Optimizing Deep Neural Networks for Single Cell Segmentation

Thesis by

Can (Sunny) Cui

In Partial Fulfillment of the Requirements for the
degree of
Bachelors of Science in Electrical Engineering

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2020
June 12, 2020

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor, Professor David Van Valen, for his guidance and mentorship. Working in your lab in the past two years has been an exciting and invaluable process of exploring open-ended questions. Your creativity and diligence when solving problems is incredibly inspiring. Thank you for always making time to meet with me and check-in with me - your constant encouragement and support helped develop my passion for deep learning in biology and for academic research. This experience has been instrumental in my growth as a student and as a researcher, and is a highlight of my Caltech experience!

I would also like to thank Erick Moen for all his advice and support. You are always eager and patient in helping me understand new concepts, even when I ask you to explain the same thing five or ten times. You are an incredible teacher and friend - thank you so much for believing in me when I did not believe in myself! I am so grateful to have worked with you!

To Will Graf, I cannot thank you enough for helping me debug. Whether my bug is silly or complex, I know you have my back. You taught me so many great coding practices from you. Hopefully, someday my code quality and debugging skills can be on par with yours!

Thank you to Dylan who welcomed me and mentored me in my first year with the Van Valen lab. Thank you to Morgan for creating the metrics packages for the analyses in my research. Thank you to Geneva for helping curate the cytoplasm datasets. Finally, a huge thank you to everyone else in the Van Valen Lab for taking me in and making it a great learning and working environment!

ABSTRACT

Analysis of live-cell imaging experiments at the resolution of single cells provides exciting insights into the inner workings of biological systems. Advances in biological imaging and computer vision allow for segmentation of natural images with a high degree of accuracy. However, automation of the segmentation pipeline at the single cell resolution remains a challenging task. Complex deep learning models require large, well-annotated datasets that are rarely available in biology. In this research, we explore various methods that optimize state of the art deep learning frameworks, despite limited resources. We trained a large permutation of models to quantify their capacity and to measure the effects of temporal information, spatial awareness and transfer learning on model performance. We find that, although training set size is most impactful in improving model accuracy, we can leverage techniques like spatial awareness and transfer learning to obtain reasonable performance when training data is sparse. These insights show that, with an abundance of data, light-weight models can be as performant as their heavy-weight counterparts in cellular analysis.

Table of Contents

List of FIGURES

List of TABLES

Chapter 1

INTRODUCTION

1.1 Background

Live-cell imaging experiments, in which individual cells are identified and tracked over time, reveal the spatial temporal information of cellular processes and provide a quantitative understanding of cellular dynamics. The analysis of such dynamic data involves three main phases: cleaning the fluorescent or bright-field microscopy images through background subtraction and drift correction, segmenting each cell across all frames of the time series data, and finally linking cell segmentations to their identities to determine their lineage. Prior work has demonstrated that deep learning techniques are advantageous in automating this pipeline to create more quantitative and cohesive records of biological images at the resolution of single cells. Moreover, these methods are proving useful for analyzing other categories of biological imaging experiments, including multiplexed imaging of tissues.

Recent advances in deep learning methods now allow us to perform image segmentation with impressive levels of accuracy. In this thesis, we explore the performance of deep learning methods for instance segmentation - i.e. class agnostic classification and identification of objects at the pixel-level - as this is the category single cell segmentation belongs to. A variety of deep learning methods and architectures have arisen in recent years, and broadly speaking they vary by post-processing method and architecture. Pixelwise segmentation methods assign each pixel as being either inside a cell, outside a cell, or at the cell boundary; thresholding is used to transform these predictions into instance labels. Deep watershed segmentation treats the image as a topographic map where detections are basins - deep learning serves to transform the raw images into exactly such a topographic map. Mask R-CNN models generate proposals of object locations then use these proposals to predict the object class, refine the bounding box, and create a pixel-level mask of the object. Vector embedding methods use low-dimensional, learned continuous vector representations of discrete variables to cluster neighbors in the embedding space. The architectures that implement these different approaches for segmentation of medical and natural images vary and include U-Net, Feature Pyramid Networks (with a variety of backbones), and DeepLab.

Ongoing efforts in the field have sought to improve model accuracy by modifying the model architecture [3], finding alternative representations for cell segmentations [18], or by using model ensembles [12]. While model accuracy is important, it is only one aspect of what makes a model useful. A model's memory footprint and its inference speed are important considerations for deployment, as they limit which hardware a model can be deployed on. The approaches to model development mentioned above often lead to bulky models that have reduced inference speeds. Here, we propose that an alternative path to model development is through big data. As demonstrated in this thesis, light-weight models can be as performant as large ones if trained on a sufficiently large and diverse dataset. These models can circumvent the myriad of issues that arise when deploying larger models. While the lack of training data has made such an approach impossible, recent work by our lab has resulted in a dataset of sufficient size to explore the value of big data for developing deep learning models for segmenting live-cell imaging data. Our dataset consists of movies of over 10,000 unique cells where each movie is 30 frames and is annotated at the pixel level to provide lineage information of appearing cells. In total, our dataset contains 191,608 instance segmentations.

In this thesis, we make use of this novel dataset to understand the impact of 6 different factors on model performance - temporal information, spatial awareness, transfer learning, segmentation schema, model capacity, dataset size. Temporal information is challenging to incorporate into model training as it requires whole movies to be annotated frame-by-frame. Fortunately, we have exactly such a dataset. To measure its importance, we create models with convolutional-recurrent layers that encode temporal information and train them with sequences of frames, with the hope that information from previous frames will better inform its prior for subsequent frames. Spatial awareness is another aspect we explore - it is known that deep learning models only make use of local information and are unaware of spatial relationships within images. This weakness can be mitigated by appending a positional encoding image to input data; we explore the impact of that approach here.

Transfer learning, in which models are initially trained on a large dataset and then fine tuned on a smaller dataset, is a simple method to boost model performance. Transfer learning is effective because natural datasets share many similarities in terms of shape, textures, and movement patterns. It is relatively simple to implement, as it involves using models with defined architectures and pre-trained weights. However, this method significantly limits innovation on the model architecture. As mentioned previously, segmentation schema is another area that merits investigation. Here, we explore three different schemas - pixelwise classification [16, 21], MaskR-CNN [5], and a deep watershed approach [8]. Model capacity is another variable we investigate by training models with backbones with low (MobilenetV2) and high (Resnet50) model capacity. Lastly, we explore the impact of dataset size. Unfortunately, this can only be done retrospectively as studying the impact of big data requires big data. While training data remains a limitation for a variety of biological image types, the dataset collected by my colleagues is of sufficient size to explore the impact of dataset size and diversity.

In the following section, we describe how we train models on defined dataset splits and benchmark their accuracy across different permutations of the above parameters. We show that endowing deep learning models with spatial awareness, using transfer learning with model backbones pre-trained on ImageNet, and using larger sets of training data are effective methods of boosting model performance. We also investigate synergies that exist between these explored parameters. We also show that some approaches like using convolutional-recurrent layers have little impact on model performance.

1.2 Related Work

Here we provide an overview of prior work from others on segmentation schema for instance segmentation, incorporating spatial awareness into deep learning models, and measuring the impact of dataset size.

Cell Segmentation Methods

Initial cell segmentation methods, like U-Net and Deepcell, perform classifications at the pixel level to predict cell boundaries, cell interiors and the background [16, 21, 10]. This pixel-wise method thresholds the final probability maps to obtain segmentation masks.

Instead of thresholding, deep learning can learn the distance transform, the distance between a pixel and the image background, to use a watershed method that is common in computer vision. Here, each object instance is represented as an energy basin. Components are linked by performing a cut at a single energy level. The deep watershed transform is particularly attractive as it offers a constant runtime regardless of the number of object instances [1]. It allows for end-to-end training and fast estimates. Another more recent deep watershed approach is deep distance, which predicts both an inner distance (distance to cell center of mass) and outer distance (distance to cell boundary - e.g. the energy basin in the previous approach) transform of the instance masks. By using the inner distance prediction to identify cell centroids, they further reduce over segmentation that can arise from having multiple extrema in a cell's energy basin representation.

