]. It rotates at high speeds to scan every part of the image. From each point the light is electrically transmitted and then reassembled remotely through a second disk. The cameras on spinning disk microscopy must have high framerates due to the speed of the spinning disk ("what is spinning disk confocal microscopy?", Teledyne).
The advantages of spinning disk confocal microscopes are the imaging speed, relatively low-light dose, and the fact that the sample does not have to be moved through the illumination [13].

1.3.1 Issues with 3D images

Despite their value, 3D confocal images introduce practical challenges. First, the annotation of 3D volumes is time-consuming. In practice, expert biologists often label 2D slices rather than full 3D volumes; for example, Data Science Bowl Challenge 2018 dataset consists of a labeled collection of 2D fluorescence images with nuclear biomarker [16]. This drawback prevents the application of supervised learning techniques. A consequence of the scarcity of annotated 3D datasets is the lack of pre-trained models on 3D confocal images. The dimensionality challenge affects also the efficiency of 3D segmentation. Often, a confocal stack contains around 200 slices of 1024×1024 pixels each with a resolution of 8-bit or 16-bit. Therefore, it can occupy a large amount of memory and its prediction can be resource-demanding and time-consuming. Because many users do not have access to a Graphics Processing Unit (GPU), CPU-based segmentation can exceed one hour per 3D volume.

1.4 Goals

The thesis aims to develop a lightweight model that performs efficiently both 2D and 3D segmentation and matches state-of-the-art accuracy on our unlabeled organoid dataset. Starting from the baseline architecture, we derive a compact version applying neural network compression strategies (e.g., pruning, quantization, lightweight model design). The objective is to reduce the number of parameters and MACs by a factor of at least 15. Since 3D segmentation is very expensive, we focus on shortening the inference time on Central Processing Unit (CPU).
Furthermore, to overcome the scarcity of biomedical data, we employ a teacher-supervised distillation approach: the baseline network predictions serve as pseudo-labels for training the compressed model on unlabeled images.
Then, we test the distilled lightweight model and the baseline on our dataset, comparing the quality of their predictions in terms of Precision, Recall, and F1 score.
Finally, we evaluate how well the distilled model generalizes beyond our organoid dataset by comparing its predictions and baseline ones on some benchmarks, such as DAPI and BitDepth datasets.

Additionally, an important goal is to use the distilled compact model to speed up applications that include the baseline. Because segmentation is the primary bottleneck in these workflows, streamlining the model is crucial. For instance, organoids can be modeled as graphs and Graph Neural Networks (GNNs) can use cell segmentation masks to cluster and classify organoid phenotypes.

Chapter 2

Related works

In this chapter, we first describe the instance segmentation task, which is performed by several state-of-the-art models to identify different instances of the target objects. For this task, we consider two well-known state-of-the-art models, StarDist [17] and Cellpose [18], U-Net like architectures presented in chronological order. In a dedicated section, both strengths and weaknesses will be analyzed, from workflow to the presence of pre-trained models.
However, these baselines demand powerful hardware to perform the task. We then focus on neural network compression techniques to make them lighter. Each compression method can improve the model architecture in terms of accuracy, inference time, and/or development cost.
Finally, a section is dedicated to a compressed version of Cellpose released in November 2023, called FastCellpose [19]: it has 12× fewer parameters than Cellpose and its optimization speeds up by 2.3 times the network inference time for 3D segmentation.

2.1 Instance segmentation

Instance segmentation combines the requests of both object detection and semantic segmentation, by localizing the objects coordinates, labeling their pixels and distinguishing the several instances. Due to the dependency to these two tasks, instance segmentation has to overcome their issues to obtain good predictions and to be efficient. The main obstacles are the presence of small objects, the geometric transformations, the image occlusions and degradations [20, 21].

2.1.1 2D instance segmentation

Instance segmentation is mainly employed on two-dimensional images. The proposed approaches combine the architectures and methodologies of object detection and semantic segmentation.
At the beginning, the approaches were based on a component that finds blobby image regions, containing some objects and features. The aim of bottom-up mask proposals approaches is to generate mask proposals and then to classify them; Selective Search [22] identifies quickly image regions that might be good bounded boxes.
With the advent of CNN, many instance segmentation models were released. The main CNN approaches can be divided into three categories: multi-stage, two-stage and single-stage method [21].
In particular,

- *Multi-stage methods* perform instance segmentation by fusing output from different sub-tasks or modules. They are not suited for real-time instance segmentation.
- *Two-stage methods* perform image detection and segmentation in sequence. The order of these two operations defines a particular approach:
  - Top-down approaches, known also as detection-followed-by-segmentation, first draw the bounding boxes and then generate the instance mask. Although a simple and robust solution, it strongly depends on object detection outcome [21].
    MASK-RCNN [23] is an example.
  - Bottom-up approaches, known also as labelling-pixels-followed-by-clustering, first segment the image and then cluster the pixels into individual instances. They are based on a powerful semantic segmentation backbone. However, these methods do not generalize well, especially with occlusions and touching objects [21].
    An example is represented by U-Net [24] with Watershed [25] post-processing technique.
- *Single-stage methods* perform detection and segmentation in a single architecture. They are computationally lighter than two-stage methods by removing pre-processing steps and using a lightweight backbone. There are two main approaches that differ in how central points are drawn to design the corresponding bounding boxes.
  - Anchor-based methods produce bounding boxes around fixed center points placed densely on a regular grid across feature maps [21, 26].
    InstanceFCN [27] and YOLACT [28] are two examples.

  - Anchor-free methods determine object coordinates in a precise way by extracting features from the input and by identifying feature points [21, 26]. There were released several FCOS [29] extensions.

2.1.2 3D instance segmentation

Instance segmentation is applied not only on 2D planes, but also on 3D volumes. 3D data contains more semantic information than 2D one in terms of depth and surface topology. However, deep learning methods still encounter many challenges. Three-dimensional instance segmentation deals with different types of data, from RGB images with depth information to meshes. The focus of our study is on voxels, volumetric pixels of three-dimensional objects used for bioimaging. In our case, since 3D confocal images are stacks of 2D slices, a voxel is identified as a pixel of a 2D slice at a certain depth [30].
The 3D instance segmentation is split in two methods:

- *Proposal-based methods* first generate object proposals and then adjust them to produce the final instance masks. These techniques can be further divided in
  - Detection-based: 3D bounding box regression
  - Detection-free: predicting proposals and evaluating them on confidence scores
- *Proposal-free methods* label each point and then group feature embeddings to obtain definitive 3D instance labels.

These approaches match respectively anchor-based and anchor-free approaches treated in 2D instance segmentation.
The models for 3D instance segmentation are usually based on 3D U-Net [31].

2.1.3 2.5D instance segmentation

However, 3D CNNs are computationally expensive, characterized by long inference times. Moreover, they have a large number of parameters that make them prone to overfitting on small biomedical datasets and highly demanding of GPU resources [32].
2.5D segmentation methods represent a promising solution to bridge the gap between 2D and 3D instance segmentation task. 2.5D images are stacks of 2D slices: XY plane has high-resolution, while z-axis shows a lower resolution.
The key benefit of 2.5D instance segmentation is the use of 2D models across the three dimensions. This means that a reduction of parameter count and of inference

time to perform the same task on the same three-dimensional images, with respect of 3D models [32, 33]. These models guarantee also an adaptation to images with varying z axis, due to anisotropy.
However, they can fail to capture inter-slices relationships and contextual information and are prone to discontinuity in 3D volumes. The improvement of performance between 2.5D CNNs and 3D ones has not been qualitatively evaluated [32].
The main categories of 2.5D instance segmentation models are three [32].

- *Multi-view fusion*: it consists of predicting each slice with 2D model from different planes and then fusing the segmentation results.
- *Incorporating inter-slice information*: it deals with anisotropic voxels, so with images with high-resolution on XY plane, but with lower information on z axis. It incorporates inter-slice information during the segmentation of 3D image with 2D models.
- *Fusing 2D/3D features*: it combines both 2D and 3D segmentation to obtain respectively the mask for each slice and the spatial information based on 2D predictions.

2.2 State-of-the-art segmentation models

The instance segmentation task [20] has a crucial role in bioimaging. The study of cell morphology is important to understand the state of the cells and their response to certain diseases and related treatments [34].
With the development and advancement of deep learning, several neural networks were developed. Our focus is on neural networks developed for biomedical images, especially for fluorescence images. The most well-known methods were tested mainly for the Data Science Bowl Challenge 2018 [16] on nucleus detection [35, 34]. Our interest is not only in 2D images (subsection 2.1.1), but also in 3D volumes. In the last case, 3D and hybrid 2.5D approaches (subsection 2.1.2 and subsection 2.1.3 respectively) obtain good results on volumetric data.
Among these, the most relevant cell segmentation models are StarDist [36] and Cellpose [18] for their broad adoption and user-friendliness. We listed the publications of these models in chronological order.

2.2.1 StarDist

StarDist [36] is a single-stage method that predicts cell nuclei in fluorescence images through *star-shaped convex polygons*, rather than axis-aligned bounding boxes. In this approach, for each pixel, the Euclidean distances to the object boundary are calculated along a set of predefined radial directions. Star-convex polygons

accurately represent spheroidal shapes. This approach is useful for overcoming the crowded-cells issue. Non-maximum suppression (NMS) is performed to prune redundant and low-confidence shapes for each object and to get the final set of polygons. StarDist separately assigns to each pixel a probability of being part of an object or the background.
It is a U-Net like architecture with an additional 3×3 convolutional layer with 128 channels. The two consecutive output layers represent respectively the object probability output and the polygon distance output. The last one has several channels equal to the number of radial directions. The suggested number of radial vectors is 32.
StarDist has a 3D version [17] whose architecture is slightly different from the 2D version. The three-dimensional version is a proposal-free method. There are two versions of the backbone.

- The 3D U-Net backbone [31] has two depth scales. Each block has two convolutional layers with kernel size 3×3×3 per depth. There is another block before upsampling.

- The 3D ResNet backbone [37, 38] starts with a 7×7×7 convolution followed by a 3×3×3 convolution. Then, the architecture presents several residual blocks, each of these containing three 3×3×3 convolutions, residual connections and a downsampling layer. After the backbone, the feature maps are upsampled in the last two output heads.

As for 2D version, the output is composed by one channel of cell pixel probabilities (using the sigmoid function) and as many channels as there are radial vectors.
For this extension, the challenges are the computation of the intersection of two star-convex polyhedra and the number of radial directions. A sufficient number of radial vectors to describe the polyhedron is 64.
During training, we minimize a loss function (averaged over all pixels) containing the following two terms:

- Binary Cross-Entropy (BCE) loss for the predicted object probabilities.

- Mean Absolute Error (MAE) on the radial distances, weighted by the object probability.

A weighted parameter λ_d balances the two loss terms. The distance loss term contains a regularization term for background pixels; the scaling parameter λ_{reg} deals with both object and background distances.
Let p and $\hat{p}$ be the predicted and actual object probabilities, and let d and $\hat{d}$ be

the predicted and actual radial distances. The loss function is defined as follows:

$$\begin{cases} L = L_{\text{obj}} + \lambda_d L_{\text{dist}} \\ L_{\text{obj}}(p, \hat{p}) = -(1-p) \cdot \log(1-\hat{p}) - p \cdot \log(\hat{p}) \\ L_{\text{dist}}(p, d, \hat{d}) = p \cdot \mathbb{1}_{p>0} \cdot \dfrac{1}{n} \sum_k |d_k - \hat{d}_k| + \lambda_{\text{reg}} \cdot \mathbb{1}_{p=0} \cdot \dfrac{1}{n} \sum_k |\hat{d}_k| \end{cases}$$

The formulation is identical for both 2D and 3D versions.

The 2D model is trained on synthetic (TRAgen [39], TOY) and real (Data Science Bowl Challenge 2018 [16]) images.
The 3D model uses WORM [40] and PARHYALE [41] volumes. This demo pre-trained model is compatible with the 3D U-Net backbone.

2.2.2 Cellpose

Cellpose [18] is a U-Net-like model capable of segmenting a large number of cell images.
Conceptually, given a two-channel input (cytoplasm and nuclei), Cellpose operates in two stages:

1. The neural network computes, for each pixel, a cell probability score and a flow field pointing toward the center of the cell. The latter represents the vertical and horizontal spatial gradients derived via a heat diffusion simulation.
2. The predicted cell probability map and flow field are then used to reconstruct the instance segmentation mask. A dynamical system is run for n iterations (by default 200) to cluster pixels into distinct instances by grouping those whose flow vectors converge to the same point.

Cellpose has an encoding and a decoding part, each of these consists of four spatial scales. However, this architecture has different features:

- The direct summation replaced feature concatenation in the upsample pass to reduce parameter count
- The residual blocks were employed instead of U-net building blocks
- The average pooling layers instead of max pooling ones

A spatial block consists of two residual blocks, each containing two 3×3 convolutions interleaved with non-linearity layers. There are skip connections from encoder to decoder at the same spatial scale.
The last encoding spatial block computes not only the feature map for the first decoding unit, but also a vector containing the style of the input image and global

image characteristics. This style embedding has 256 channels and it is obtained as a global average pooling over the final feature map of the downsampling path [18, 42]. This feature map is first normalized, then added to the last three convolutional layers of each decoding unit.
The output head consists of a residual block with 1×1 convolution that computes the outcome [42].

Model hyper-parameters are the diameter, the flow threshold, and the cell probability threshold. While the last two affect the quality of predicted masks, the diameter represents the mean size of the target objects. The network rescales images based on both the user-provided diameter and the one learned by the model.

The loss function is defined as the sum of two terms [18, 42]:

- Mean Squared Error (MSE) over the horizontal and vertical flow fields

- Binary Cross-Entropy (BCE) loss between the predicted cell probability map and the ground truth (GT)

Let C denote the Cellpose output, where C_0 and C_1 are, respectively, horizontal and vertical flows, and C_2 is the class probability map. Concerning GT values, let H and W be the vertical and horizontal flows, and let P be the binary mask. The loss function can then be written as

$$\begin{cases} L = L_{\text{flows}} + L_{\text{prob}} \\ L_{\text{flows}}(C, H, W) = \dfrac{\|C_0 - \lambda H\|_2^2 + \|C_1 - \lambda W\|_2^2}{2} \\ L_{\text{prob}}(C, P) = -(1-P) \cdot \log(1 - \sigma(C_2)) - P \cdot \log(\sigma(C_2)) \end{cases}$$

where $\sigma(\cdot)$ is the sigmoid function. The parameter λ rescales the normalized ground-truth flow vectors so that they can be directly compared with the predicted vectors.

Cellpose performs well segmenting both 2D and 3D images. The 3D variant reuses the 2D model, so new training or fine-tuning is unnecessary. The segmentation on three-dimensional images works by running Cellpose on 2D planes (XY, ZY, ZX) and by averaging the resulting flows (multi-view fusion, described in subsection 2.1.3). This implies that segmentation on 3D images is repeated three times, i.e. one time for each orientation. Therefore, the inference time is at least 3× longer than on 2D images.

Cellpose contains a collection of several pre-trained models, where each of these can identify specific biomarkers within the image (cytoplasm, nuclei, membranes). The authors periodically retrain the model on the data from the community.