Object-detection-based methods have been adapted for cell segmentations. These methods predict bounding boxes for all objects in an image and use non-max suppression (NMS) to remove redundant boxes. The Mask-RCNN model, developed in 2017 for semantic segmentation and natural object segmentation, is one of the most accurate methods with a bounding box approach. It extends the Fast-RCNN model with an ROI-Align layer that preserves exact spatial locations [7]. The output of the ROI-Align layer allows for accurate construction of segmentation masks, and with a new mask head, which enables independent mask and class predictions. Mask-RCNN models with ResNet-50 or ResNet-101 based feature pyramid network (FPN) backbones have been shown to have greater speed and accuracy [6].

Other segmentation methods that make use of NMS include StarDist which localizes cell nuclei via star-convex polygons [18]. StarDist does not have an explicit segmentation step. Instead, the segmentation is obtained by combining distances from the detected center to boundaries in 32 radial directions. Like Mask-RCNN, it uses NMS to suppress overlapping detections. However, this method provides a finer shape representation than bounding boxes does and does not require shape refinement.

It is also possible to treat the cell segmentation problem as a vector embedding problem [15]. A discriminative loss function assigns pixels in the same instance to the same vector and pixels in different instances to different vectors. These vectors are then clustered in the embedding space to identify objects.

A recent algorithm that falls within this vector-embedding category is Cellpose [20], which applies a reversible transform to instance masks to create vector flows and then uses a deep learning model to predict this vector flow field directly from images. Their approach makes use of a diverse set of cellular segmentations, as they sought to segment brightfield and fluorescent cytoplasmic images of cells rather than the cell nuclei.

Improving Spatial Awareness

Deep convolutional networks generally lack spatial awareness which can result in blurry edges and noisy segmentations. Most approaches tune the network architecture to combine low-level features with high-level features to improve boundary detection. Other methods track object centers and regress a vector pointing to the instance's centroid for each pixel [1]. This allows class and spatial relationship information to be determined for each pixel in each object for instance segmentation. This information can also be encapsulated by creating spatial embeddings of pixels and using a loss function that clusters together vectors belonging to the same instance [14]. Here, center localization and clustering is performed in post processing.

Analysis of Dataset Size on Model Performance

It is generally accepted the deep learning models trained on more data outperform those trained on less data [13]. Previous research has shown that high quality data can significantly benefit cell segmentation accuracy [2]. The analysis was performed on 23,165 cells from the BBBC021 dataset. With our large training set of 191,608 annotations, we extend upon this analysis.

With crowd sourcing and annotation, many digital image collections have increased rapidly in size [4]. However, for tasks like cell segmentation, the amount of publicly available data remains scarce. Techniques like geometric transformations, image mixing and general adversarial network (GAN) based augmentations have been used to expand limited datasets to take advantage of the capabilities of big data [19].

Chapter 2

METHODS

2.1 Data Annotation and Splitting

We trained our models using fluorescent cytoplasm images of four different cell lines -- RAW264, HeLa S3, NIH 3T3 and HEK293. For the temporal analysis, we trained models on each of the cell lines. For the other analyses, we combined all four cell lines into a single dataset consisting of 191608 unique cells.

With the combined nuclear data, we first randomized the data with one of three fixed seeds. After performing the train-test split, we selected the first n movies in the training set for training. Here, n/N is our desired dataset fraction with N being the total number of movies in our entire nuclear training set. For example, a larger dataset fraction of 0.5 would contain all the movies in a smaller dataset fraction of 0.25. Doing the data split in this manner ensured that dataset diversity was a monotonically increasing function of dataset size. Alternative approaches to splitting do not have this guarantee. Once split, the datasets were resized into 128x128 tiles, with overlaps. For each seed and each dataset fraction, we evaluated the performance of each model based on the number of unique annotations (the number of cells across all frames prior to resizing).

2.2 Model Architecture

All models are composed of a backbone attached to a Feature Pyramid Network (FPN) head [9]. The FPN architecture features top-down and bottom-up pathways with lateral connections (Fig 1). The bottom-up pathway computes feature maps at multiple scales. In the top-down pathway, higher resolution features are upsampled from spatially coarser, but semantically stronger, feature maps from higher pyramid levels. Lateral connections merge the feature maps of the same spatial size from the two pathways. A 3x3 convolution is applied to the merged maps to reduce the aliasing from upsampling.

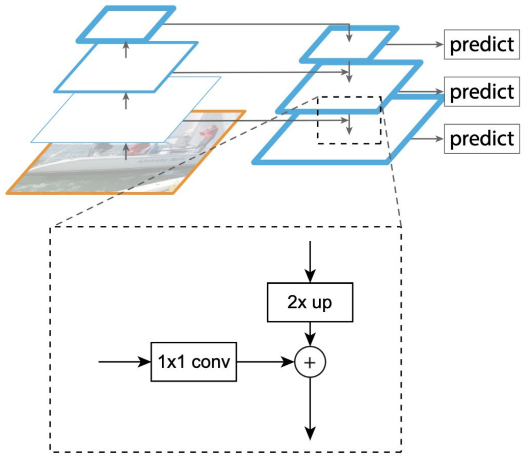


Fig 1. Feature Pyramid Network from Lin et al. [9]

Pixelwise and deep watershed models have semantic heads, which take as input the top of the feature pyramid and produce an output (via multiple rounds of upsampling and convolutions) with the same dimensions as the input image. The top layer of a semantic head has either a relu activation or softmax activation for regression and classification tasks respectively. For RetinaMask models, the FPN is extended with two 1x1 convolutions after the 3x3 convolutions to create classification and regression heads. The object detection criterion and bounding box regression target are defined with respect to predefined anchors. Anchors are boxes of a given size that tile an image and the classification head of a RetinaMask model seeks to identify which anchors contain an object and which ones do not. We used anchor sizes of 16 and 32 pixels in P3 and P4 layers, respectively. We use the criteria from the original RetinaNet paper to identify anchors that contain objects; anchors that have an intersection over union (IoU) greater than 0.5 with a ground truth object's bounding box are considered positive, anchors with an IoU less than 0.4 are considered negative, and all other anchors are ignored for the purposes of model training.

Each of the aforementioned models has a backbone architecture of either ResNet50 [6] or MobileNetV2 [17]. ResNet features residual blocks with skip connections that add outputs of previous layers with stacked layers. MobileNetV2 is more lightweight than ResNet. It has three-layer residual blocks with depthwise separable convolutions. There are bottlenecks between each block and shortcuts between these bottlenecks to add the input of one bottleneck to the output of another. Both types of backbones are effective feature extractors; benchmarking from ImageNet demonstrates that ResNet50 has higher model capacity at the cost of reduced inference speed.

2.3 Model Training

For each segmentation schema, we train the models a different, but relevant, loss function but otherwise equal training parameters. For deep watershed models, we use the Mean Squared Error (MSE). Pixelwise models are trained using weighted categorical cross entropy. The weights for the three classes are computed on-the-fly and are used to weight the Keras categorical cross-entropy. Formally, we define this as,

$$L = -\sum_{c=1}^{C} w_c * y_{i,c} * log(p_{i,c}) \tag{1}$$

Where C is the number of classes, w_c is the weight of class c, $y_{i,c}$ is a binary indicator denoting the class of the sample and $p_{i,c}$ is the predicted probability between 0 and 1 for that class and sample. For the RetinaMask models, we use a focal loss for the classification head, a smooth L1 loss for the regression head, and a binary cross-entropy for the mask prediction head. Each loss is given equal weight, but is normalized by the number of positive anchors.