2.2.3 Comparison between Cellpose and StarDist

These state-of-the-art segmentation models differ not only in the approach they use, but also in terms of results. Several studies have compared them, identifying the strengths and weaknesses of each model. Cellpose and StarDist are evaluated in their application on 3D fluorescence images with stained proliferative cell nuclei [43].
As noted in subsection 2.2.2, Cellpose has a collection of pre-trained models, and it does not require further training, making it a useful tool for novel and inexperienced biologists. StarDist has few pre-trained models on 2D images, such as the one from Data Science Bowl Challenge 2018 [16] (subsection 2.2.1). However, there is only a demo pre-trained model on 3D images and it would not be the best choice if users need a ready model. StarDist is the best option if a large and labeled collection of samples is available for a supervised learning session, even though it is time-consuming. Furthermore, these datasets should have highly varied shapes and background intensity specimens. Some tools can reduce StarDist training time on small datasets. Like StarDist, Cellpose relies on supervised training techniques. It also offers a human-in-the-loop for 2D images: at each iteration, the model generates segmentation masks, then the user corrects them and finally it retrains on the updated labels [44].
Cellpose produces better segmentation masks than StarDist and other U-Net like methods because the resulting mask consists of pixel-cell probability and also the gradient of flow, to overcome some minor classification errors. The use of simulated diffusion to predict the flow field is particularly helpful for segmenting irregularly shaped objects, like elongated cells; indeed, star-convex polygon can not identify these objects [42]. While other methods look for the center of the cell and guess the diameter to the outer boundary, Cellpose diffusion algorithm predicts the point where the flow gradient of each pixel converges as the center of the cell. Moreover, it is expected to be more resilient due to image resize on diameter value [42]. Yet, the last one represents a challenge because of different image resolutions. Using a fixed value for cell diameter will lead to varying the accuracy depending on the input resolution.
StarDist should be used when Cellpose predictions are not accurate enough to be used. The latter model performs better when images contain high contrast between objects and background, spaced-out objects with uniform intensity and regular shape. While with high noise, Cellpose could miss several objects and the specialized StarDist can detect blurred cells. StarDist outperforms Cellpose in terms of prediction time, indeed the first is significantly faster than the second. Cellpose runtime is dependent on hardware limitations. Therefore, it can be improved with higher processing power [43].

2.3 Neural Network Compression

Deep neural networks achieve strong performances in their tasks, but it is hard to deploy these models on low-resource devices due to the demands in terms of memory storage and energy consumption [45, 46].
Compression techniques target different aspects of a convolutional neural network's structure and parameters. We can categorize methods based on whether they preserve the original network architecture or remove components such as layers, channels, etc.
In particular, structure-preserving techniques are mainly applied to pre-trained models. In this way, reference network weights can not be lost at all and the task can be still performed. Usually, these techniques do not require any prior knowledge of inter-layer relationships [45]. Instead, other techniques work by changing the topology of the network. In this manner, retraining is required to derive an efficient, compact model.
To compare the neural networks and their compressed versions, the main complexity metrics should be mentioned [47, 48]:

- *number of parameters*: it represents the number of variables that characterize the deep neural network. The learnable parameters can be adjusted during the training process to minimize the loss function. This number describes the computational complexity and memory footprint of the model.
- *FLOPs (FLoating Point Operations)*: it represents the number of floating-point operations. They could be sum, multiplication, subtraction, or division. It is a measure of the computational cost of a neural network.
- *MACs (Multiply-Accumulate Computations)*: it represents the number of couples of addition-multiplication operations. As a rule, we consider 1 MAC = 2 FLOPs.

Let N and M be the number of input and output channels, D_F and D_G be the size of the square input and output feature map, and D_K the size of the kernel.

$$\#Parameters = N \cdot M \cdot D_F \cdot D_F$$

$$\#MACs = N \cdot M \cdot D_G \cdot D_G \cdot D_F \cdot D_F$$

$$\#FLOPs = 2 \cdot MACs$$

Inference time measures the amount of time required to process the input and produce an output. It is related to the number of parameters and the number of computations. This value also varies with hardware and software implementation.

This work focuses more on the number of parameters, MACs, and inference time.

2.3.1 Pruning

Typically, artificial neural networks are dense and over-parametrized. This leads to increased memory footprint and computational cost [49].
Pruning is a technique that removes redundant weights without significant impact on model performance [46, 50]. The objective is to obtain sparse networks that retain generalization ability while reducing storage requirements. There are different kinds of pruning:

- Unstructured: these approaches remove individual model parameters based on heuristics.
- Structured: entire weight structures (filters, layers, etc.) are pruned.

Unstructured pruning can remove a large number of unimportant parameters with a low impact on model accuracy. In contrast, structured pruning often causes a larger performance drop at the same pruning ratio. It is necessary to know well the network before applying structured pruning [46, 50].
The learnable parameters could be removed using various criteria: magnitude, sensitivity, etc. [50]. Among them, the most relevant one is magnitude-based pruning. The weights with smaller absolute values have little influence on the overall performance. Lp-pruning is a technique based on magnitude that computes Lp norm of a weight tensor: the values with smaller norms are more likely to be cut off. L1-based pruning is suited to prune the network without fine-tuning, while L2-based pruning performs well if it is followed by fine-tuning [51].
Furthermore, pruning techniques can be classified as local or global. The difference lies in pruning a subset of layers or the whole network. Setting a prune ratio for the layers to prune could be a limitation to local pruning, leading to sub-optimal sparsity. In contrast, global pruning automatically assigns a prune ratio value for each layer [50].
Pruning is usually applied to a pre-trained model, in order to compare dense model performance with its pruned version. Sometimes, training the subnetwork for several epochs is necessary to regain performance. Alternatively, pruning an initialized model eliminates the cost of pre-training, as the sparse model is trained from scratch [50].

2.3.2 Quantization

Quantization is the process of representing values with a reduced number of bits. These values can be weights, activations, or gradients. This technique reduces inference time for models, decreases power consumption, and saves memory space [46, 52]. If value representation is 32-bit, then it is needed $param.mem = \#params \cdot 32$ bits to store weights [53].

The most common value representations are half-precision for floating point (FP-16) and integer arithmetic (INT-16). The current research focus is on lower-precision integer representations, such as INT-8, INT-4, INT-2, or 1-bit (binarization) formats [52]. However, reducing the number of bits can lead to a drop in performance. A standard approach is to replace floating-point operations with their integer equivalents. A further training session of the quantized model can regain much of the lost accuracy. This process, called quantization-aware training, applies quantization during both forward and backward propagation. By contrast, post-training quantization offers a simpler alternative: it performs quantization and adjusts the weights without any fine-tuning [46]. Alternatively, another compression method includes the quantization as a penalty term in the loss and minimizes it [45]. Consequently, designing quantized model architectures facilitates both training and deployment on resource-constrained devices [52].

2.3.3 Low-Rank Decomposition

Low-rank decomposition simplifies convolutional layers by approximating each full-rank weight matrix in a neural network by a product of two smaller matrices [46, 54]. Typically, this decomposition technique is applied layer-by-layer [46].
The low-rank factorization of the filter tensors eliminates redundancies in the parameters and improves the compression ratio [52]. It can reduce the number of multiply-accumulate operations and potentially accelerate inference. However, in many architectures, it does not speed up the model significantly, since the computational operations of CNN are mainly in the convolution layer.
Singular Value Decomposition (SVD) is the standard method for computing a low-rank factorization of weight matrices in both fully connected and convolutional layers [45].
However, the implementation is not that easy since it involves a computationally expensive operation, such as decomposition [54].

2.3.4 Knowledge Distillation

Knowledge distillation is a strategy to transfer expertise from a large teacher model to a small student model. The small model is trained to mimic the teacher's behaviour. It works by providing the same input to both models and comparing their output to adjust student's weights [46]. Teacher knowledge is transferred to the student using class probabilities as "soft targets" [55]. When ground-truth is available, a standard cross-entropy loss between student predictions and true labels, known as "hard targets", is included in the objective function [56].
There are three main types of teacher-supervised techniques [57]:

- *Response-based knowledge* trains the student to directly mimic the final prediction of the teacher
- *Feature-based knowledge* represents an extension of the response-based approach: it supervises the training of the student by also leveraging the feature maps of the intermediate layers
- *Relation-based approach* is based on the relationships between different layers of student and teacher models and their corresponding feature maps

Knowledge distillation is divided into three main learning schemes [57]:

- *Offline distillation*: a fixed, pre-trained teacher model transfers its knowledge to the student model.
- *Online distillation*: teacher and student learn together.
- *Self-distillation*: the same network is used for both teacher and student. A network teaches itself.

2.3.5 Lightweight Model Design

The *lightweight model design* consists of reducing the computational complexity of the neural network by employing more compact convolution kernels or methods. A CNN architecture can be simplified by introducing efficient convolutional modules or by removing redundant layers. Moreover, beyond classical techniques, an optimal neural network can be obtained using complex mechanisms with feedback loop, like NAS [58, 53, 46].
An overview of the lightweight architectures and their efficient convolutional modules is then presented. Next, the focus is on depthwise separable convolutions, a cheap and effective convolution module that replaces standard convolution.

Overview of lightweight model design techniques

The research pays attention on the simplification of CNN structure to deploy the architecture on lower resource-demanding devices. The design of a lightweight model leads to a reduction in the number of computations and in inference time [58, 46].

Efficient convolution modules The first move is to design a cheap convolutional layers. Obviously, the goal is not only to perform less computations, but also to efficiently compute the input to preserve the most relevant features. Several efficient convolutional modules were published and adopted instead of standard convolutions.

In general, the 1×1 kernel convolution, known as *pointwise convolution*, are really employed. These convolutions reduce the computational complexity, but also work effectively across channels, mixing the information and maintaining the input spatial dimensions. They are $9\times$ cheaper than 3×3 convolutions.
Among the most common cheap neural networks, there is SqueezeNet [59] that reduces by 9 times the number of parameters and decreases the number of channels by replacing 3×3 convolution kernels with a 1×1 convolution kernel. MobileNet [60] splits the standard convolution in two sequential operations, depthwise convolution and pointwise convolution. ShuffleNet [61] replaces group convolution with pointwise convolution and the outcome is combined with channel shuffle [58, 53, 46].
Moreover, it is important to add residual connections or skip connections to these structures to not lose information and even to gain model accuracy [58].

Layer removal The *layer removal* is applied to further streamline the backbone. For example, a study [62] compared functionalities of different layers: the bottom layers contain general information, while the upper ones contain abstract information relevant to a specific dataset and task. Removing upper pre-trained convolutional layers can improve network performance. In this way, it is possible to avoid retraining or fine-tuning. The number of layers to remove before applying transfer learning depends on the nature of the dataset.

Combination of lightweight architecture design and compression techniques The lightweight model design and model compression techniques (i.e., pruning, quantization) can be combined to optimize CNN structure. While the first approach defines a new architecture, the second one refines some details, like removing redundant weights or changing their value representation without losing too much in performance [53]. Both aim to deploy the architecture on low-performing devices.

Future directions The future research directions include automatic design method, like *Neural Architecture Search (NAS)*. It is a system composed by three parts (search space, evaluation method and search method), whose goal is to obtain the optimal neural network satisfying a multi-objective optimization through a feedback loop [58, 46].

Since the convolution operation can be divided in shift and multiplication operations, the focus is also to optimize the *shift operations* to further trim memory footprints [58].

Depthwise Separable Convolution Layers

We now present a detailed overview of *depthwise separable convolutions*, which will be important in the development of lightweight models in the Methodology chapter (subsection 3.3.4).
A standard convolutional layer filters and combines inputs into a new set of outputs in a single step. Its computational cost depends on the size of the input feature map, the number of input and output channels, and the filter sizes. To make convolution operations more efficient, Xception [63] and MobileNet [60] introduced a new module, called depthwise separable convolution layer. In this module, the standard convolution layer is split into a filter step and a combination step. These two operations are performed in sequence.

1. Depthwise convolution applies one filter to each input channel.
2. Pointwise convolution applies a 1×1 convolution to linearly combine the resulting feature maps.

MobileNet applies the non-linearities after each layer [60], whereas in Xception the non-linearity is applied only at the end of the stack of operations [63]. The depth (i.e., number of channels) of the intermediate feature spaces is critical to the usefulness of the non-linearity. In detail, non-linearity is helpful for deep feature spaces, while it may discard information in shallow feature maps [63].

Let D_F be the square input feature map size, N the number of input channels, D_G the square output feature map size, M the number of output channels, and D_K the kernel size. Then, the computational cost of a standard convolution is

$$M \cdot D_K \cdot D_K \cdot N \cdot D_F \cdot D_F$$

whereas the cost of a depthwise separable convolution is

$$N \cdot D_K \cdot D_K \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$$

This leads to a reduction in the computation of

$$\frac{1}{M} + \frac{1}{D_K^2}$$

2.4 FastCellpose

FastCellpose [19] is a lightweight variant of Cellpose specialized in segmenting whole kidney glomeruli. This model was obtained by simplifying Cellpose backbone and optimizing the mask reconstruction.
Concerning the architecture, the authors modified two key parameters: neural

network width and depth. They are respectively the number of base feature maps per depth scale and the number of convolutional layers per unit.
The best number of channels at the first depth scale is 16, then this number doubles when downsampling until reaching 128 channels, then, this value halves during upsampling. By contrast, Cellpose initial number of feature map channels is 32 and the maximum number is 256.
Furthermore, the authors found that each spatial block should contain two convolutional layers instead of four. In the upsample block, the unpooling layer was replaced by a transposed convolution, followed by one standard convolutional layer. In each downsample unit, the pooling layer remains present.
Therefore, the computational complexity was drastically reduced by 92 % in number of learnable parameters.

Since inference time is a sum of network prediction and mask reconstruction times, the authors also optimized the latter. For mask reconstruction optimization, a U-Net-like approach was adopted to operate on the network prediction: first, the final outcome (both cell probability mask and flow field) is downsized. Next, the label mask is computed as in Cellpose. Then, the resulting mask is upsampled back to the original image size. Moreover, the number of gradient flow tracking iterations is reduced from 200 to 50 to further speed up the mask reconstruction process. The authors removed the style branches, as they were designed to simultaneously segment different types of cellular images.

FastCellpose was trained by scratch on renal glomerulus samples, without recurring to a Cellpose pre-trained model. The human-in-the-loop approach was adopted to train the neural network on mouse kidneys of different ages. This technique is a supervised learning approach that integrates human knowledge and experience [64]. This pre-trained model also performs well on neural soma segmentation.

This architecture is designed for multi-threaded and parallel computing. In this way, segmentation speed can be up to 12 $\times$ faster than Cellpose.

Chapter 3

Methodology

This chapter describes the dataset and the workflow employed for training and inference.
First, we introduce our dataset of 3D confocal images of prostate organoids with nuclear biomarker. We also cover data annotation. Next, we select our baseline by comparing StarDist and Cellpose and motivating why one is better suited to our scenario and dataset. Subsequently, we present the lightweight models, describing how they were compressed and conducting a complexity analysis against the baseline to quantify the efficiency gains. Furthermore, we describe how we applied knowledge distillation to match baseline performance. Finally, we perform a comparative analysis of Cellpose and our distilled models by measuring segmentation quality on 2D images against ground-truth annotations.

3.1 Dataset

Our dataset contains confocal images of mice prostate organoids that were sampled in two laboratories, one in Nice and one in Paris. These organoids were mostly acquired with the confocal microscopy, but some of them with Spinning Disk confocal microscopy. Two *objectives* were used to capture the images: 20 $\times$ and 40 $\times$ magnification. There is not a detailed description of the immersion medium under each objective. Confocal microscopy employs light beams on a nuclear fluorophore to highlight this bio-marker in the images. Therefore, the images contain only one channel, the nuclear one. Each organoid was sampled on the 7th day.
The images are divided into three sets; each set represents a specific *phenotype*. As mentioned in the subsection 1.2.2, the phenotypes for prostate organoids are cystic, compact, and cauliflower.
These image stacks are in `.tif` format and they can be opened with Fiji tool [65]. We removed some images because they were blurred and state-of-the-art models

can not predict accurate segmentation mask on them.
This dataset is **unlabeled**. It represents a limitation, as we could not perform supervised training on these images and could not quantitatively evaluate the prediction accuracy relative to the ground-truth. Table 3.1 contains the number of images grouped per laboratory and magnification. Most of the samples belong to the cystic phenotype. There is a low quantity of compact organoids.

Laboratory	Objective	# samples	Cauliflower	Compact	Cystic
Nice	20 ×	104	17	31	56
Paris	20 ×	5	0	0	5
Paris	40 ×	190	106	1	83

Table 3.1: Number of images for each laboratory and objective

We partitioned the data into an 80% training set, and a 20% test set (Table 3.2). The training portion was further split into 67% for model fitting and 33% for validation. After completing the distillation sessions, we added new samples to the test set.
Because samples are unevenly distributed across phenotypic classes and instance segmentation focuses only on assigning image pixels to object instances, we ignored these labels. Instead, we grouped samples by laboratory and imaging objective, accepting their natural distribution.