For all models, we used the Adam optimizer with a learning rate of 10^{-5} and clip norm of 0.001, batch size of 4, and L2 regularization strength of 10^{-5}. For the temporal analysis, models trained for 10 epochs, and for all other analyses models trained for 15 epochs on a NVIDIA V100 graphics card.

We trained the following models for our analyses:

- Temporal information analysis
  - RetinaMask models trained on four different nuclear datasets (NIH 3T3, HEK 293, HeLa S3, Raw 264) with convolutional modes of None, Gated Recurrent Units (GRU), Long Short Term Memory (LSTM) and time-axis convolution (Conv3D)
- Spatial awareness analysis
  - Watershed models trained on the full dataset without and without location layer for three seeds
- Transfer learning analysis
  - Pixelwise, RetinaMask and Watershed models trained on the full dataset without and without ImageNet weights for three seeds
- Dataset analysis
  - Pixelwise, RetinaMask and Watershed models for dataset fractions of 0.01, 0.1, 0.25, 0.5 and 1 for three seeds

2.4 Benchmarking

We post-processed model predictions with and without removing small objects. Removing small objects both removes artifacts and also removes partial cells at the border. While we present both sets of results here for completeness, we believe removing small objects represents a more realistic view of model performance in practice, as images are usually considerably larger than the 128x128 patches analyzed here, and these partial cells are a smaller fraction of the total number of cells in these larger images.

We used both pixel-based and object-based approaches in segmentation benchmarking. The object-based approach compares ground truth and prediction segmentations to identify segmentation errors [11]. We first link prediction and ground truth segmentations based on object overlap, the *iou* (intersection over union of those two cells). The iou of leaving a ground truth cell unassigned (missed detection) or a prediction cell unassigned (gained detection) is 0.4. For a ground truth and predicted cell to be linked, they must have an iou of at least 0.6.

All the unassigned cells are then gathered for a graph construction to classify error types (Table 1). A summary of this approach and a descriptive figure (Fig 2), both of which are taken from our lab's prior manuscript [11], are given below:

"We view each cell as a node and examine all pairs of unassigned ground truth cells and unassigned predicted cells; we link two cells if they have an iou greater than 0.1. We then extract all the subgraphs of this graph and categorize them into three groups. The first group consists of all subgraphs that have a single node (i.e. the highest degree of any node is 0). If the node is a ground truth cell this corresponds to a missed detection (i.e. a false negative); if the node is a predicted cell this corresponds to a gained detection (i.e. a false positive). The next group consists of all subgraphs where the highest degree node has degree 1. Each subgraph in this group has one node that corresponds to a ground truth cell and one node that corresponds to a prediction cell. Because these cells were not assigned in the first round of graph construction, they correspond to a missed detection and a gained detection. The third group are all subgraphs with a node that has degree > 1. This group can be further divided into three subgroups based on the type and uniqueness of the highest degree node. If the highest degree node is a ground truth cell and is unique, then it corresponds to a splitting error. If the highest degree node is a predicted cell and is unique, it corresponds to a merge error. If the highest degree node is not unique, then it falls into a third class which we call catastrophes."

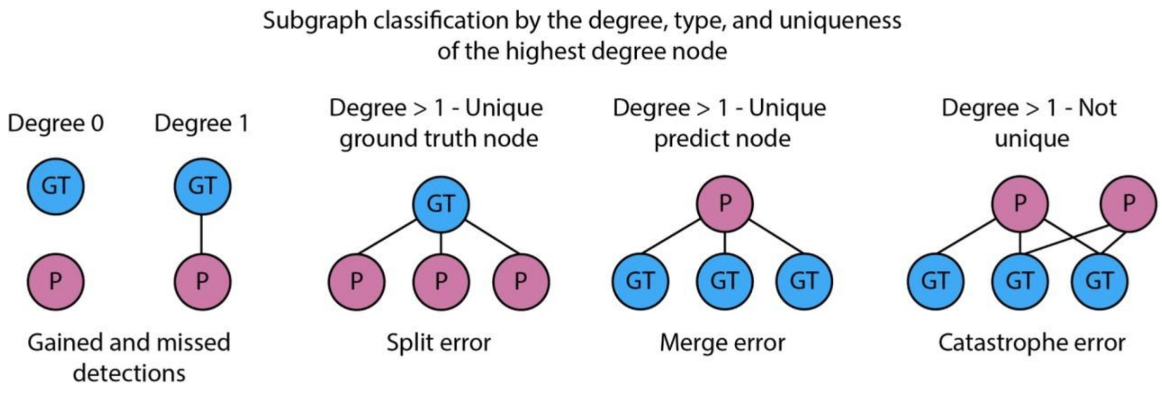

Subgraph classification by the degree, type, and uniqueness of the highest degree node
Degree 0
Degree 1
Degree > 1 - Unique ground truth node
Degree > 1 - Unique predict node
Degree > 1 - Not unique
GT
P
Gained and missed detections
Split error
Merge error
Catastrophe error

Fig 2. Identification of merge, split, and catastrophe errors in cell segmentations though subgraph classification. Reprinted from "Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning"

Metric/ Error Type	Definition
Precision	The proportion of correct positive detections TP/(TP + FP)
Recall	The proportion of actual positives that are correctly identified TP/(TP + FN)
F1 Score	2 * Recall * Precision / (Recall + Precision)
Merges	Two or more cell is detached as a single cell.
Splits	A single cell is detected as two or more cells.
Catastrophes	Both splits and merges are involved.
Jaccard Index	The size of the intersection divided by the size of the union of the sample sets (ground truth and predictions).

Table 1. Definitions of metrics and error types used in benchmarking. TP - true positives; FP - false positives; FN - false negatives.

Chapter 3

RESULTS

3.1 Temporal Information

We aimed to enhance the performance of popular deep learning algorithms in analyzing dynamic cellular data by modifying their architecture with various memory units. We postulated that temporal units allow models to stochastically retain information from previous frames which helps them better learn and predict on future frames. As such, we trained RetinaMask models with modified ResNet50, Featurenet (a conv-net in which the input image is convolved with a set of filters. Each filter acts as a local feature extractor) [21] and MobileNetV2 backbones on a combination of cell lines and benchmarked them on each individual cell line. For the MobilenetV2 backbone trained with 3 frames per batch and the ResNet50 backbone trained with 5 frames per batch, temporal information provided marginal improvements across several cell lines (Tables 2 and 3). For the ResNet50 backbone trained with 3 frames per batch, temporal information resulted in a regression in performance. Overall, the addition of such temporal information in the training of these neural nets has no consistent effect on the accuracy of model predictions.

Trained Using Frames Per Batch = 3		RetinaMask with FeatureNet Backbone			
		F1 Score			
		Conv3D	GRU	LSTM	None
Live-Cell Imaging Datasets	3T3	0.8825	0.8829	0.8780	0.8952
	HEK293	0.8093	0.8030	0.8081	0.8239
	HeLa	0.6498	0.8734	0.8648	0.8348
	RAW264.7	0.8875	0.8850	0.8830	0.8949

Trained Using Frames Per Batch = 3		RetinaMask with ResNet50 Backbone			
		F1 Score			
		Conv3D	GRU	LSTM	None
Live-Cell Imaging Datasets	3T3	0.8839	0.8849	0.8877	0.8892
	HEK293	0.8209	0.8205	0.8273	0.8313
	HeLa	0.9016	0.8914	0.8841	0.8849
	RAW264.7	0.8978	0.8899	0.8943	0.9082

Trained Using Frames Per Batch = 3		RetinaMask with MobileNetV2 Backbone			
		F1 Score			
		Conv3D	GRU	LSTM	None
Live-Cell Imaging Datasets	3T3	0.8568	0.8774	0.8711	0.8173
	HEK293	0.8269	0.8402	0.8397	0.8047
	HeLa	0.7927	0.8602	0.8614	0.8235
	RAW264.7	0.8838	0.8956	0.8964	0.8530