Laboratory	Objective	# train samples	# test samples
Nice	20 ×	80	24
Paris	20 ×	4	1
Paris	40 ×	104	86

Table 3.2: Training and test set per each objective

For **prediction**, we used both 3D volumes and 2D slices extracted from them.
For **training**, we divided each image into 256 × 256 patches to avoid memory-allocation (Out-Of-Memory) issues during distillation process. We filtered the patches to keep only those containing nuclei and to discard background-only patches. This ensures that segmentation models are tested on their ability to detect nuclei rather than background. We collected training patches by objective, regardless of the originating laboratory. The following table Table 3.3 shows the number of patches for each objective.

Objective	# train patches
20 ×	43,739
40 ×	224,187

Table 3.3: Training patches for each magnification

3.1.1 Annotation

Given 3D samples, we extracted some 2D slices for each dataset (laboratory, objective) and labeled nuclei using multi-point tool of *Fiji* [65]. The resulting annotations were useful as ground-truth for the evaluation of the cell segmentation models. The coordinates of the labeled objects were saved in `.csv` files. Table 3.4 shows how many 2D slices were annotated.

Laboratory	Objective	# slices	Cauliflower	Compact	Cystic
Nice	20 ×	17	5	6	6
Paris	20 ×	1	0	0	1
Paris	40 ×	8	3	0	5

Table 3.4: Number of annotated 2D slices for each laboratory and objective

We annotated only 2D slices, and not full 3D image stacks, because labeling 3D volumes is time-consuming. The motivations of this choice are explained in section 5.5.

3.2 Baseline definition

The comparison between Cellpose and StarDist, reported in subsection 2.2.3, helped us to choose Cellpose as baseline for our work.
The **key advantages of Cellpose** are its single architecture for both 2D and 3D segmentation and its set of pre-trained models (subsection 2.2.2), including the one specialized on fluorescence images with nuclear biomarker (*"nuclei"* model). Cellpose is ideal for non-experienced users due to its ease of use.
By contrast, StarDist requires distinct models for 2D and 3D segmentation, as noted in subsection 2.2.1. StarDist has few pre-trained models on 2D images, but for 3D segmentation there is only an unofficial version trained on 20 samples. Community members have also released models trained on 3D images, such as *StarDist Plant Nuclei 3D ResNet*, but we did not test them. StarDist obtains

accurate predictions after training on a large set of labeled images, but our unlabeled datasets (section 3.1) are not suited for this purpose. Although StarDist is lighter and faster than Cellpose, these benefits did not compensate for its limitations in 3D segmentation.
Finally, we reported a numerical comparison in section 4.3: we computed predictions of both models on 2D slices from our datasets. The results demonstrated that Cellpose outperforms StarDist by at least 5 % in F1 score.
These combined technical, usability, and performance advantages led us to adopt Cellpose as our project baseline.

3.3 Cellpose compression

This project aimed to develop a neural network that achieves baseline performance while keeping low inference costs. Because FastCellpose is currently the lightest variant, we focused on designing an even more compact architecture. On the baseline backbone, we employed different **neural network compression techniques**. These methodologies are complementary since they streamline the architecture at the expense of accuracy, inference time, or development cost. We evaluated the main compression families: we discarded the methods that change weight representation (quantization, subsection 2.3.2) or that remove redundant parameters to produce a moderately sparse network at the cost of accuracy (pruning, subsection 2.3.1). Moreover, we did not consider those that decompose the convolutions, like low-rank factorization (subsection 2.3.3). Therefore, we turned to model design (subsection 2.3.5) and knowledge distillation (subsection 2.3.4). The first reduces the computational complexity of the architecture, although it comes at a performance cost. The second compensates for the loss in accuracy by transferring baseline knowledge to the compact model.
We first redesigned Cellpose via structural changes, then performed knowledge distillation.

We applied the following compression techniques to streamline baseline architecture:

1. Removing redundant layers
2. Replacing standard convolutions with more efficient variants
3. Reducing the number of feature map channels

As noted in subsection 2.2.2, inference time also includes a component due to mask reconstruction. This process can be optimized by adjusting the number of iterations for "follow the flows" and resizing the flow-vector field, as proposed by FastCellpose [19].

3.3.1 Dilated Cellpose

We began compressing the *downsample (encoding) path.*
As we described in subsection 2.2.2, a spatial block consists of four convolutions. The first step was to remove two convolutional layers in each encoding unit, reducing also the number of residual additions to one. After this change, the output feature map has a 5×5 receptive field, smaller than 9×9 one with four 3×3 convolutions. To capture more context of the feature map without adding layers, we introduced dilation in some convolutional layers. The **dilated convolution** (or "**atrous convolution**") expands the *receptive field* without increasing the number of parameters or reducing spatial resolution [66, 67]; dilation inserts spaces between kernel element [67]. Given a kernel size k and a dilation rate d, the *effective kernel size* is

$$\hat{k} = k + (k - 1)(d - 1)$$

Given the chain of two layers, we replaced the second layer with a dilated convolution ($dilation = 2$). We set the dilation rate of the second convolution layer to 2 to increase the receptive-field size. Instead of a 5×5 receptive field, the new configuration provides an output feature map with a 7×7 receptive field. It represents a trade-off between the two convolutions chain (5×5 receptive field) and a four convolutions chain (9×9 receptive field). Furthermore, we adjusted padding so that each dilated convolution preserves feature map size. This aligns with the Hybrid Dilated Convolution (HDC) approach, a stack of convolutions with progressive higher dilation rates [68, 69].
With these changes, we reduced the number of parameters by 23.78% and the number of MACs by 23.85%, compared to the original Cellpose architecture. It is possible to get predictions from this model by loading the weights from a pretrained model. Despite removing two convolutional layers per spatial block in the downsample path and introducing dilation, the model still works, albeit less accurately than the baseline (a drop between 20 % and 40% relative to Cellpose in terms of F1 score).

3.3.2 Upsample Cellpose

We compressed the *upsample (decoding) path* as in the downsample path.
In fact, we removed two convolutions from each block in the decoder.

In Cellpose backbone, upsampling (unpooling) layers are employed to increase the feature map size; these layers use the Nearest-Neighbor approach. The interpolation smoothens the feature maps, especially around edges, and it often misses out on thin details [70].
To better control the feature map restoration, we replaced the unpooling layers with **transposed convolutional layers**, also known as fractionally strided convolutions.

Compared to standard convolutions, the transposed ones work oppositely: they produce an output feature map larger than the input. Moreover, forward and backward operations are swapped [67].
The main disadvantage of transposed convolutions is the introduction of zero-padding in input and of learnable parameters in the neural network (unlike unpooling layers). In transposed convolution layers, overlaps may happen during the computations, due to stride and kernel size hyper-parameters. In particular, the *"uneven overlap"* occurs when the kernel size is not divisible by the stride [71] or when the stride is smaller than the filter size [70]. This unevenness creates checkerboard-like artifacts in the output. To avoid this "grid", we used a kernel size that is divisible by the stride.
We combined upsampling layers and a transposed convolutionals: in this way, we leverage the strengths of both techniques, while mitigating their individual drawbacks.
To double the feature map size, we explored various hyper-parameter configurations. We selected those configurations that had a kernel size divisible by the stride. We chose the one that introduces the lowest number of learnable parameters [70]:

- Kernel size = 2
- Padding = 0
- Dilation = 1
- Stride = 2
- Output padding = 0

Using this setup, we reduced the number of parameters by 22.23% and the number of MACs by 23.22%, compared to the original Cellpose architecture.
The learnable parameters of the new transposed convolution layers were initialized with *Kaiming approach*: it consists of applying a Normal distribution with $\mu = 0$ and $\sigma = \frac{2}{n}$ where n is the number of inputs to the node.
As with Dilated Cellpose (subsection 3.3.1), the pre-trained weights were loaded, but the introduction of new layers with randomly initialized weights prevented acceptable predictions. Therefore, it was necessary to find a new set of weights for this model, avoiding training and fine-tuning steps.

3.3.3 Compressed Cellpose

Based on our previous findings, we applied the compression methods on both *downsample* and *upsample paths.* This further reduced the number of parameters and multiply-accumulate computations.

By removing two convolutional layers per spatial block, introducing dilation and interleaving with unpooling and transposed convolutions, we obtained a model with 46.01% fewer parameters and 46.87% fewer MACs.

3.3.4 Cellpose with depthwise separable convolutions

However, removing convolutional layers alone is not sufficient. Inspired by Xception [63] and MobileNet [60], we replaced all the convolutional layers of Cellpose backbone with **depthwise separable convolutional layers** (section 2.3.5). In this way, each convolutional layer yields a parameter reduction of

$$\frac{1}{M} + \frac{1}{D_K^2}$$

The number of parameters dropped by 81.06% and the number of MACs dropped by 84.11%.

3.3.5 Compressed Cellpose with depthwise separable convolutions

To further streamline the previous version, we simplified the encoder-decoder architecture as done in Compressed Cellpose. Specifically, we removed additional convolutions and some upsampling layers, and introduced dilated and transposed convolutions. In upsampling, we inserted **depthwise separable deconvolutions** [72] instead of transposed convolutions. While depthwise separable convolution consists of depthwise convolution followed by pointwise one, our decoding variant uses transposed convolution for each channel followed by 1×1 convolution. As in Compressed Cellpose, we alternated unpooling layers with depthwise separable transposed convolutions to avoid artifacts and limit the increasing of parameters. We reduced the number of parameters to less than a million. In particular, we removed 89.89% of parameters relative to the original architecture. The number of MACs decreased by a factor of approximately 10.

3.3.6 FasterCellpose

The methodologies we applied were not sufficient to obtain a CNN smaller than FastCellpose (section 2.4).

The authors of FastCellpose presented an operation to further streamline Cellpose backbone: the *number of feature map channels* was halved at each level of the U-Net. Instead of having four depth scales with $32 \rightarrow 64 \rightarrow 128 \rightarrow 256$ output feature map channels (and the reverse in upsampling), the new architecture used four depth scales with $16 \rightarrow 32 \rightarrow 64 \rightarrow 128$ channels. This single action reduces

the computational complexity by approximately 4×.
Therefore, we halved the number of channels and then we resorted to *depthwise separable convolution.* This compact model represents the **first version of FasterCellpose**, surpassing FastCellpose in both parameter count and MACs. FasterCellpose has 95.42% drop in parameters compared to Cellpose. It is 1.73× smaller than FastCellpose.

Based on the compression techniques explored previously, we developed a **second version** that is also lightened in both the downsample and upsample paths. It has 97.15% fewer parameters and 97.19% fewer MACs than Cellpose.

A **third version** was also implemented by using only depthwise separable transposed convolutions instead of the interleaving of these ones and unpooling layers. Compared to the previous version, we gained in terms of computational complexity, but artifacts may occur during upsampling. This version has a reduction of 98.24% in parameter count and 98.5% in computation count than the baseline.

3.3.7 Computational Complexity Analysis

Table 3.5 presents Cellpose, its lightweight models, and FastCellpose in terms of computational metrics. All compact models are compared to the baseline in terms of number of learnable parameters and Multiply-Accumulate Computations (MACs) on a 1024×1024 tensor, since most 2D images have this resolution. Notice that each successive model reduces both metrics relative to the baseline. As shown in Figure 3.1, models follow an approximately linear relationship between the number of learnable parameters and the number of MACs. Rather than applying only compression techniques based on weights (e.g., pruning, quantization), we redesigned the convolutional layers to reduce both parameters and operations [46].

Recalling from section 2.3 that

$$\#Parameters = N \cdot M \cdot D_F \cdot D_F$$

$$\#MACs = N \cdot M \cdot D_G \cdot D_G \cdot D_F \cdot D_F$$

it follows that the relationship between parameters and weights is

$$\#MACs = D_G^2 \cdot \#Parameters$$

From section 2.3.5 we identified a reduction of parameters and computations using depthwise separable convolutions as follows

$$\frac{1}{M} + \frac{1}{D_K^2}$$

Because we removed entire convolutional layers and replaced others with depthwise separable convolutions, both parameter count and MACs decrease approximately

Model	# parameters	params ratio wrt Cellpose (%)	# MACs	MACs ratio wrt Cellpose (%)
Cellpose	6,600,843	100.00	326,837,313,536	100.00
Dilated Cellpose (downsample)	5,031,244	76.22	246,658,998,272	75.47
Upsample Cellpose (upsample)	5,133,164	77.77	250,953,719,808	76.78
Compressed Cellpose (downsample + upsample)	3,563,565	53.99	173,644,308,480	53.13
Cellpose with depthwise separable convolutions	1,250,019	18.94	51,942,629,376	15.89
Compressed Cellpose with depthwise separable convolutions	720,100	10.91	33,923,653,632	10.38
FastCellpose	564,807	8.56	30,102,550,528	9.21
FasterCellpose v1	326,531	4.95	14,984,243,200	4.58
FasterCellpose v2	186,865	2.83	9,187,653,632	2.81
FasterCellpose v3	115,969	1.76	4,912,609,280	1.50

Table 3.5: Comparison of Cellpose, FastCellpose and lightweight models in terms of computational complexity

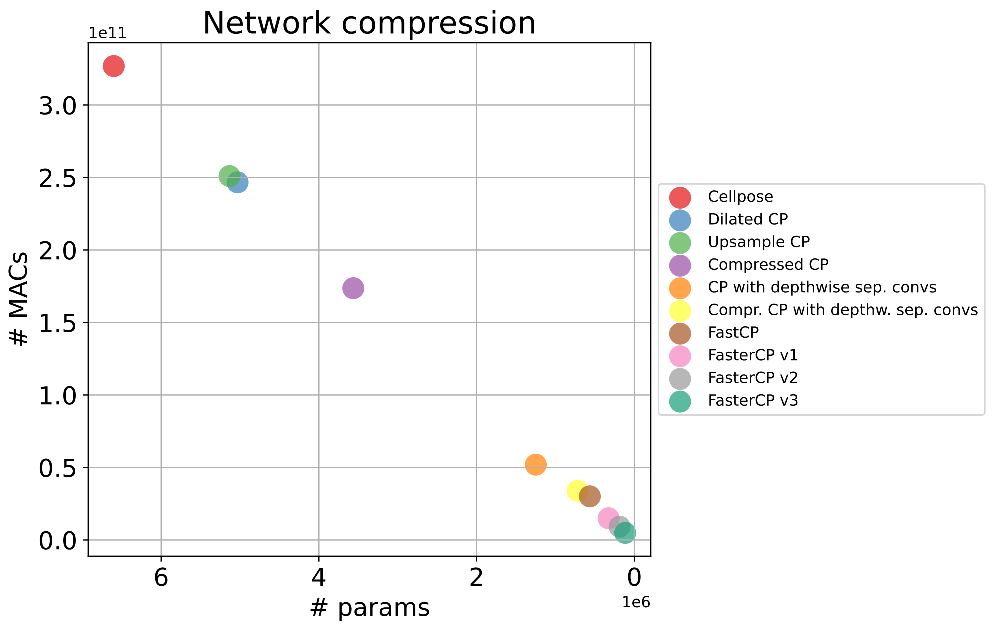


Figure 3.1: Models in terms of number of parameters and MACs

linearly across our model variants. However, Upsample Cellpose is computationally more complex than Dilated Cellpose due to the introduction of transposed convolutions [67].

3.4 Knowledge Distillation on Cellpose

The knowledge distillation stage follows the Cellpose compression stage. As described in subsection 2.3.4, the goal is for the student model to closely mimic the teacher model in performing the instance segmentation task.
The pre-trained Cellpose model on nuclei bio-marker images served as the teacher. We adopted the lightweight models obtained in the previous stage as students. Most of the student models were initialized with Kaiming approach.
Since Cellpose does not natively support teacher-student learning, we implemented a knowledge distillation method inspired by its codebase.

We adopted *response-based knowledge* and *offline approach* to update the weights of all student's layers, using teacher's reliable predictions. As mentioned in section 3.1, we used a dataset of 2D patches for these. Since our dataset is unlabeled, we could use only *soft targets* derived from teacher predictions.
Given the acquisition of fluorescence images at different magnifications, we trained a separate model for each objective.