Table 2. Use of temporal information and model accuracy for models trained with 3 frames per batch

Trained Using Frames Per Batch = 5		RetinaMask with FeatureNet Backbone			
		F1 Score			
		Conv3D	GRU	LSTM	None
Live-Cell Imaging Datasets	3T3	0.8673	0.8569	0.8534	0.8646
	HEK293	0.7916	0.8002	0.7941	0.8274
	HeLa	0.8310	0.8254	0.8425	0.8674
	RAW264.7	0.8779	0.8758	0.8765	0.8931

Trained Using Frames Per Batch = 5		RetinaMask with ResNet50 Backbone			
		F1 Score			
		Conv3D	GRU	LSTM	None
Live-Cell Imaging Datasets	3T3	0.8694	0.8669	0.8710	0.8703
	HEK293	0.8198	0.8196	0.8136	0.8243
	HeLa	0.8915	0.8890	0.8842	0.8849
	RAW264.7	0.8955	0.8917	0.8924	0.8932

Trained Using Frames Per Batch = 5		RetinaMask with MobileNetV2 Backbone			
		F1 Score			
		Conv3D	GRU	LSTM	None
Live-Cell Imaging Datasets	3T3	0.8162	0.8285	0.8508	0.8498
	HEK293	0.8038	0.8242	0.8172	0.8334
	HeLa	0.8152	0.7652	0.8682	0.8648
	RAW264.7	0.8810	0.8848	0.8695	0.8921

Table 3. Use of temporal information and model accuracy for models trained with 5 frames per batch

3.2 Spatial Awareness

We initialized the watershed model with ImageNet weights and trained them with and without the location layer. We found that, regardless of the size of the training data, location information improves the performance of models with ResNet50 backbones but regresses that of models with MobilenetV2 backbones (Tables 4 and 5). These changes in performance are relatively small, which suggests that adding spatial awareness does not have a substantial impact on model performance.

Model Backbone		Watershed Model Trained on 10% of Data Before Removing Small Objects	
		F1 Score Mean	F1 Score SD
ResNet50	w/ Location	0.8795	0.0044
	w/o Location	0.8743	0.0043
MobilenetV2	w/ Location	0.8565	0.0076
	w/o Location	0.8667	0.0046

Model Backbone		Watershed Model Trained on Full Data Before Removing Small Objects	
		F1 Score Mean	F1 Score SD
ResNet50	w/ Location	0.8968	0.0062
	w/o Location	0.8862	0.0143
MobilenetV2	w/ Location	0.8843	0.0105
	w/o Location	0.8871	0.0010

Table 4. Spatial awareness and performance of watershed models before removing small objects.

Model Backbone		Watershed Model Trained on 10% of Data After Removing Small Objects	
		F1 Score Mean	F1 Score SD
ResNet50	w/ Location	0.9496	0.0005
	w/o Location	0.9481	0.0023
MobilenetV2	w/ Location	0.9374	0.0021
	w/o Location	0.9424	0.0021

Model Backbone		Watershed Model Trained on Full Data After Removing Small Objects	
		F1 Score Mean	F1 Score SD
ResNet50	w/ Location	0.9584	0.0020
	w/o Location	0.9540	0.0054
MobilenetV2	w/ Location	0.9525	0.0027
	w/o Location	0.9532	0.0016

Table 5. Spatial awareness and performance of watershed models before removing small objects.

3.3 Transfer Learning

To test the benefit of transfer learning, we initialized one batch of models with pre-trained ImageNet weights and one without. Our results are shown in Tables 6 and 7. We found that for all models, transfer learning significantly improves the accuracy and consistency of predictions. We note that the watershed model in this analysis used a location layer. These results demonstrate that transfer learning can be an effective means of improving model performance without altering model complexity or training time.

Model Backbone		Pixelwise Model Trained on Full Data Before Removing Small Objects	
		F1 Score Mean	F1 Score SD
ResNet50	w/ Transfer Learning	0.8047	0.0362
	w/o Transfer Learning	0.7761	0.0149
MobilenetV2	w/ Transfer Learning	0.7858	0.0374
	w/o Transfer Learning	0.7572	0.0069

Model Backbone		RetinaMask Model Trained on Full Data Before Removing Small Objects	
		F1 Score Mean	F1 Score SD
ResNet50	w/ Transfer Learning	0.8751	0.0025
	w/o Transfer Learning	0.8559	0.0043
MobilenetV2	w/ Transfer Learning	0.8638	0.0149
	w/o Transfer Learning	0.8489	0.0018

Model Backbone		Watershed Model Trained on Full Data Before Removing Small Objects	
		F1 Score Mean	F1 Score SD
ResNet50	w/ Transfer Learning	0.8968	0.0062
	w/o Transfer Learning	0.8903	0.0048
MobilenetV2	w/ Transfer Learning	0.8843	0.0105
	w/o Transfer Learning	0.8860	0.0024

Table 6. Transfer learning and model accuracy before removing small objects

Model Backbone		Pixelwise Model Trained on Full Data After Removing Small Objects	
		F1 Score Mean	F1 Score SD
ResNet50	w/ Transfer Learning	0.9026	0.0023
	w/o Transfer Learning	0.8961	0.0133
MobilenetV2	w/ Transfer Learning	0.8908	0.0075
	w/o Transfer Learning	0.8759	0.0040

Model Backbone		RetinaMask Model Trained on Full Data After Removing Small Objects	
		F1 Score Mean	F1 Score SD
ResNet50	w/ Transfer Learning	0.9524	0.0027
	w/o Transfer Learning	0.9351	0.0038
MobilenetV2	w/ Transfer Learning	0.9487	0.0009
	w/o Transfer Learning	0.9277	0.0016

Model Backbone		Watershed Model Trained on Full Data After Removing Small Objects	
		F1 Score Mean	F1 Score SD
ResNet50	w/ Transfer Learning	0.9584	0.0020
	w/o Transfer Learning	0.9557	0.0025
MobilenetV2	w/ Transfer Learning	0.9525	0.0027
	w/o Transfer Learning	0.9481	0.0011

Table 7. Transfer learning and model accuracy after removing small objects

To understand the distinct and aggregate effects of transfer learning and spatial awareness on watershed models, we trained another batch with neither location layer or imagenet weights. The results affirm that using both location and spatial awareness is optimal for models with a ResNet50 backbone, but there is no ideal configuration for models with MobilenetV2 backbone (Table 8).

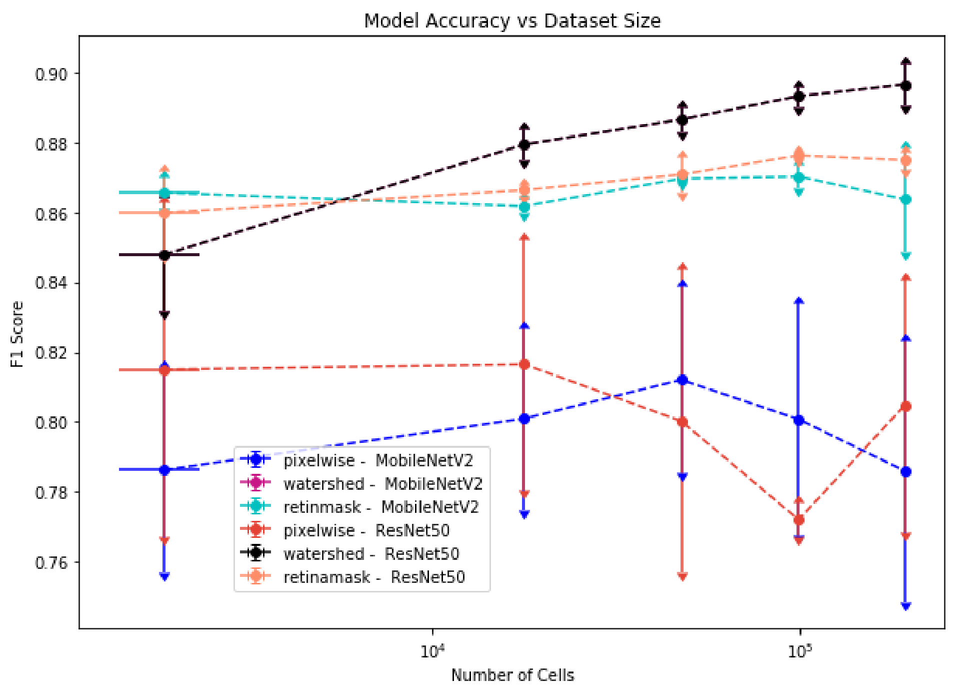
Model Backbone		Watershed Model Trained on Full Data Before Removing Small Objects	
		F1 Score Mean	F1 Score SD
ResNet50	Location & Transfer Learning	0.8968	0.0062
	Location Only	0.8903	0.0048
	Transfer Learning Only	0.8862	0.0143
	None	0.8915	0.0026
MobilenetV2	Location & Transfer Learning	0.8843	0.0105
	Location Only	0.8860	0.0024
	Transfer Learning Only	0.8871	0.0010
	None	0.8896	0.0027