To evaluate models' output, we were unable to use instance segmentation masks due to compatibility issues with tensors. Instead, we used segmentation masks coming from cell probability maps. However, using segmentation masks may lead to not separate two touching objects.

Metrics We used *Intersection over Union (IoU)* [73, 74] to compare teacher and student cell probability masks. This metric ranges from 0 (no overlap) to 1 (perfect overlap). To obtain binary segmentation masks, we applied a threshold of 0.5 to both probability maps. This value allows to identify pixels most confidently labeled as nuclei. Teacher's binary output serves as pseudo ground-truth. Formally:

$$IoU = \frac{TP}{TP + FP + FN}$$

where

- TP is the number of pixels assigned to nuclei class by both teacher and student
- FP is the number of pixels assigned to nuclei class by the student, but labeled as background class by the teacher
- FN is the number of pixels assigned to background class by the student, but labeled as nuclei by the teacher.

A similar metric is *Dice* [74]:

$$Dice = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

This metric has a specific loss, called *Dice loss*.
In our experiments, we reported only IoU.

Distillation Settings We ran teacher-student learning for each student at least for 100 epochs over all the training samples, using Adam optimizer [75]. At the end of each epoch, IoU was computed on the training dataset. The frequency of IoU computation on the validation set can be user-defined.
We implemented an *early stopping criterion* to terminate the teacher-student training session. It triggers when the validation IoU value decreases by more than 5 points (overfitting) or stagnates for three validation steps (plateau).
Given the large size of our dataset, we did not consider regularization techniques, such as data augmentation, during student training.

Loss definition In subsection 2.2.2, we described Cellpose loss function and we mentioned λ scaling parameter [18, 42]. By default, this parameter is set to 5, as supervised training involves converting label masks into flow representations with values between 0 and 1. In our case, we set λ to 1 because we compared directly the vector fields of teacher and student models.

3.4.1 Hyper-parameter optimization

A crucial hyper-parameter for Cellpose is **diameter**, which is used to rescale the input image before segmentation. Other important hyper-parameters for the knowledge distillation process are **learning rate** and **batch size**. We performed a *K-Fold Cross Validation* by splitting the dataset into four folds. For each fold, we ran knowledge distillation for 15 epochs and collected the validation IoU to determine which hyper-parameter configuration yielded the highest IoU.

3.5 Evaluation

We used *DAccuracy tool* (Eric Débreuve, DAccuracy, 2024, `https://src.koda.cnrs.fr/eric.debreuve/daccuracy`) to evaluate the predictions against the annotations on 2D slices. This tool computes performance metrics, such as Precision, Recall, and F1 score [73].

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Masks are always images, while GT can be either images or `.csv` files containing object coordinates. If GT is provided as a comma-delimited text file, users must specify which columns correspond to row–column (or x–y) coordinates. Furthermore, DAccuracy can display the predictions over the ground truth (GT) for visual inspection.

Each model was evaluated on the corresponding magnification level it was trained for.

The limitation of this tool is represented by the evaluation of 3D volumes, which is extremely time-consuming on CPU. Moreover, we did not annotate 3D images, as noted in subsection 3.1.1. So, no 3D quantitative evaluation was performed.

Chapter 4

Results

This chapter reports both quantitative and qualitative outcomes of Cellpose and distilled lightweight models across various datasets. First, we conduct a comparative analysis of segmentation performance on our data, followed by evaluations on two benchmarks: DAPI [76] and BitDepth [77] datasets. We display some graphs that represent the evaluation of predictions against ground-truths in terms of F1 score and parameter count. Next, we shift our focus to 3D segmentation. The evaluation of 2D masks is straightforward, assessing predictions on 3D volumes is more challenging. Therefore, we also include a qualitative analysis of the masks and we show how inference time drops from Cellpose to FasterCellpose v3.

4.1 Hardware specifications

As mentioned in section 1.4, we aimed to obtain a lightweight model that segments biomedical images efficiently on the CPU.

We performed **knowledge distillation** using a *high-performing GPU*, which enabled faster training on a large number of samples and the use of larger batch sizes (e.g., batch size of 8 samples). The average training time per epoch on 200,000 samples is around 40-50 minutes. Table 4.1 shows specifications for this stage.

Component	**Specification**
CPU	AMD EPYC 7313 @ 3.0GHz
RAM	512 GB
GPU	NVIDIA A100 PCIe (40GB VRAM)
Storage	31.8TB
Operating System	Ubuntu 24.04 LTS (Noble Numbat)

Table 4.1: Hardware specifications used for knowledge distillation

For **prediction**, we used the *CPU* because we would like to emulate a laboratory environment that is not provided with a GPU. Indeed, the inference time is related to the specific hardware: the average inference time is around 15 seconds per 1024×1024 image using Cellpose. Table 4.2 shows CPU specification.

Component	**Specification**
CPU	Intel(R) Xeon(R) E-2186M @ 2.90 GHz
RAM	64 GB
Operating System	Windows 10 Pro 22H2

Table 4.2: Hardware specifications used for inference

4.2 Results from internal K-Fold Cross Validation

As mentioned in subsection 3.4.1, we performed internal K-Fold Cross Validation (K=4, 15 epochs per fold, Adam optimizer [75]) to identify the best hyperparameter configuration (subsection 3.4.1). We ran this on each dataset of patches (section 3.1), using the Cellpose pre-trained model as teacher.

First, we used Dilated Cellpose (subsection 3.3.1) as our student model, since it was the first developed lightweight variant. The Cellpose pre-trained weights were loaded onto this student.

Next, we used FastCellpose (section 2.4) with randomly initialized weights, as our goal was to focus exclusively on nuclear biomarker images rather than glomeruli.

The optimal hyperparameter values for each student model and dataset are summarized in Table 4.3

Student	Dataset	Learning Rate	Batch Size	Diameter	Validation IoU (%)
Dilated CP	20×	10^{-3}	8	25	85.53
Dilated CP	40×	10^{-3}	8	25	84.39
FastCP	20×	10^{-3}	8	25	82.17
FastCP	40×	10^{-3}	8	25	80.82

Table 4.3: Results from K-Fold Cross Validation (K=4; # epochs per fold = 15; # patches = 10,000 per magnification)

To confirm that the best diameter is 25, we computed the optimal diameter size on the patches of different phenotypic classes. As shown in Figure 4.1, this value fits well with most of the patches.

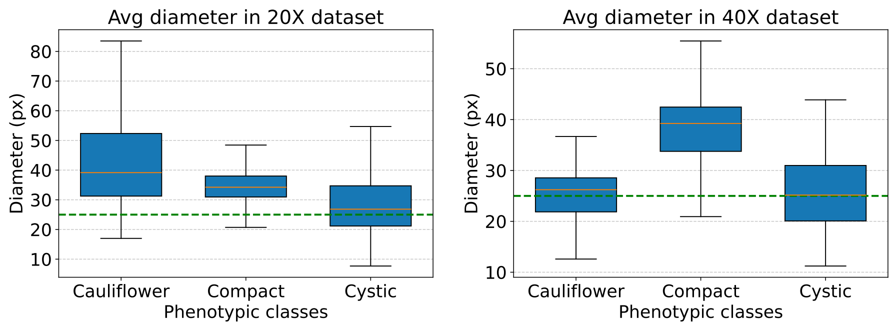


Figure 4.1: Average diameter value for each phenotype and magnification

4.3 Performance of Cellpose and StarDist

Based on the considerations in subsection 2.2.3, we performed a further analysis of Cellpose and StarDist on 2D slices. Using annotated 2D slices at each magnification level, we collected performance statistics of models' predictions against the ground truth. As shown in Figure 4.2, Cellpose outperforms StarDist across all phenotypic classes and both 20× and 40× magnification levels.

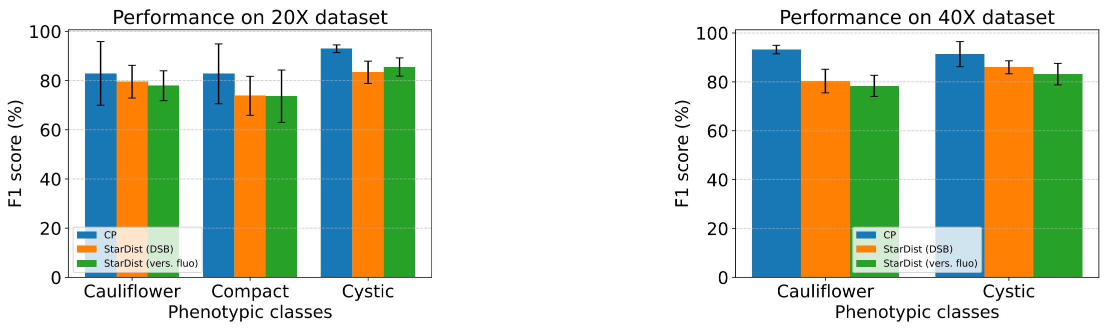
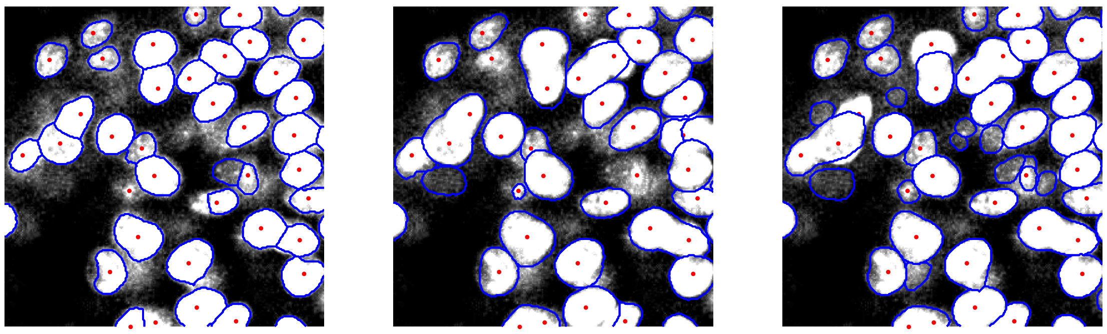


Figure 4.2: Quantitative comparison between Cellpose and StarDist in terms of F1 score on 2D slices per each phenotypic class at 20× and 40× magnifications in terms of F1 score. On fluorescence nuclear-biomarker images, there is one pre-trained model for Cellpose ('nuclei') and two pre-trained models for StarDist (one on solely DSB [16], another also on other images)

In Figure 4.3, we qualitatively compare Cellpose and StarDist segmentation masks. Cellpose produces more accurate detections than StarDist. Figure 4.3b and Figure 4.3c are respectively the masks obtained with StarDist pre-trained model

’DSB’ and ’versatile_fluo’: StarDist captures spheroidal objects, but often these shapes are incorrectly assigned to nuclei, increasing the number of false positives detections. By contrast, in Figure 4.3a Cellpose consistently traces precise contours, and effectively separates well touching objects into distinct instances.



(a) Cellpose **(b)** StarDist (DSB) **(c)** StarDist (fluo)

Figure 4.3: Qualitative comparison between Cellpose and StarDist predictions on a slice. The blue contours represent the predicted object boundaries, the red dots represent the ground-truth nuclei centroids

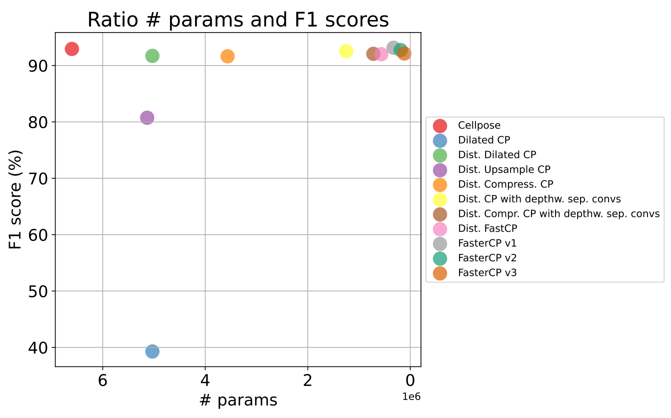
4.4 Performance on our dataset

We compared Cellpose with various lightweight and distilled models to assess whether they can match baseline performance.
Loading pre-trained Cellpose weights into the lightweight architectures resulted in poor segmentation masks, as expected. Layer removal causes a performance drop, and the insertion of new layers (transposed convolutions and depthwise separable convolutions) requires well-tuned weights to yield good segmentation.
We thus applied knowledge distillation to adjust the weights of each model on our training set. We evaluated the distilled models on annotated 2D test slices.
These results confirm that our students learnt effectively from the teacher. We observe in Table 4.4 the results on cystic samples with magnification 40×. Compared to the baseline, Dilated Cellpose had a drop of around 22 % in the F1 score, but after applying knowledge distillation, performance was largely restored. Furthermore, we observed that FasterCP v3 delivers a 3.89× speedup in inference time over the original Cellpose.

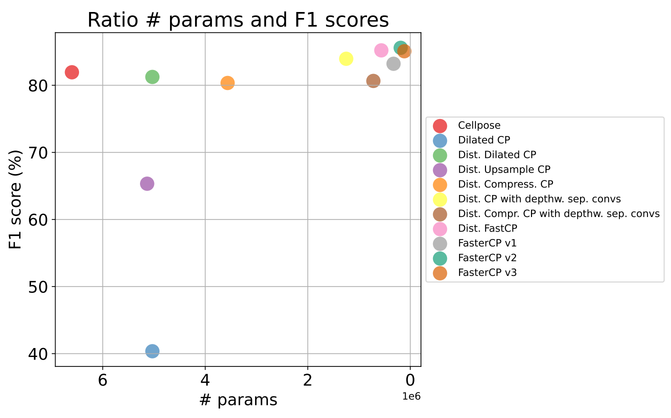
Model	Precision (%)	Recall (%)	F1 score (%)	net. inference time (s)
Cellpose	91.45 ± 7.13	91.37 ± 3.73	91.34 ± 5.13	12.69 ± 0.72
Dilated Cellpose	99.79 ± 0.47	54.02 ± 10.73	69.55 ± 9.48	11.00 ± 0.77
Distill. Dilated Cellpose	89.66 ± 6.50	96.00 ± 1.95	92.60 ± 3.31	10.75 ± 0.72
Distill. Upsample Cellpose	97.68 ± 1.71	43.74 ± 19.71	58.45 ± 18.96	10.48 ± 0.76
Distill. Compressed Cellpose	91.66 ± 3.28	82.55 ± 19.67	85.65 ± 12.80	9.32 ± 0.51
Distill. Cellpose with depthw. sep. convs	93.85 ± 3.40	90.03 ± 5.91	91.80 ± 3.48	7.85 ± 0.62
Distill. Compr. Cellpose with depthw. sep. convs	92.80 ± 5.29	91.45 ± 6.31	91.94 ± 3.70	6.63 ± 0.38
Distill. FastCellpose	92.65 ± 2.56	92.99 ± 4.76	92.76 ± 2.71	4.55 ± 0.23
FasterCP v1	93.85 ± 2.95	91.04 ± 5.88	92.33 ± 3.32	4.75 ± 0.28
FasterCP v2	91.75 ± 5.68	92.17 ± 4.25	91.85 ± 3.55	3.76 ± 0.30
FasterCP v3	94.01 ± 3.43	90.05 ± 6.78	91.89 ± 4.31	3.26 ± 0.20

Table 4.4: Evaluation of the model on 2D slices with magnification 40× coming from Cystic phenotypic class (5 slices)

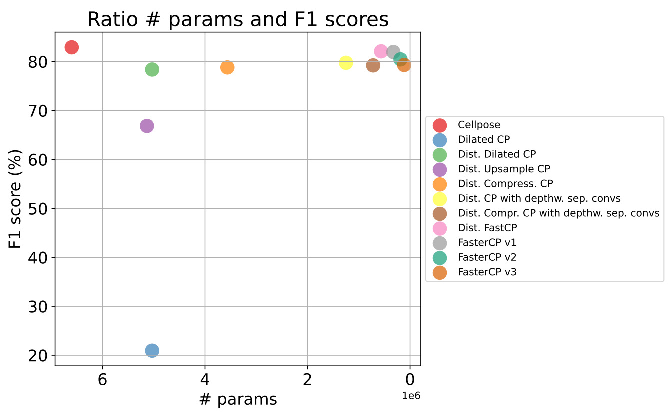
Figure 4.4 and Figure 4.5 clearly illustrate the performance of the models in F1 score on different magnification and phenotype, while decreasing their computational complexity. Only once Figure 4.4b, the distilled models achieved a higher F1 score than the baseline.