Model Backbone		Watershed Model Trained on Full Data After Removing Small Objects	
		F1 Score Mean	F1 Score SD
ResNet50	Location & Transfer Learning	0.9584	0.0020
	Location Only	0.9557	0.0025
	Transfer Learning Only	0.9540	0.0054
	None	0.9555	0.0008
MobilenetV2	Location & Transfer Learning	0.9525	0.0027
	Location Only	0.9481	0.0011
	Transfer Learning Only	0.9532	0.0016
	None	0.9503	0.0010

Table 8. Combined effects of spatial awareness and transfer learning on model accuracy before (top) and after (bottom) removing small objects

3.4 Dataset Size

We found that training set size significantly impacts the performance of watershed and RetinaMask models but has variable results for pixelwise models. Our analysis also shows that watershed models perform the best while pixelwise models perform the worst across all training set sizes (Fig 3). However, we find that pixelwise models benefit the most from post processing like the removal of small objects. This is likely because pixelwise models have many artifacts in their predictions. These artifacts do not resemble any cellular morphology, so by removing detections of less than 100 pixels, we can significantly improve the accuracy by reducing the number of gained detections (Figs 4 and 5).



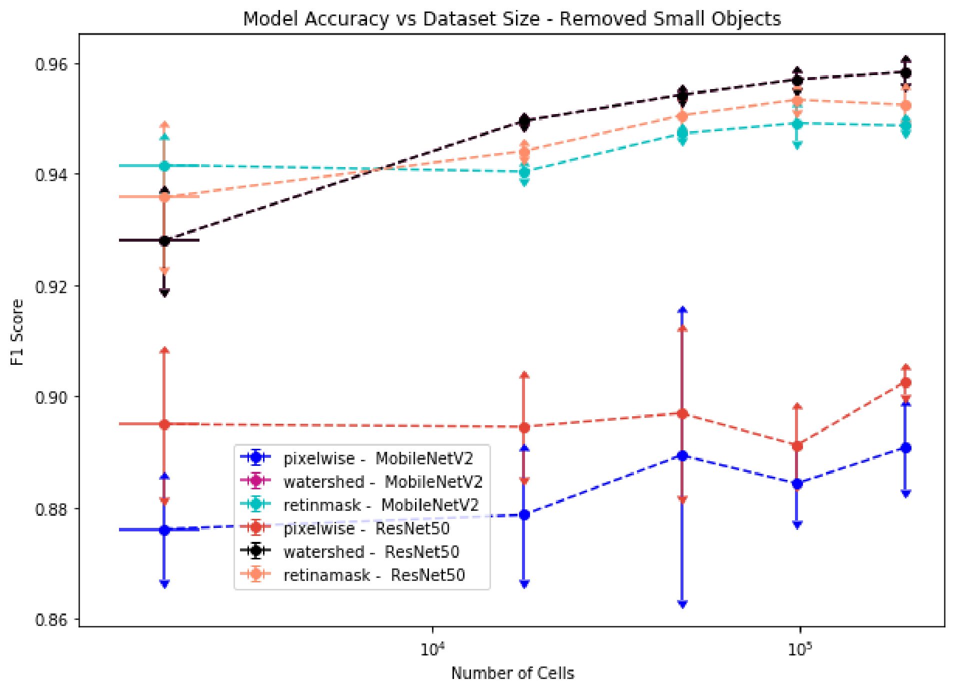
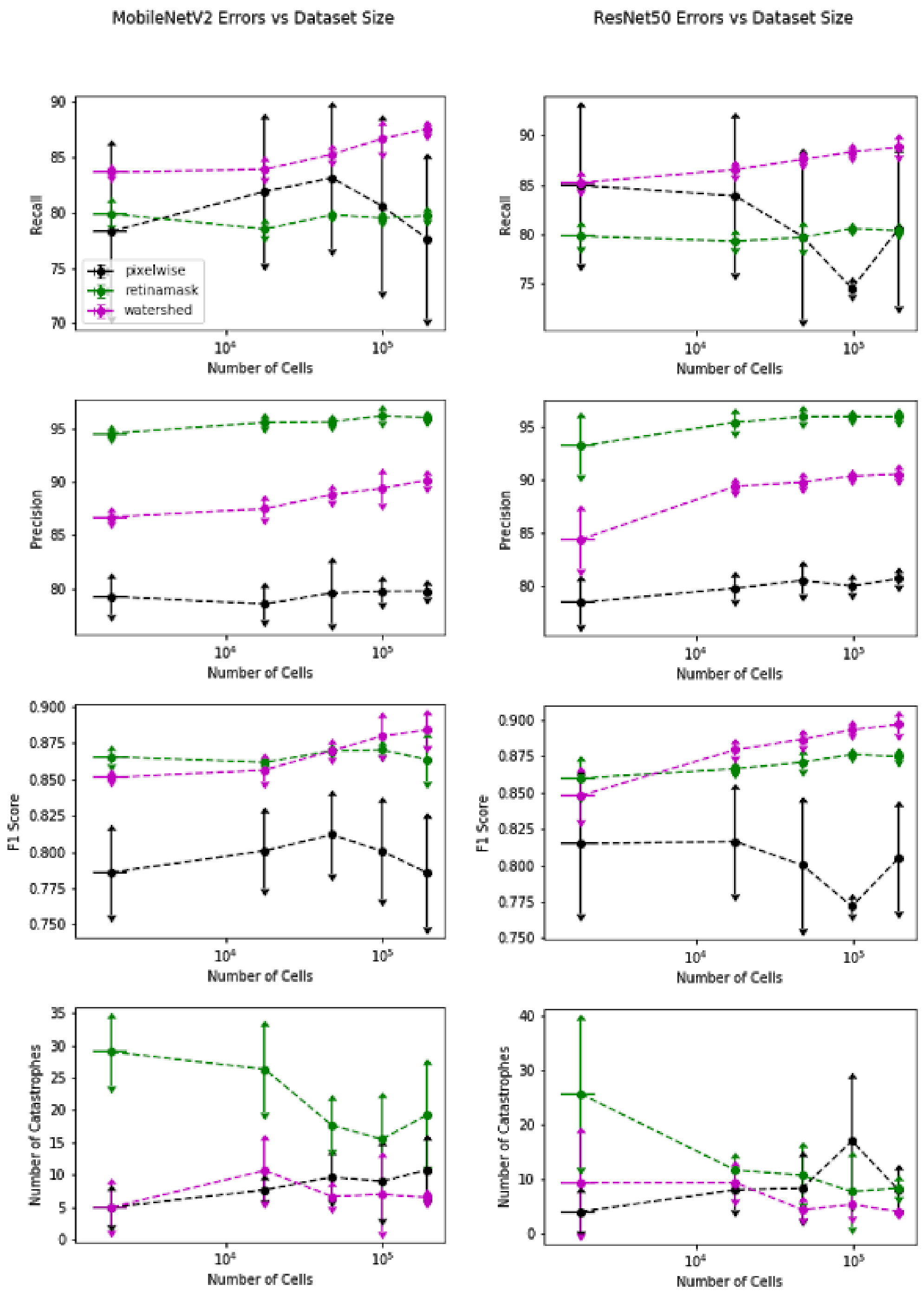


Fig 3. Model F1 score versus dataset size before (top) and after (bottom) removing small objects in post processing

We further assessed the models performance by looking at the different types of segmentation errors. We found that the improved performance in watershed and RetinaMask models results from decreased number of merges and splits (Figs 4-5). The performance of pixelwise models sees limited improvements since increasing dataset size decreases the number of merges but increases the number of splits. The improvements in one category of error are counterbalanced by regressions in another (Figs 4-5).