(a) Cystic phenotype (6 slices)

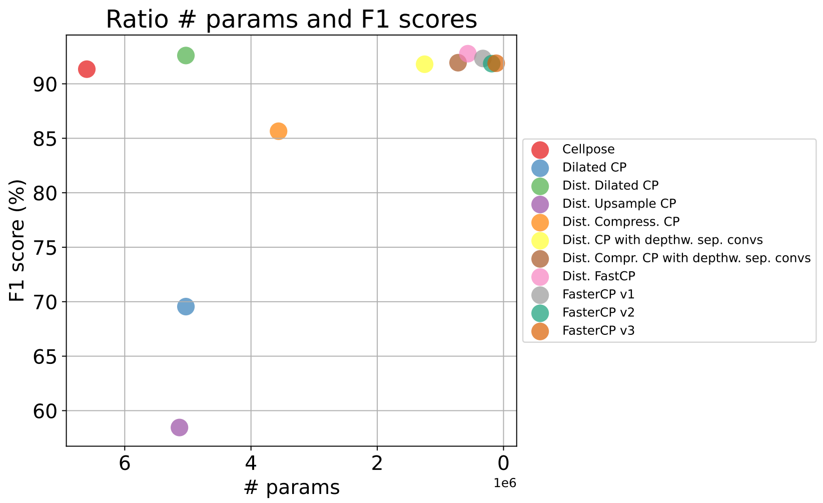


(b) Compact phenotype (7 slices)

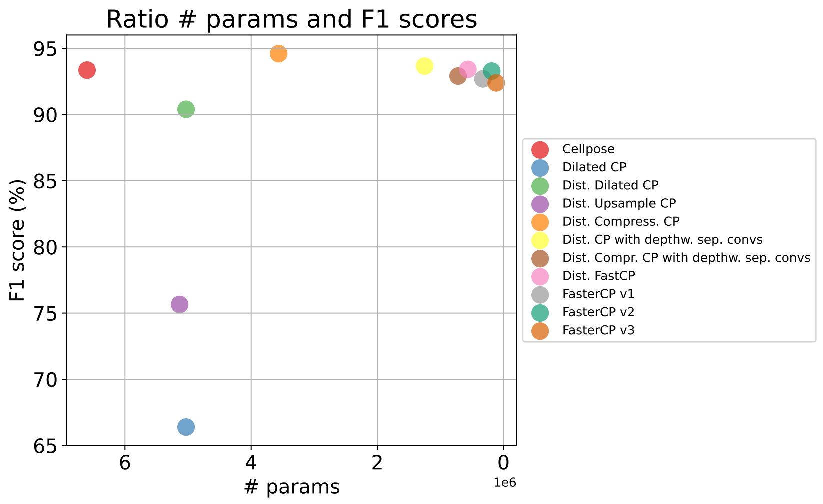


(c) Cauliflower phenotype (5 slices)

Figure 4.4: Performance on 2D slices of 20× magnification



(a) Cystic phenotype (5 slices)



(b) Cauliflower phenotype (5 slices)

Figure 4.5: Performance on 2D slices of 40× magnification (no samples for compact phenotype)

Figure 4.6 shows the models in terms of parameters and network inference times. Again, we observe an approximately linear relationship between the number of parameters and inference time.

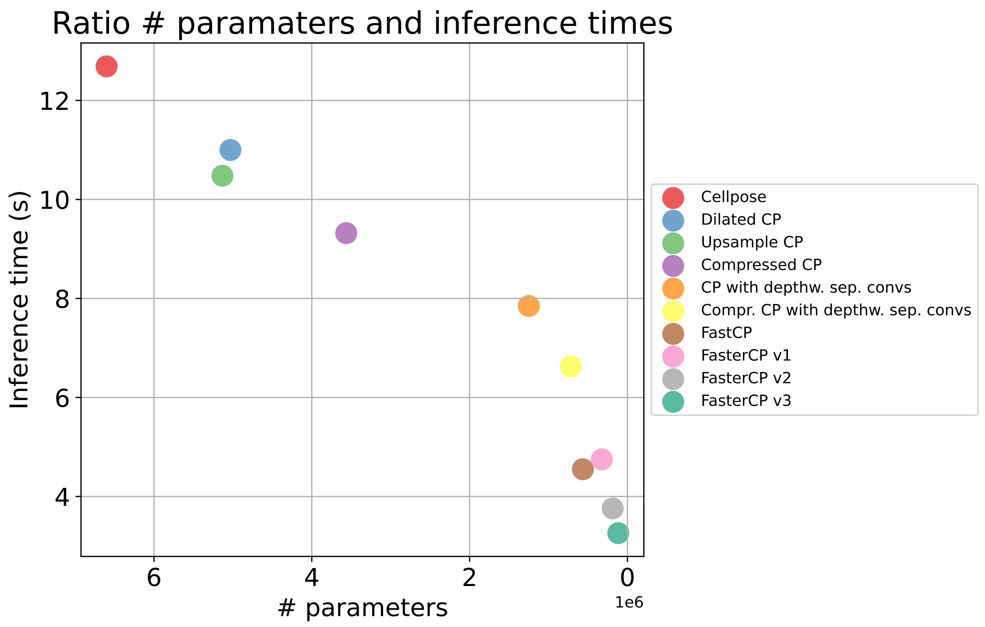


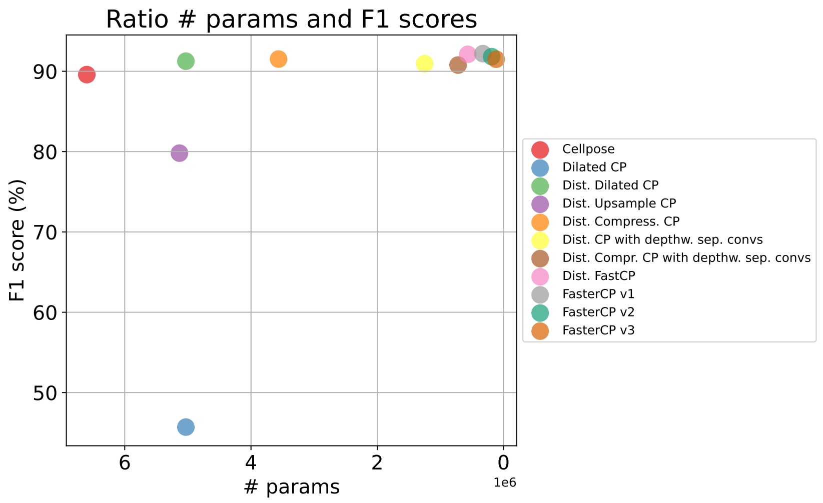
Figure 4.6: Comparison of models in terms of number of parameters and inference time on our dataset

4.5 Performance on benchmarks

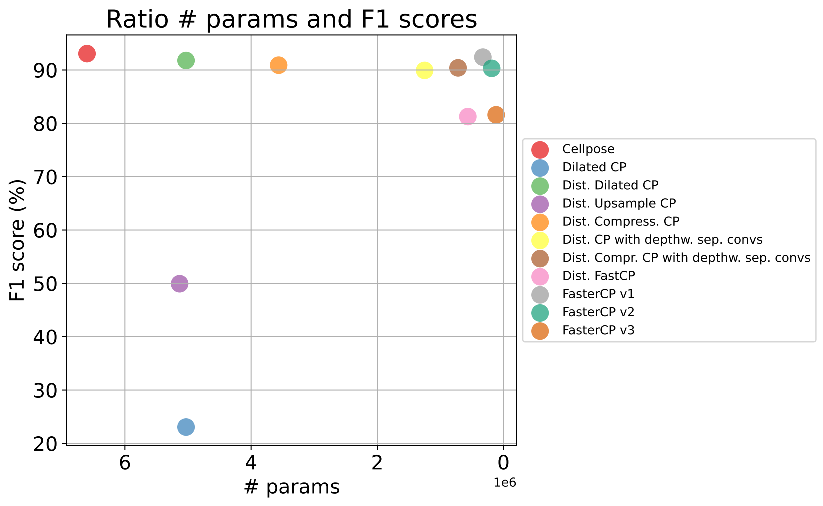
In the previous section, we assessed the performance of the distilled models on the test sets of our dataset. To determine whether the students can generalize as well as the teacher, we tested them on two widely used public datasets:

- *DAPI dataset* [76]: it contains 79 fluorescence images of densely packed nuclei of various tissues, sampled with multiple microscopies and magnifications. Expert biologists labeled these pictures. The main challenge is represented by touching nuclei: deciding whether a group of pixels consists of one or multiple nuclei.
- *BitDepth dataset* [77]: labeled fluorescence-nuclear images from five mouse organs, captured at multiple magnifications. The grayscale images are available at 8-bit and 16-bit resolutions.

Figure 4.7 and Figure 4.8 confirm that distilled models perform like the baseline. We observed a small drop in F1 score is present for FasterCellpose v3 compared to the baseline F1 score in Figure 4.7b. These graphs show that knowledge distillation from pre-trained Cellpose model enables our student models to generalize as effectively as the teacher on nuclear-biomarker images.

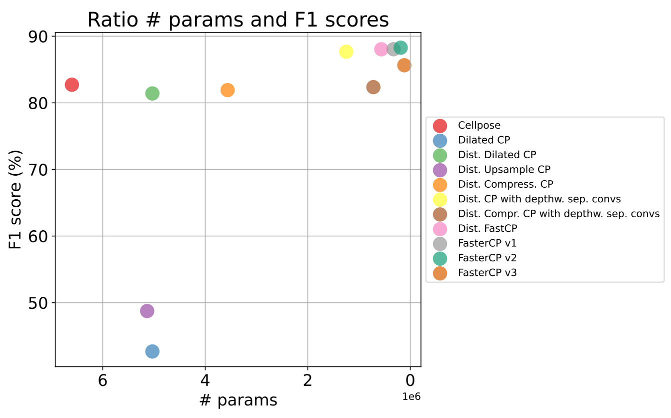


(a) 20× magnification

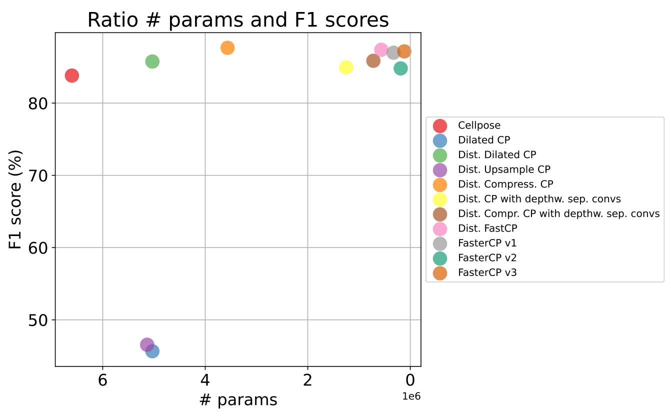


(b) 40× magnification

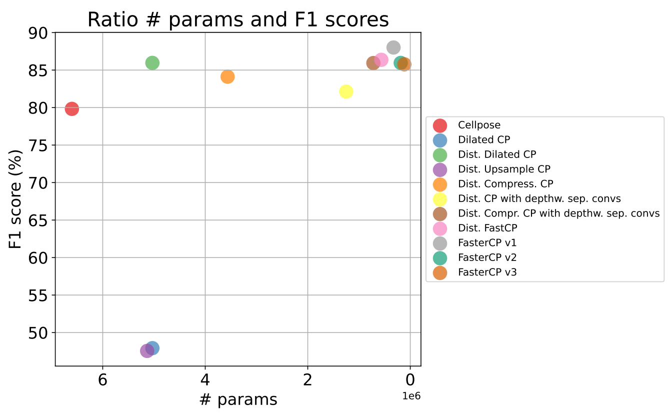
Figure 4.7: Performances on DAPI dataset



(a) 20× magnification



(b) 40× magnification (air immersion)



(c) 40× magnification (oil immersion)

Figure 4.8: Performances on BitDepth dataset

4.6 Performance on 3D images

Since our goal was to segment efficiently 3D volumes, we also compared the baseline and distilled models on 3D images. As observed in section 3.5, quantitative evaluation of 3D masks is challenging due to the scarcity of annotations. However, we estimated their quality through a qualitative visual comparison and measured inference times.
In Table 4.5, we listed the corresponding inference time for each model. Figure 4.9 summarizes 3D segmentation by plotting parameter count versus inference time on a sample with 237 slices 1024 × 1024 pixels sized.

Model	Network inf. time (min)	Percentage net. inf. time (%)
Cellpose	82.67	100.00
Dilated CP	63.54	76.86
Upsample CP	62.41	75.49
Compressed CP	51.26	62.00
CP with depthwise separable convs	39.36	47.61
Compr. CP with depthw. sep. convs	26.16	31.65
FastCP	20.11	24.32
FasterCP v1	16.53	20.00
FasterCP v2	12.50	15.12
FasterCP v3	9.79	11.84

Table 4.5: Comparison of network inference time on 3D volumes

Figure 4.10 and Figure 4.11 represent the mask and the crops respectively obtained by Cellpose and FasterCellpose v3. We observe also in Figure 4.11 that both models detect all nuclei, but their prediction differ with adjacent objects. On the XY plane both models recognize well the nuclei; but in one case two touching objects were merged into a single instance. On the YZ and XZ planes, Cellpose successfully splits some touching nuclei, whereas FasterCP v3 can not. Overall, this qualitative evaluation suggests that FasterCP v3 produces masks (Figure 4.10b) that appear reliable as those of Cellpose (Figure 4.10a).

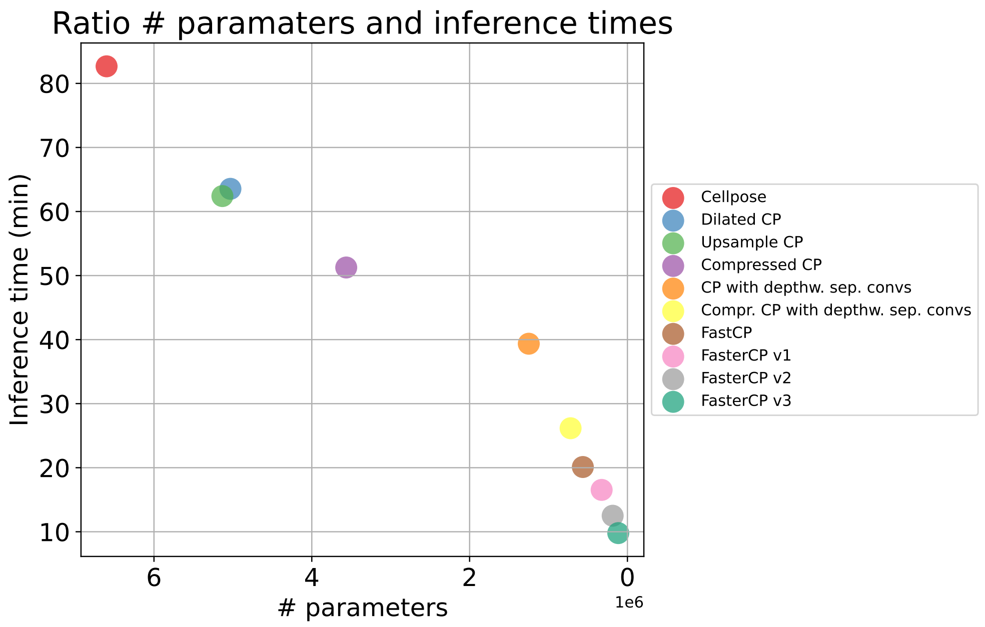
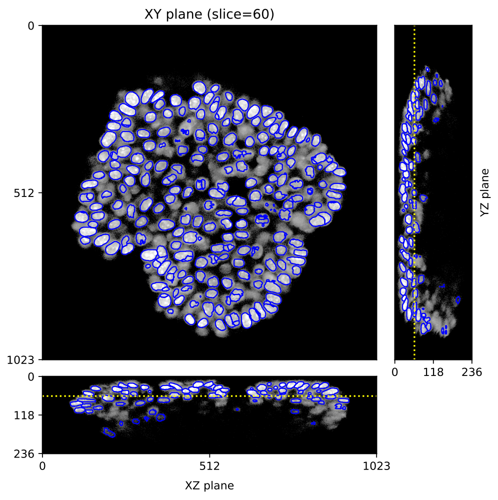
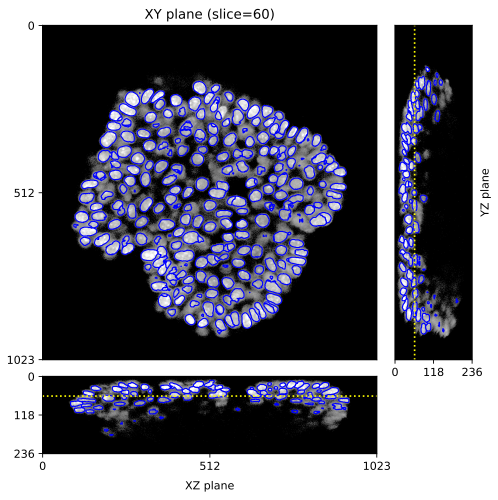


Figure 4.9: Comparison of models in terms of number of parameters and inference time on a 3D sample



(a) Cellpose



(b) FasterCellpose v3

Figure 4.10: Qualitative comparison between Cellpose and FasterCP v3 on 3D segmentation. The blue contours are the predicted object boundaries. The dotted yellow line on YZ and XZ planes represents the slice showed on XY plane.