MobileNetV2 Errors vs Dataset Size
ResNet50 Errors vs Dataset Size
Recall
Precision
F1 Score
Number of Catastrophes
Number of Cells
pixelwise
retinamask
watershed
10^4
10^5

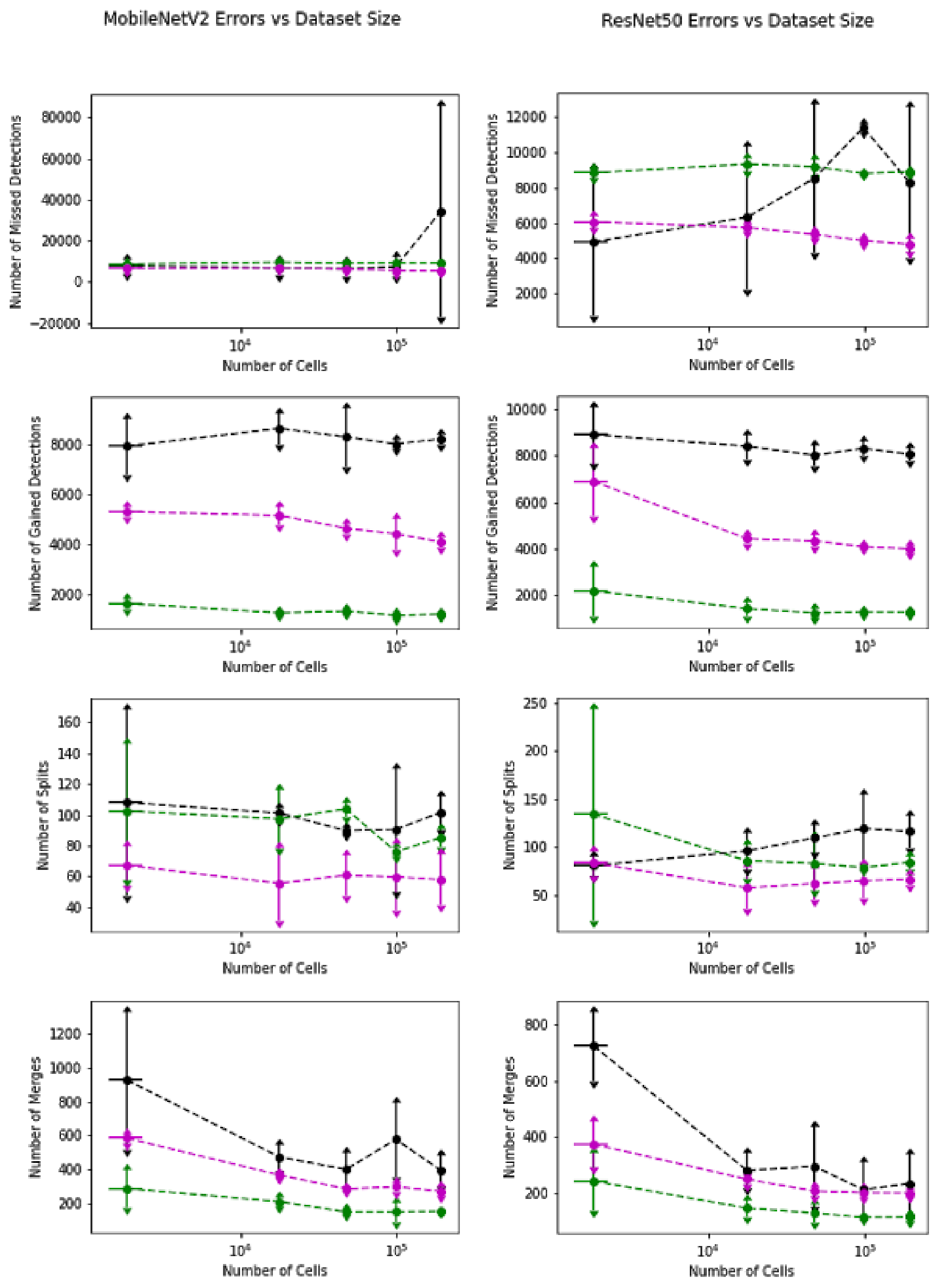
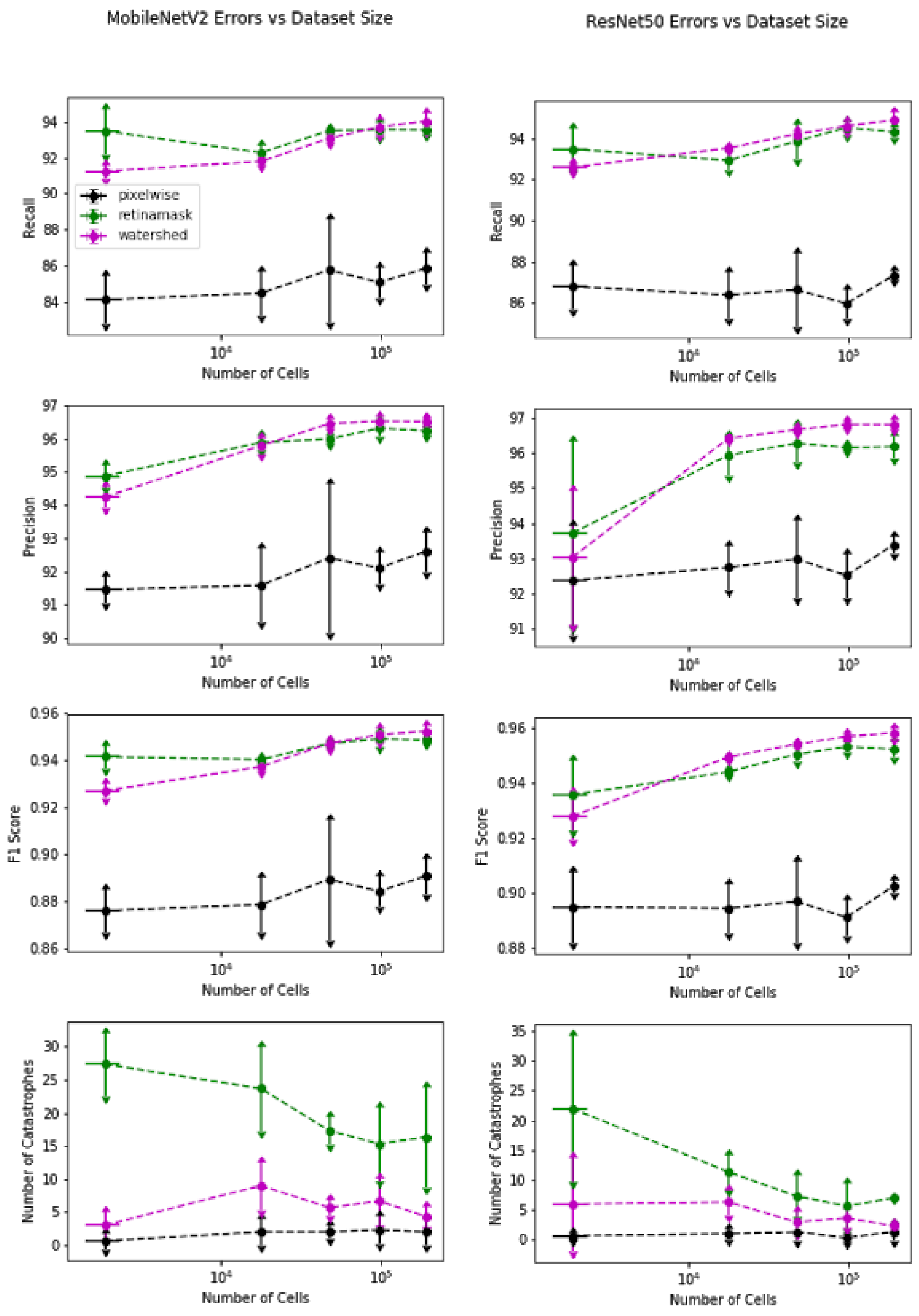