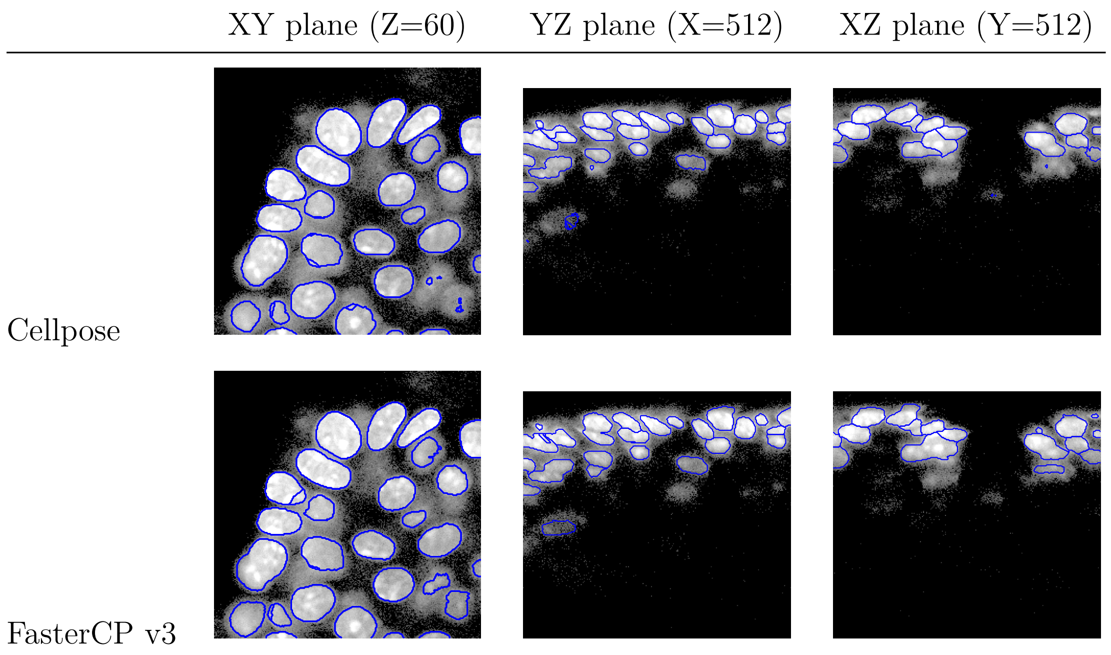


Figure 4.11: Qualitative comparison between Cellpose and FasterCP v3 predictions on crops of 3D volume. Each row represents the segmentation model, each column represents the plane of the 3D volume. The blue contours are the predicted object boundaries.

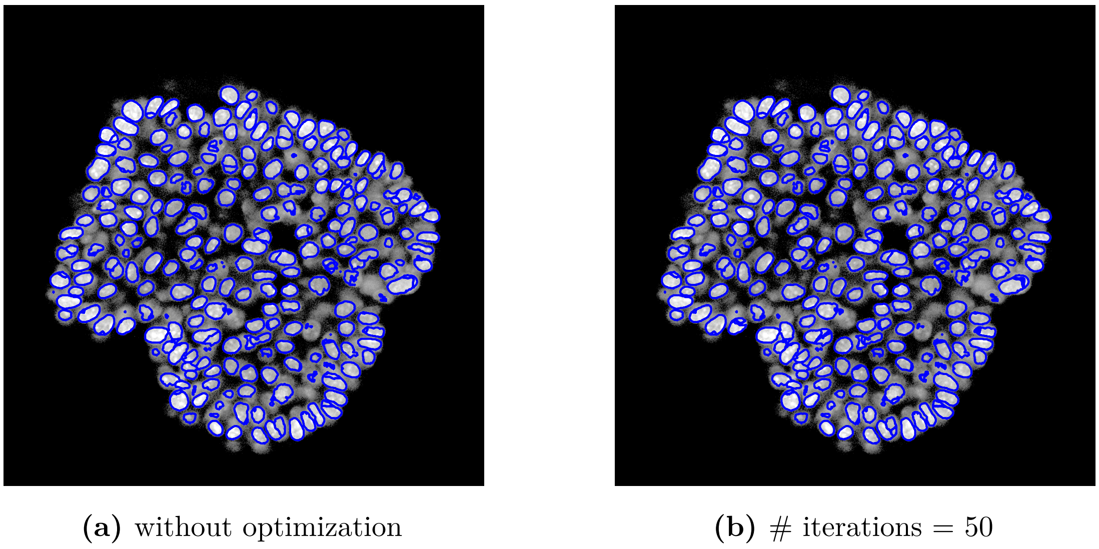
4.6.1 Mask reconstruction optimization

As mentioned in section 3.3, mask reconstruction in post-processing can be optimized by **resizing the vector field** and/or **reducing the number of "follow the flows" iterations**. Table 4.6 presents how these optimization techniques affect mask reconstruction time using FasterCellpose v2.

Model	Resize factor (×)	# grad. tracking iter.	Mask reconstruction time (s)	Percent. mask reconst. time (%)
FasterCP v2	1	200	2231.49	100.00
FasterCP v2	2	200	331.85	14.87
FasterCP v2	1	50	637.06	28.55
FasterCP v2	2	50	113.48	5.09

Table 4.6: Optimization on mask reconstruction time using FasterCellpose v2

Furthermore, Figure 4.12 shows nuclear segmentation with and without mask optimization. In both cases, the objects were correctly segmented. More specifically, Figure 4.13 highlights the differences in a 256 × 256 crop. Vector field resize brings some changes in the mask reconstruction: with the classic post-processing approach, we obtain smooth contours for each labeled object (Figure 4.13a), whereas downsampling the vector field and upsampling the mask yields rougher contours (Figure 4.13c). Depending on the number of gradient tracking iterations, a single nucleus could be split into multiple instances (Figure 4.13d) or considered as a single instance (Figure 4.13a).


(a) without optimization **(b)** # iterations = 50

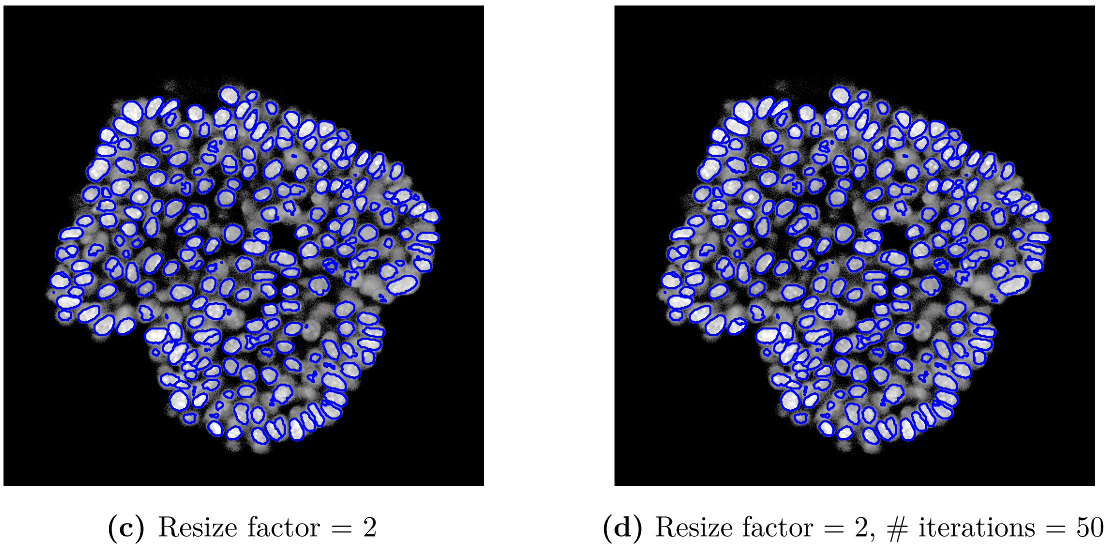
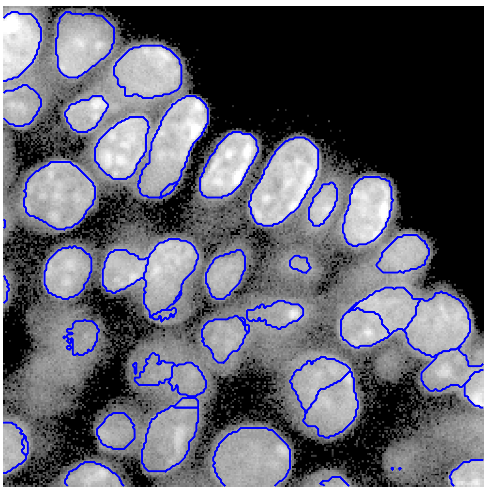
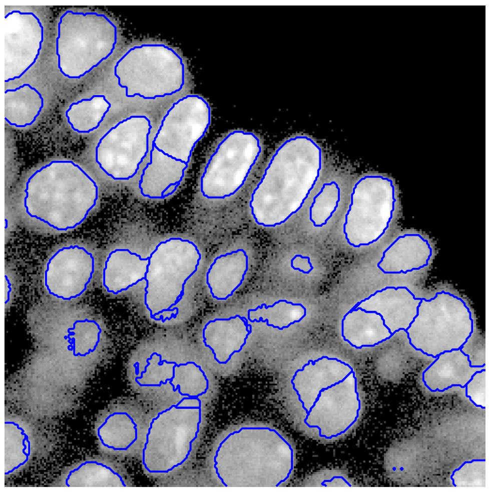
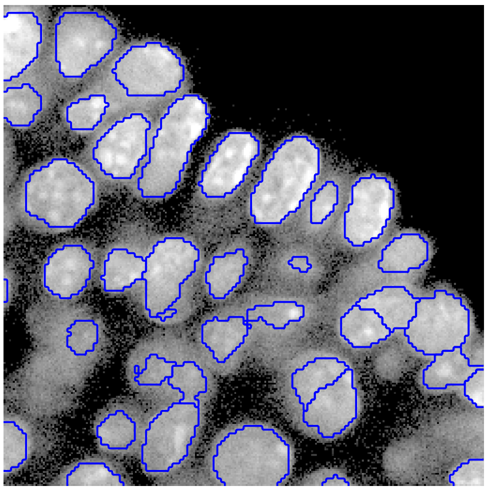

(c) Resize factor = 2 **(d)** Resize factor = 2, # iterations = 50

Figure 4.12: Comparison of masks with and without mask optimization techniques


(a) Without optimization


(b) # iterations = 50


(c) Resize factor = 2

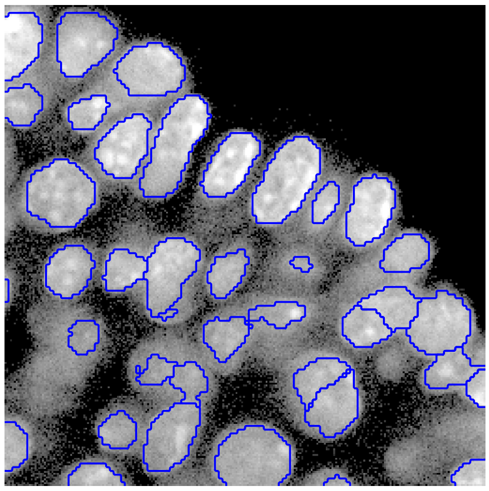

(d) Resize factor = 2, # iterations = 50

Figure 4.13: Mask optimization technique on a portion of 2D slice

4.7 Applications on 3D segmentation pipelines

As mentioned in section 1.4, we aimed also to accelerate the pipelines that include 3D instance segmentation. This task represents a bottleneck for the workflow. Therefore, we adopted our distilled lightweight model, FasterCellpose v3, for instance segmentation stage to speed up the application pipelines.

4.7.1 Application to GNN

Our work supported "Organoid Phenotypes Mapping and Modeling: Toward an Endocrine Disruptors Classification" project, led by MORPHEME team: one goal is to analyze organoids using graphs. *Graph Neural Network (GNN)* builds graphs from instance segmentation masks. Cell Graph Convolutional Network (CGC-Net) is an example of GNN that builds the graph from a segmented histology mask, with the nuclei as nodes and the cellular interactions as edges [78]. Although CGC-Net was developed for histological images, MORPHEME team applies a similar GNN architecture on confocal masks of organoids, using the nuclei as nodes and the interactions as edges.
The objective is to analyze cell structure and functionality via the graph, and then to classify the sample in a specific phenotype.

State-of-the-art models failed to produce accurate masks in a low amount of time, so MORPHEME team looked to some efficient alternatives.
The team initially used the *ellipsoidal model*, a model-based segmentation approach, to compute 3D segmentation masks by fitting ellipsoids to each nuclei. Its key benefit is the speed to perform the task. However, it does not segment accurately as Cellpose or StarDist. An example of ellipsoidal model is FitEllipsoid [79].
Therefore, we tested FasterCellpose v3 capabilities to improve the quality of the predictions, by maintaining a low inference time for segmentation stage.

The Figure 4.14 shows a qualitative comparison between the graphs obtained from the ellipsoidal model and FasterCellpose: the images contain nodes and edges, where the marker size represents the nuclei dimensions. The graph belonging to ellipsoidal model mask (figure on the left) contains plenty of nodes, but there are some areas where nodes are missing. Furthermore, the detected nuclei have a small size. Whereas, the graph on FasterCellpose mask (figure on the right) has a lower number of nodes, but some of them are larger than the ones in the other graph. Moreover, the empty areas encountered with ellipsoidal model graph were filled by nodes and connections.

MORPHEME team assessed that the graph obtained on FasterCellpose mask (Figure 4.15) represents well the organoid, while taking only a couple of minutes for segmentation.

The use of this distilled lightweight model allowed the GNN to cluster organoid phenotypes on more accurate masks and in short time (few minutes per 3D volume).

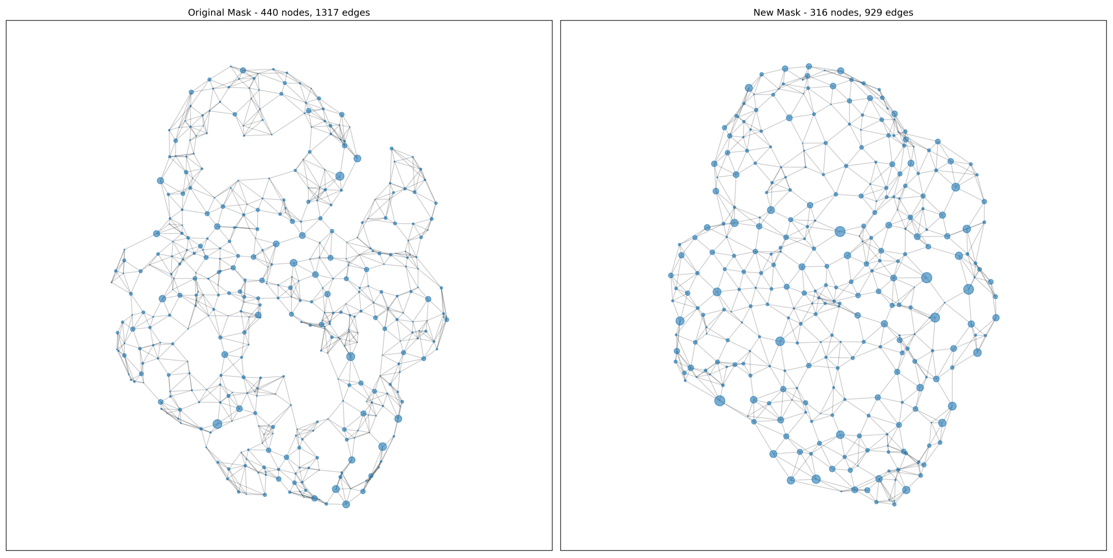

Original Mask - 440 nodes, 1317 edges
New Mask - 316 nodes, 929 edges

Figure 4.14: Comparison of graphs obtained by ellipsoidal model and FasterCP segmentation masks. On the left, the graph obtained by ellipsoidal model mask. On the right, the graph computed from FasterCP mask.

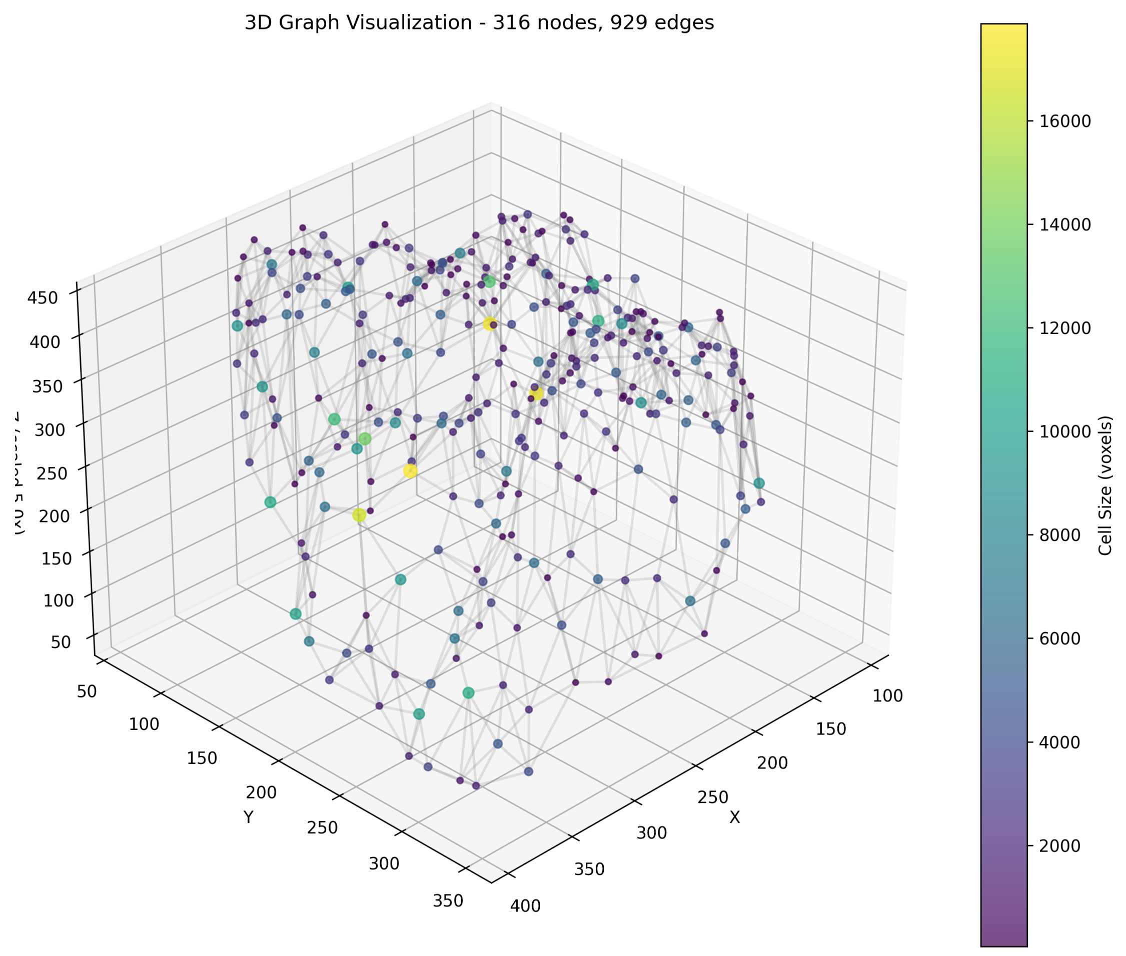


Figure 4.15: 3D representation of the graph obtained on FasterCP v3 segmentation mask

Chapter 5

Discussion

5.1 Overview

Given 3D confocal images with nuclear biomarker of mice prostate gland, we aimed to obtain a lightweight model that performs well in instance segmentation on CPU. We adopted Cellpose (subsection 2.2.2) as baseline because of its ease of use and of its pre-trained models. To achieve this goal we applied compression techniques on its architecture and we obtained two set of weights for each compact model using knowledge distillation (chapter 3).

5.2 Lightweight architecture design

Concerning lightweight model design, as shown in Table 3.5, FasterCellpose v3 is significantly lighter than both Cellpose (subsection 2.2.2) and FastCellpose (section 2.4). To our knowledge, no other compression frameworks for Cellpose beyond FastCellpose appeared before this work. Replacing every standard convolution with depthwise separable convolution (section 2.3.5) reduced the computational complexity by a factor of 5×: in fact, we passed from 6.6 M parameters and 326.8 G MACs to 1.2 M parameters and 51.9 G MACs. Next, halving the number of feature map channels at each depth scale significantly decreased the parameter count and the number of operations. Eventually, to achieve the final lightweight architecture many convolutional layers were removed; dilated convolutions and transposed ones were introduced to preserve mask quality.
Consequently, FasterCellpose v3 achieves significantly reduced inference times for both 2D and 3D segmentation. FasterCellpose v3 produce a 2D and a 3D mask in around 15 seconds and 15 minutes, respectively.

5.3 Distilled models and their performance

Before applying knowledge distillation, we loaded pre-trained Cellpose weights to the lightweight models, if possible. Among them, Dilated Cellpose performed relatively well, but Upsample Cellpose did not, even when using only unpooling layers instead of transposed convolutions in the upsample path. FastCellpose also failed, since it was trained to segment glomeruli images rather than our images.
Therefore, we leveraged the knowledge distillation to adjust lightweight models' weights. In terms of quantitative performance (Precision, Recall and mainly F1 score), section 4.4 show that the distilled variants match baseline performance on our dataset, on both 20× and 40× magnifications. Moreover, we observed that they are also able to generalize to other nuclear-image datasets, as described in section 4.5.
It means that distilled models learn to accurately segment not only the dataset they were trained on, but also samples they have never seen, thanks to the baseline expertise.
Occasionally, the distilled models even outperformed the baseline. We attributed this to random weight initialization and the optimizer converging to a different (local or global) optimum.

5.3.1 Considerations on Dilated Cellpose

In subsection 3.3.1 we presented dilated convolutions. These layers compute convolution by inserting spaces between kernel elements to increase the receptive field [67].
For each spatial block, we used two 3×3 convolutional layers where the first is a standard convolution, while the second is a dilated convolution. The result is a feature map with 7×7 receptive field, a trade-off between 3×3 using two standard convolutions and 9×9 using four standard convolutions.
However, there is a possibility to achieve an output feature map with 9×9 receptive field using two convolutions: the solution is to use two layers with $dilation = 2$.
Although two 3×3 dilated convolutions ($d = 2$) would match a 9×9 receptive field, we avoided this configuration because consecutive dilated convolutions can amplify gridding artifacts in the output feature map [80].
There are several solutions to avoid these artifacts, such as adding convolutional layers with progressively lower dilation until reaching $dilation = 1$ [80] or using a Hybrid Dilated Convolution (HDC) composed by n consecutive convolutions with progressively higher dilation [68, 69]. Other solutions employ dilated convolutions in parallel to capture more context [69].
We concluded that the HDC proposal is the best option to have fewer artifacts, better preservation of detail, and a progressive context aggregation.

5.3.2 Considerations on Upsample Cellpose

We developed several lightweight models based on U-Net backbone. The U-Net backbone is symmetric and consists of an encoder for feature extraction and a decoder for detail reconstruction [81]. Dilated and Upsample Cellpose are two intermediate models obtained by compressing one part of the network and not the whole network.

In Upsample Cellpose (subsection 3.3.2), we introduced transposed convolutions to better control the image restoration. However, they can introduce checkerboard artifacts. To avoid this gridding style, we interleaved transposed convolutions and unpooling layers in decoding.

The graphs of section 4.4 and section 4.5 showed that Upsample Cellpose model does not reach satisfying results, even after distillation. In contrast, Dilated Cellpose, an encoding-compressed model, achieved good results even by loading pre-trained Cellpose weights.
We attributed the Upsample Cellpose inefficiency mainly to an under-powered decoder. With only two convolutions per block and with transposed convolutions and unpooling layers, it can not progressively restore segmentation maps from high-dimensional abstract features. Consequently, the decoder suffers both interpolation and transposed convolution artifacts, and degraded feature map quality [81].

Compressed Cellpose presents two convolutions per unit on both encoder and decoder and preserves the prediction quality employing dilation and transposed convolution layers.
The excellent results with distilled Compressed Cellpose confirmed that maintaining the symmetry of U-Net architecture is crucial for robust segmentation under compression.

Furthermore, instead of transposed convolutions, a solution is to perform nearest-neighbor upsampling followed by a standard convolution [71].

5.4 Reduction of memory footprint

Although these findings are encouraging, we should also keep in mind that model compression does not necessarily reduce memory usage. Indeed, as observed in a study [48], although parameter and computation counts drop with MobileNet depthwise separable convolutions, it can inadvertently lead to an increase in memory usage due to larger activation maps. This impacts training more than inference.

Cellpose weights and activations are represented by 32-bit floating point values. To save memory we can change the value representation by applying quantization (subsection 2.3.2).

Otherwise, pruning can be applied to have a sparse model from an over-parametrized neural network (subsection 2.3.1).

5.5 Lack of annotated 3D dataset

Another limitation is due to the annotation of 3D samples (subsection 3.1.1): labels for 3D volumes would be really useful to evaluate segmentation masks.
Manual annotation of 3D images is time-consuming. Tools such as *Labkit* [82] support dense manual labeling or sparse annotation followed by automated segmentation via random forest pixel classification. However, because full manual annotation in 3D is very laborious and semi-automatic annotation is not as reliable as expert biologists' labeling, we chose to not use Labkit in this project.

Chapter 6

Future works

In the previous chapter, we demonstrated the strengths and the weaknesses of our work. Albeit the efficiency of FasterCellpose v3, it takes up memory space. Furthermore, we worked on unlabeled 3D images. It would be interesting to annotate samples with tools that we have already cited.

6.1 Reduction of memory footprint

We demonstrated how to get a lightweight and effective model to perform segmentation on nuclear biomarker images like Cellpose. Although FasterCellpose v3 is 56 $\times$ smaller than the baseline, this model still takes up memory space during execution. Since our goal is to deploy this model also on low-performing CPU, reducing computational complexity can not be sufficient, but it would be necessary to reduce the memory storage.
The main techniques to reduce the space in memory are two:

- Pruning removes redundant parameters (subsection 2.3.1)
- Quantization converts weights and activations from high precision to low precision (subsection 2.3.2)

Quantization is likely more effective, since it shrinks both parameter count and forward/backward pass size, while pruning mainly reduces parameter size.
As future work we could cast value from floating point to integer data type and reduce the number of bits to represent the values. Furthermore, we could apply unstructured pruning to remove redundant weights without losing in accuracy, otherwise to structured pruning with a further training.

6.2 Annotation of 3D images

Another issue is the lack of labeled 3D images. Indeed, in our work we resorted to unsupervised learning approach to train small networks on these samples. Some public tools can annotate 3D images in a semi-automatic way, like Labkit [82]. Since we can not rely solely on labels automatically applied by a tool, we would like expert biologists to annotate these confocal image stacks. These annotations would enable not only the application of supervised techniques, but also the quantitative evaluation of 3D segmentation masks.

Chapter 7

Conclusion

This work demonstrated how to obtain lightweight neural network that segments efficiently 3D confocal images on CPU.
FasterCellpose was designed from the Cellpose architecture by removing layers and feature map channels, and replacing all standard convolutions with depthwise separable ones. Our latest model is 5× smaller than FastCellpose, the latest compressed version of Cellpose of our knowledge.
We trained this compact model on unlabeled dataset of confocal images with nuclear biomarker through knowledge distillation: we adopted the offline approach, where the teacher is the pre-trained Cellpose model and the students are the lightweight models. The results indicated that the distilled models achieve baseline performance not only on our confocal images, but also on other fluorescence image benchmarks, showing effective generalization across datasets.
The network inference time decreased on both 2D and 3D segmentation: FasterCellpose v3 is 8.44× faster than Cellpose on CPU, producing a 3D mask in around 15 minutes.
Building on these gains, future work will focus on making this model lighter in memory via pruning and quantization.

Bibliography

[1] Claude Monneret. «What is an endocrine disruptor?» In: *Comptes Rendus. Biologies* 340.9-10 (2017), pp. 403–405 (cit. on p. 1).

[2] Andrea C Gore, Vesna A Chappell, Suzanne E Fenton, Jodi Anne Flaws, Angel Nadal, Gail S Prins, Jorma Toppari, and R Thomas Zoeller. «EDC-2: the endocrine society's second scientific statement on endocrine-disrupting chemicals». In: *Endocrine reviews* 36.6 (2015), E1–E150 (cit. on pp. 1, 2).

[3] Eva Rahman Kabir, Monica Sharfin Rahman, and Imon Rahman. «A review on endocrine disruptors and their possible impacts on human health». In: *Environmental toxicology and pharmacology* 40.1 (2015), pp. 241–258 (cit. on pp. 1, 2).

[4] Omar ML Alharbi, Rafat A Khattab, Imran Ali, et al. «Health and environmental effects of persistent organic pollutants». In: *Journal of Molecular Liquids* 263 (2018), pp. 442–453 (cit. on pp. 1, 2).

[5] Ruth Lehmann et al. «Human organoids: a new dimension in cell biology». In: *Molecular biology of the cell* 30.10 (2019), pp. 1129–1137 (cit. on pp. 2, 3).

[6] Thomas K Nelson. «ARE HUMAN CELL LINES HUMAN?» In: *European Scientific Journal* (2015) (cit. on p. 2).

[7] Aurélie Lacouture, Camille Lafront, Cindy Peillex, Martin Pelletier, and Étienne Audet-Walsh. «Impacts of endocrine-disrupting chemicals on prostate function and cancer». In: *Environmental Research* 204 (2022), p. 112085 (cit. on pp. 2, 3).

[8] Giuliana Rossi, Andrea Manfrin, and Matthias P Lutolf. «Progress and potential in organoid research». In: *Nature Reviews Genetics* 19.11 (2018), pp. 671–687 (cit. on p. 2).

[9] Siqi Yang et al. «Organoids: The current status and biomedical applications». In: *MedComm* 4.3 (2023), e274. DOI: https://doi.org/10.1002/mco2.274. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mco2.274. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mco2.274 (cit. on pp. 2, 3).

[10] Ravian L van Ineveld, Hendrikus CR Ariese, Ellen J Wehrens, Johanna F Dekkers, Anne C Rios, et al. *Single-cell resolution three-dimensional imaging of intact organoids*. MyJove Corporation, 2016 (cit. on p. 4).