Fig 4. Model performance and errors versus dataset size before removing small objects


MobileNetV2 Errors vs Dataset Size
ResNet50 Errors vs Dataset Size
Recall
Precision
F1 Score
Number of Catastrophes
Number of Cells
pixelwise
retinamask
watershed
10^4
10^5

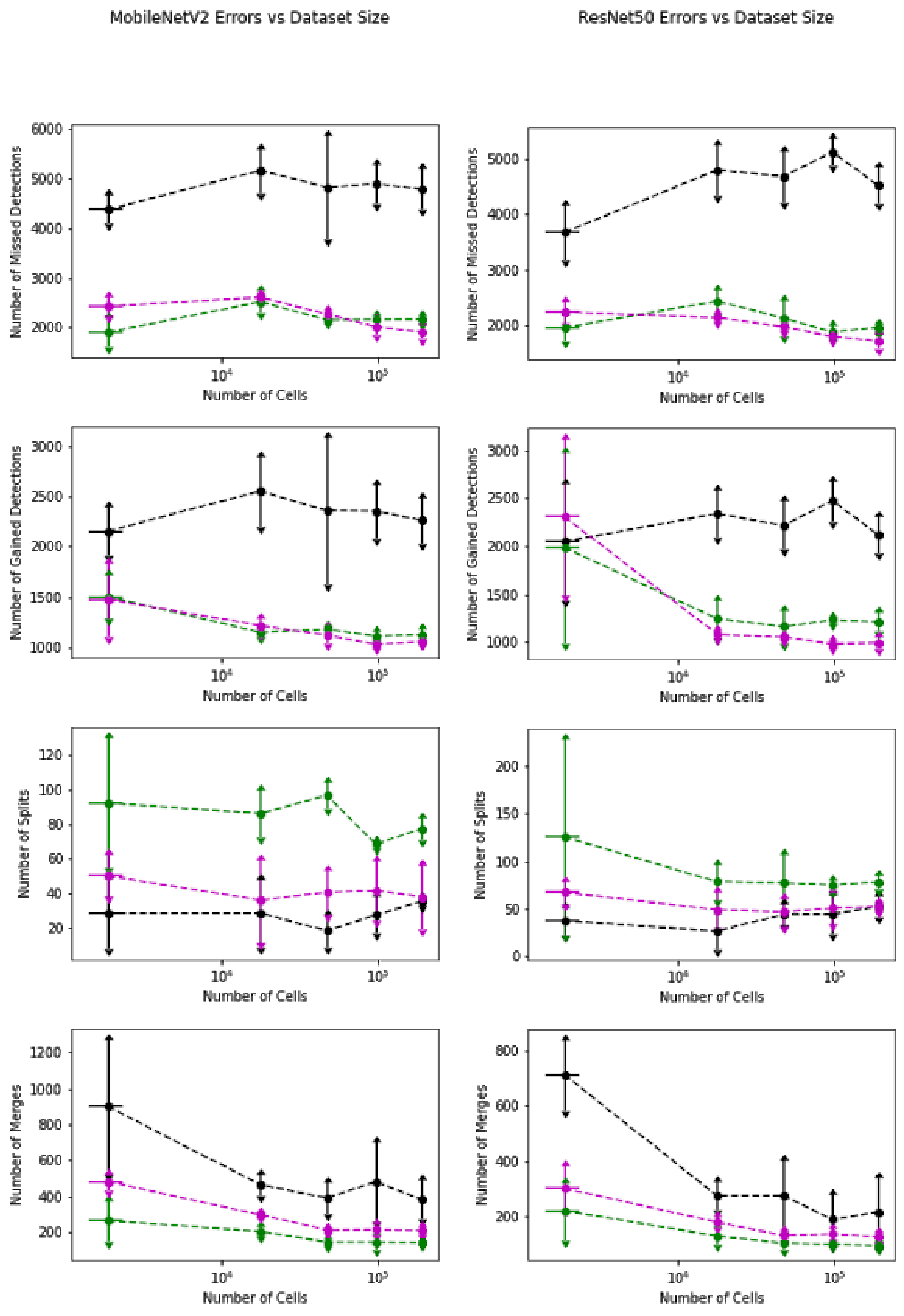


Fig 5. Model performance and errors versus dataset size after removing small objects

Chapter 7

CONCLUSION

7.1 Summary

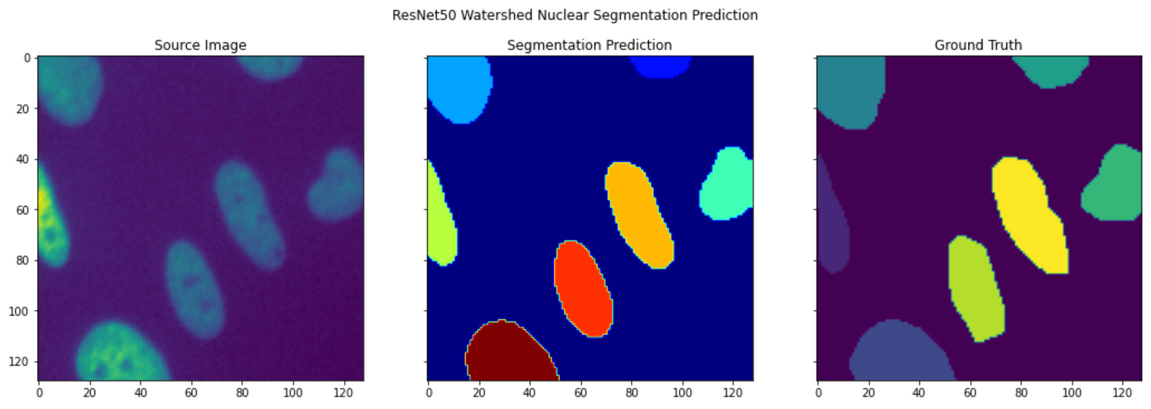
We assessed several techniques for their ability to boost Mask R-CNN model performance in cell segmentation tasks. Temporal information beyond what is included in the dataset has marginal utility. For most datasets and ways in which temporal information is included, little improvement, if not regression, was shown in model performance. On the contrary, with transfer learning, we see improvements across all model types. Spatial awareness may be beneficial for deep watershed models with ResNet backbones. When transfer learning and spatial awareness is used together, the deep watershed approach can outperform RetinaMask approach in accuracy.

We found that model performance is a monotonically increasing function of dataset size for all model types, however, the exact relationship depends on the model type. Pixelwise models reach a plateau in performance for a smaller dataset size (~0.25 dataset fraction). RetinaMask models see more improvement, but plateau after slightly more data (~0.5 dataset fraction). This is likely because the RetinaMask method predicts cells within bounding boxes. The bounding boxes impose a prior that aids with accuracy in low data settings, but becomes a hindrance with big data. Deep watershed models have the best relationship with big data. As we increase the dataset size, we see some diminishing returns in the number of missed detections, but the overall accuracy increases steadily. The improvement experienced by watershed models suggests that similar models like CellPose can also benefit from large data. CellPose and deep watershed may have different transforms, but they both rely on transformations and MSE loss.