[11] Adaobi Nwaneshiudu, Christiane Kuschal, Fernanda H Sakamoto, R Rox Anderson, Kathryn Schwarzenberger, and Roger C Young. «Introduction to confocal microscopy». In: *Journal of Investigative Dermatology* 132.12 (2012), pp. 1–5 (cit. on p. 4).

[12] Anne C Rios and Hans Clevers. «Imaging organoids: a bright future ahead». In: *Nature methods* 15.1 (2018), pp. 24–26 (cit. on p. 4).

[13] Amicia D Elliott. «Confocal microscopy: principles and modern practices». In: *Current protocols in cytometry* 92.1 (2020), e68 (cit. on p. 4).

[14] Tony Wilson. «Spinning-disk microscopy systems». In: *Cold Spring Harbor Protocols* 2010.11 (2010), pdb–top88 (cit. on p. 4).

[15] Thomas Klonisch, Landon Wark, Sabine Hombach-Klonisch, and Sabine Mai. «Nuclear imaging in three dimensions: a unique tool in cancer research». In: *Annals of Anatomy-Anatomischer Anzeiger* 192.5 (2010), pp. 292–301 (cit. on p. 4).

[16] Juan C Caicedo et al. «Nucleus segmentation across imaging experiments: the 2018 Data Science Bowl». In: *Nature methods* 16.12 (2019), pp. 1247–1253 (cit. on pp. 4, 9, 11, 13, 35).

[17] Martin Weigert, Uwe Schmidt, Robert Haase, Ko Sugawara, and Gene Myers. «Star-convex polyhedra for 3D object detection and segmentation in microscopy». In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2020, pp. 3666–3673 (cit. on pp. 6, 10).

[18] Carsen Stringer, Tim Wang, Michalis Michaelos, and Marius Pachitariu. «Cellpose: a generalist algorithm for cellular segmentation». In: *Nature methods* 18.1 (2021), pp. 100–106 (cit. on pp. 6, 9, 11, 12, 32).

[19] Yutong Han et al. «FastCellpose: a fast and accurate deep-learning framework for segmentation of all glomeruli in mouse whole-kidney microscopic optical images». In: *Cells* 12.23 (2023), p. 2753 (cit. on pp. 6, 19, 24).

[20] Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. «A survey on instance segmentation: state of the art». In: *International journal of multimedia information retrieval* 9.3 (2020), pp. 171–189 (cit. on pp. 6, 9).

[21] Wenchao Gu, Shuang Bai, and Lingxing Kong. «A review on 2D instance segmentation based on deep neural networks». In: *Image and Vision Computing* 120 (2022), p. 104401 (cit. on pp. 6–8).

[22] Koen EA Van de Sande, Jasper RR Uijlings, Theo Gevers, and Arnold WM Smeulders. «Segmentation as selective search for object recognition». In: *2011 international conference on computer vision*. IEEE. 2011, pp. 1879–1886 (cit. on p. 7).

[23] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. «Mask r-cnn». In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969 (cit. on p. 7).

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: `1505.04597 [cs.CV]`. URL: `https://arxiv.org/abs/1505.04597` (cit. on p. 7).

[25] Ilya Levner and Hong Zhang. «Classification-driven watershed segmentation». In: *IEEE Transactions on Image Processing* 16.5 (2007), pp. 1437–1445 (cit. on p. 7).

[26] Peigeng Li. «A Comparative Analysis of Single-Stage Detectors from the Perspectives of Anchor-Free and Anchor-Based Approaches». In: *Highlights in Science, Engineering and Technology* 72 (Dec. 2023), pp. 774–782. DOI: `10.54097/s37sgq58` (cit. on pp. 7, 8).

[27] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. «Instance-sensitive fully convolutional networks». In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*. Springer. 2016, pp. 534–549 (cit. on p. 7).

[28] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. «Yolact: Real-time instance segmentation». In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9157–9166 (cit. on p. 7).

[29] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. «Fcos: Fully convolutional one-stage object detection». In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9627–9636 (cit. on p. 8).

[30] Yong He, Hongshan Yu, Xiaoyan Liu, Zhengeng Yang, Wei Sun, Saeed Anwar, and Ajmal Mian. «Deep learning based 3D segmentation: A survey». In: *arXiv preprint arXiv:2103.05423* (2021) (cit. on p. 8).

[31] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. «3D U-Net: learning dense volumetric segmentation from sparse annotation». In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*. Springer. 2016, pp. 424–432 (cit. on pp. 8, 10).

[32] Yichi Zhang, Qingcheng Liao, Le Ding, and Jicong Zhang. «Bridging 2D and 3D segmentation networks for computation-efficient volumetric medical image segmentation: An empirical study of 2.5 D solutions». In: *Computerized Medical Imaging and Graphics* 99 (2022), p. 102088 (cit. on pp. 8, 9).

[33] Amarjeet Kumar et al. «A flexible 2.5 D medical image segmentation approach with in-slice and cross-slice attention». In: *Computers in Biology and Medicine* 182 (2024), p. 109173 (cit. on p. 9).

[34] Yuxing Wang, Junhan Zhao, Hongye Xu, Cheng Han, Zhiqiang Tao, Dawei Zhou, Tong Geng, Dongfang Liu, and Zhicheng Ji. «A systematic evaluation of computational methods for cell segmentation». In: *Briefings in Bioinformatics* 25.5 (2024), bbae407 (cit. on p. 9).

[35] Ping Liu, Jun Li, Jiaxing Chang, Pinli Hu, Yue Sun, Yanan Jiang, Fan Zhang, and Haojing Shao. «Software Tools for 2D Cell Segmentation». In: *Cells* 13.4 (2024), p. 352 (cit. on p. 9).

[36] Uwe Schmidt, Martin Weigert, Coleman Broaddus, and Gene Myers. «Cell detection with star-convex polygons». In: *Medical image computing and computer assisted intervention–MICCAI 2018: 21st international conference, Granada, Spain, September 16-20, 2018, proceedings, part II 11.* Springer. 2018, pp. 265–273 (cit. on p. 9).

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep residual learning for image recognition». In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 770–778 (cit. on p. 10).

[38] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. «Learning spatio-temporal features with 3d residual networks for action recognition». In: *Proceedings of the IEEE international conference on computer vision workshops.* 2017, pp. 3154–3160 (cit. on p. 10).

[39] Vladimír Ulman, Zoltán Orémuš, and David Svoboda. «TRAgen: a tool for generation of synthetic time-lapse image sequences of living cells». In: *Image Analysis and Processing—ICIAP 2015: 18th International Conference, Genoa, Italy, September 7-11, 2015, Proceedings, Part I 18.* Springer. 2015, pp. 623–634 (cit. on p. 11).

[40] Fuhui Long, Hanchuan Peng, Xiao Liu, Stuart K Kim, and Eugene Myers. «A 3D digital atlas of C. elegans and its application to single-cell analyses». In: *Nature methods* 6.9 (2009), pp. 667–672 (cit. on p. 11).

[41] Frederike Alwes, Camille Enjolras, and Michalis Averof. «Live imaging reveals the progenitors and cell dynamics of limb regeneration». In: *Elife* 5 (2016), e19766 (cit. on p. 11).

[42] Kwanyoung Lee, Hyungjo Byun, and Hyunjung Shim. «Cell Segmentation in Multi-modality High-Resolution Microscopy Images with Cellpose». In: *Competitions in Neural Information Processing Systems*. PMLR. 2023, pp. 1–11 (cit. on pp. 12, 13, 32).

[43] Giona Kleinberg, Sophia Wang, Ester Comellas, James R Monaghan, and Sandra J Shefelbine. «Usability of deep learning pipelines for 3D nuclei identification with Stardist and Cellpose». In: *Cells & development* 172 (2022), p. 203806 (cit. on p. 13).

[44] Marius Pachitariu and Carsen Stringer. «Cellpose 2.0: how to train your own model». In: *Nature methods* 19.12 (2022), pp. 1634–1641 (cit. on p. 13).

[45] Giosué Cataldo Marinó, Alessandro Petrini, Dario Malchiodi, and Marco Frasca. «Deep neural networks compression: A comparative survey and choice recommendations». In: *Neurocomputing* 520 (2023), pp. 152–170. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.11.072. URL: https://www.sciencedirect.com/science/article/pii/S0925231222014643 (cit. on pp. 14, 16).

[46] Zhuo Li, Hengyi Li, and Lin Meng. «Model compression for deep neural networks: A survey». In: *Computers* 12.3 (2023), p. 60 (cit. on pp. 14–18, 28).

[47] Simone Bianco, Remi Cadene, Luigi Celona, and Paolo Napoletano. «Benchmark Analysis of Representative Deep Neural Network Architectures». In: *IEEE Access* 6 (2018), pp. 64270–64277. DOI: 10.1109/ACCESS.2018.2877890 (cit. on p. 14).

[48] Nandan Kumar Jha, Sparsh Mittal, and Govardhan Mattela. «The ramifications of making deep neural networks compact». In: *2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)*. IEEE. 2019, pp. 215–220 (cit. on pp. 14, 53).

[49] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. «Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks». In: *Journal of Machine Learning Research* 22.241 (2021), pp. 1–124 (cit. on p. 15).

[50] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. «A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024) (cit. on p. 15).

[51] Sunil Vadera and Salem Ameen. «Methods for pruning deep neural networks». In: *IEEE Access* 10 (2022), pp. 63280–63300 (cit. on p. 15).

[52] JT O'Neill. «An survey of neural network compression». In: *arXiv preprint arXiv:2006.03669* (2020) (cit. on pp. 15, 16).

[53] Pierre Vilar Dantas, Waldir Sabino da Silva Jr, Lucas Carvalho Cordeiro, and Celso Barbosa Carvalho. «A comprehensive review of model compression techniques in machine learning». In: *Applied Intelligence* 54.22 (2024), pp. 11804–11844 (cit. on pp. 15, 17, 18).

[54] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. «Model compression and acceleration for deep neural networks: The principles, progress, and challenges». In: *IEEE Signal Processing Magazine* 35.1 (2018), pp. 126–136 (cit. on p. 16).

[55] Dong-Hyun Lee et al. «Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks». In: *Workshop on challenges in representation learning, ICML*. Vol. 3. 2. Atlanta. 2013, p. 896 (cit. on p. 16).

[56] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. «Distilling the knowledge in a neural network». In: *arXiv preprint arXiv:1503.02531* (2015) (cit. on p. 16).

[57] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. «Knowledge distillation: A survey». In: *International Journal of Computer Vision* 129.6 (2021), pp. 1789–1819 (cit. on pp. 16, 17).

[58] Zonglei Lyu, Tong Yu, Fuxi Pan, Yilin Zhang, Jia Luo, Dan Zhang, Yiren Chen, Bo Zhang, and Guangyao Li. «A survey of model compression strategies for object detection». In: *Multimedia tools and applications* 83.16 (2024), pp. 48165–48236 (cit. on pp. 17, 18).

[59] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. «SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size». In: *arXiv preprint arXiv:1602.07360* (2016) (cit. on p. 18).

[60] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. «Mobilenets: Efficient convolutional neural networks for mobile vision applications». In: *arXiv preprint arXiv:1704.04861* (2017) (cit. on pp. 18, 19, 27).

[61] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. «Shufflenet: An extremely efficient convolutional neural network for mobile devices». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6848–6856 (cit. on p. 18).

[62] Weiming Zhi, Zhenghao Chen, Henry Wing Fung Yueng, Zhicheng Lu, Seid Miad Zandavi, and Yuk Ying Chung. «Layer removal for transfer learning with deep convolutional neural networks». In: *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part II 24*. Springer. 2017, pp. 460–469 (cit. on p. 18).

[63] François Chollet. «Xception: Deep learning with depthwise separable convolutions». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258 (cit. on pp. 19, 27).

[64] Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. «A survey of human-in-the-loop for machine learning». In: *Future Generation Computer Systems* 135 (2022), pp. 364–381 (cit. on p. 20).

[65] Johannes Schindelin et al. «Fiji: an open-source platform for biological-image analysis». In: *Nature methods* 9.7 (2012), pp. 676–682 (cit. on pp. 21, 23).

[66] Fisher Yu and Vladlen Koltun. «Multi-scale context aggregation by dilated convolutions». In: *arXiv preprint arXiv:1511.07122* (2015) (cit. on p. 25).

[67] Vincent Dumoulin and Francesco Visin. «A guide to convolution arithmetic for deep learning». In: *arXiv preprint arXiv:1603.07285* (2016) (cit. on pp. 25, 26, 30, 52).

[68] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. «Understanding convolution for semantic segmentation». In: *2018 IEEE winter conference on applications of computer vision (WACV)*. Ieee. 2018, pp. 1451–1460 (cit. on pp. 25, 52).

[69] Zhengyang Wang and Shuiwang Ji. «Smoothed dilated convolutions for improved dense prediction». In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 2486–2495 (cit. on pp. 25, 52).

[70] Shashank Agnihotri, Julia Grabinski, and Margret Keuper. «Improving stability during upsampling–on the importance of spatial context». In: *arXiv preprint arXiv* 2311 (2023) (cit. on pp. 25, 26).

[71] Augustus Odena, Vincent Dumoulin, and Chris Olah. «Deconvolution and Checkerboard Artifacts». In: *Distill* (2016). DOI: `10.23915/distill.00003`. URL: `http://distill.pub/2016/deconv-checkerboard` (cit. on pp. 26, 53).

[72] Huaping Zhou, Qi Zhao, and Kelei Sun. «D2-Net: Dilated Contextual Transformer and Depth-wise Separable Deconvolution for Remote Sensing Imagery Detection.» In: *International Journal of Advanced Computer Science & Applications* 14.11 (2023) (cit. on p. 27).

[73] Irem Ulku and Erdem Akagündüz. «A survey on deep learning-based architectures for semantic segmentation on 2d images». In: *Applied Artificial Intelligence* 36.1 (2022), p. 2032924 (cit. on pp. 31, 32).

[74] Dominik Müller, Iñaki Soto-Rey, and Frank Kramer. «Towards a guideline for evaluation metrics in medical image segmentation». In: *BMC Research Notes* 15.1 (2022), p. 210 (cit. on p. 31).

[75] Diederik P Kingma and Jimmy Ba. «Adam: A method for stochastic optimization». In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on pp. 31, 34).

[76] Florian Kromp et al. «An annotated fluorescence image dataset for training nuclear segmentation methods». In: *Scientific Data* 7.1 (2020), p. 262 (cit. on pp. 33, 40).

[77] Amirreza Mahbod, Gerald Schaefer, Christine Löw, Georg Dorffner, Rupert Ecker, and Isabella Ellinger. «Investigating the impact of the bit depth of fluorescence-stained images on the performance of deep learning-based nuclei instance segmentation». In: *Diagnostics* 11.6 (2021), p. 967 (cit. on pp. 33, 40).

[78] Maciej Krzywda, Szymon Łukasik, and Amir H Gandomi. «Graph neural networks in computer vision-architectures, datasets and common approaches». In: *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2022, pp. 1–10 (cit. on p. 48).

[79] Bastien Kovac, Jérôme Fehrenbach, Ludivine Guillaume, and Pierre Weiss. «FitEllipsoid: a fast supervised ellipsoid segmentation plugin». In: *BMC bioinformatics* 20 (2019), pp. 1–8 (cit. on p. 48).

[80] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. «Dilated residual networks». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 472–480 (cit. on p. 52).

[81] Weibin Yang, Longwei Xu, Pengwei Wang, Dehua Geng, Yusong Li, Mingyuan Xu, and Zhiqi Dong. «More complex encoder is not all you need». In: *arXiv preprint arXiv:2309.11139* (2023) (cit. on p. 53).

[82] Matthias Arzt, Joran Deschamps, Christopher Schmied, Tobias Pietzsch, Deborah Schmidt, Pavel Tomancak, Robert Haase, and Florian Jug. «LABKIT: labeling and segmentation toolkit for big image data». In: *Frontiers in computer science* 4 (2022), p. 777728 (cit. on pp. 54, 56).