We also find that big data can help lightweight models achieve accuracy on par with bulkier ones. We show that deep watershed models with MobileNetV2 backbones trained on large datasets can be as accurate as those with ResNet50 backbones trained on smaller datasets. These findings are insightful for future model selection and training. The MobileNetV2 backbones are sufficiently lightweight that they can be on CPUs and edge TPUs, allowing data processing to be performed locally and with low latency.

Furthermore, from a software engineering perspective, deep watershed models hold a significant advantage over RetinaMask models. In image processing, deep watershed models can use a tiling approach while RetinaMask models require the full image and an additional channel (In Tensorflow, (None, None, None, 1) for RetinaMask). This makes deep watershed models much more efficient in processing large images. RetinaMask models also require relatively more post processing since they can assign the same pixel to two different cells -- these artifacts must be removed via post processing. In addition, deep watershed models have a fixed memory footprint which reduces Out Of Memory GPU errors and is an optimal characteristic for deployment.

Although our models were trained on nuclear segmentation, they can also be trained for cytoplasm segmentation tasks (Fig 6). It would be insightful to complete analyses similar to what we have done here for other data types.



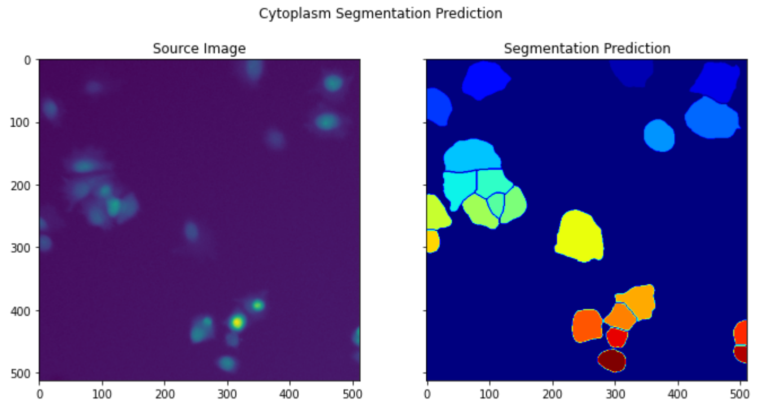


Fig 6. Nuclear (top) and cytoplasm (bottom) segmentations

7.2 Next Steps

Our analysis provides a limited look into different model enhancing techniques and our results reinforce the importance of collecting and crowdsourcing large biological datasets. As more datasets are built, these large-scale analyses of model performance would become easier and easier. It would be interesting to explore the big data impact for other data types, like cell cytoplasm and tissues. There are a lot of other factors and methods that have yet to be investigated. Architecture elements like self attention could provide an additional boost to model performance. We used supervised learning for all our models, so analysis on the impact of self-supervision can be insightful. These analyses will help the biological community create more accurate and more deployable data processing pipelines.

BIBLIOGRAPHY

1. Bai M, Urtasun R. Deep Watershed Transform for Instance Segmentation. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI: IEEE; 2017. pp. 2858–2866. doi:10.1109/CVPR.2017.305

2. Caicedo JC, Cooper S, Heigwer F, Warchal S, Qiu P, Molnar C, et al. Data-analysis strategies for image-based cell profiling. Nature Methods. 2017;14: 849–863. doi:10.1038/nmeth.4397

3. Caicedo JC, Goodman A, Karhohs KW, Cimini BA, Ackerman J, Haghighi M, et al. Nucleus segmentation across imaging experiments: the 2018 Data Science Bowl. Nat Methods. 2019;16: 1247–1253. doi:10.1038/s41592-019-0612-7

4. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L. ImageNet: A Large-Scale Hierarchical Image Database. : 8.

5. He K, Gkioxari G, Dollár P, Girshick R. Mask R-CNN. arXiv:170306870 [cs]. 2018 [cited 19 May 2020]. Available: http://arxiv.org/abs/1703.06870

6. He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. arXiv:151203385 [cs]. 2015 [cited 30 May 2020]. Available: http://arxiv.org/abs/1512.03385

7. Johnson JW. Adapting Mask-RCNN for Automatic Nucleus Segmentation. arXiv:180500500 [cs]. 2020;944. doi:10.1007/978-3-030-17798-0

8. Koyuncu CF, Gunesli GN, Cetin-Atalay R, Gunduz-Demir C. DeepDistance: A Multi-task Deep Regression Model for Cell Detection in Inverted Microscopy Images. arXiv:190811211 [cs]. 2019 [cited 28 May 2020]. Available: http://arxiv.org/abs/1908.11211

9. Lin T-Y, Dollár P, Girshick R, He K, Hariharan B, Belongie S. Feature Pyramid Networks for Object Detection. arXiv:161203144 [cs]. 2017 [cited 30 May 2020]. Available: http://arxiv.org/abs/1612.03144

10. Moen E, Bannon D, Kudo T, Graf W, Covert M, Van Valen D. Deep learning for cellular image analysis. Nature Methods. 2019;16: 1233–1246. doi:10.1038/s41592-019-0403-1

11. Moen E, Borba E, Miller G, Schwartz M, Bannon D, Koe N, et al. Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning. bioRxiv. 2019; 803205. doi:10.1101/803205

12. Moshkov N, Mathe B, Kertesz-Farkas A, Hollandi R, Horvath P. Test-time augmentation for deep learning-based cell segmentation on microscopy images. Scientific Reports. 2020;10: 1–7. doi:10.1038/s41598-020-61808-3

13. Najafabadi MM, Villanustre F, Khoshgoftaar TM, Seliya N, Wald R, Muharemagic E. Deep learning applications and challenges in big data analytics. Journal of Big Data. 2015;2: 1. doi:10.1186/s40537-014-0007-7

14. Neven D, De Brabandere B, Proesmans M, Van Gool L. Instance Segmentation by Jointly Optimizing Spatial Embeddings and Clustering Bandwidth. 2019 [cited 4 Jun 2020]. Available: https://arxiv.org/abs/1906.11109v2

15. Payer C, Štern D, Neff T, Bischof H, Urschler M. Instance Segmentation and Tracking with Cosine Embeddings and Recurrent Hourglass Networks. In: Frangi AF, Schnabel JA, Davatzikos C, Alberola-López C, Fichtinger G, editors. Medical Image Computing and Computer Assisted Intervention – MICCAI 2018. Cham: Springer International Publishing; 2018. pp. 3–11. doi:10.1007/978-3-030-00934-2_1

16. Ronneberger O, Fischer P, Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:150504597 [cs]. 2015 [cited 4 Jun 2020]. Available: http://arxiv.org/abs/1505.04597

17. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. arXiv:180104381 [cs]. 2019 [cited 30 May 2020]. Available: http://arxiv.org/abs/1801.04381

18. Schmidt U, Weigert M, Broaddus C, Myers G. Cell Detection with Star-convex Polygons. arXiv:180603535 [cs]. 2018 [cited 19 May 2020]. doi:10.1007/978-3-030-00934-2_30

19. Shorten C, Khoshgoftaar TM. A survey on Image Data Augmentation for Deep Learning. J Big Data. 2019;6: 60. doi:10.1186/s40537-019-0197-0

20. Stringer C, Michaelos M, Pachitariu M. Cellpose: a generalist algorithm for cellular segmentation. bioRxiv. 2020; 2020.02.02.931238. doi:10.1101/2020.02.02.931238

21. Valen DAV, Kudo T, Lane KM, Macklin DN, Quach NT, DeFelice MM, et al. Deep Learning Automates the Quantitative Analysis of Individual Cells in Live-Cell Imaging Experiments. PLOS Computational Biology. 2016;12: e1005177. doi:10.1371/journal.pcbi.100