

DeepFE: Deep Learning for Frog Eggs Quantification

BY: DeVision Team



Aquatic Germplasm & Genetic Resource Center (AGGRC)

- The AGGRC's mission is to lead globally in developing germplasm repositories and promoting the commercialization of genetic resources for aquatic species through collaborative efforts.
- Instead of being a centralized global repository, the AGGRC focuses on empowering various communities within aquatic species to establish their own germplasm repositories by providing necessary technologies, practices, and services.



Aquatic Germplasm & Genetic Resource Center (AGGRC)

- The main challenge addressed is the lack of standardization and reproducibility in **cryopreservation** for aquatic species, which has hindered research progress due to inefficiency and failures.
- The statistics of the eggs to be preserved is important for record.



Team DeVision

- The DeVision team is a homogenous blend of Professor Wolenski and his staff of undergraduates and graduate student at different years.

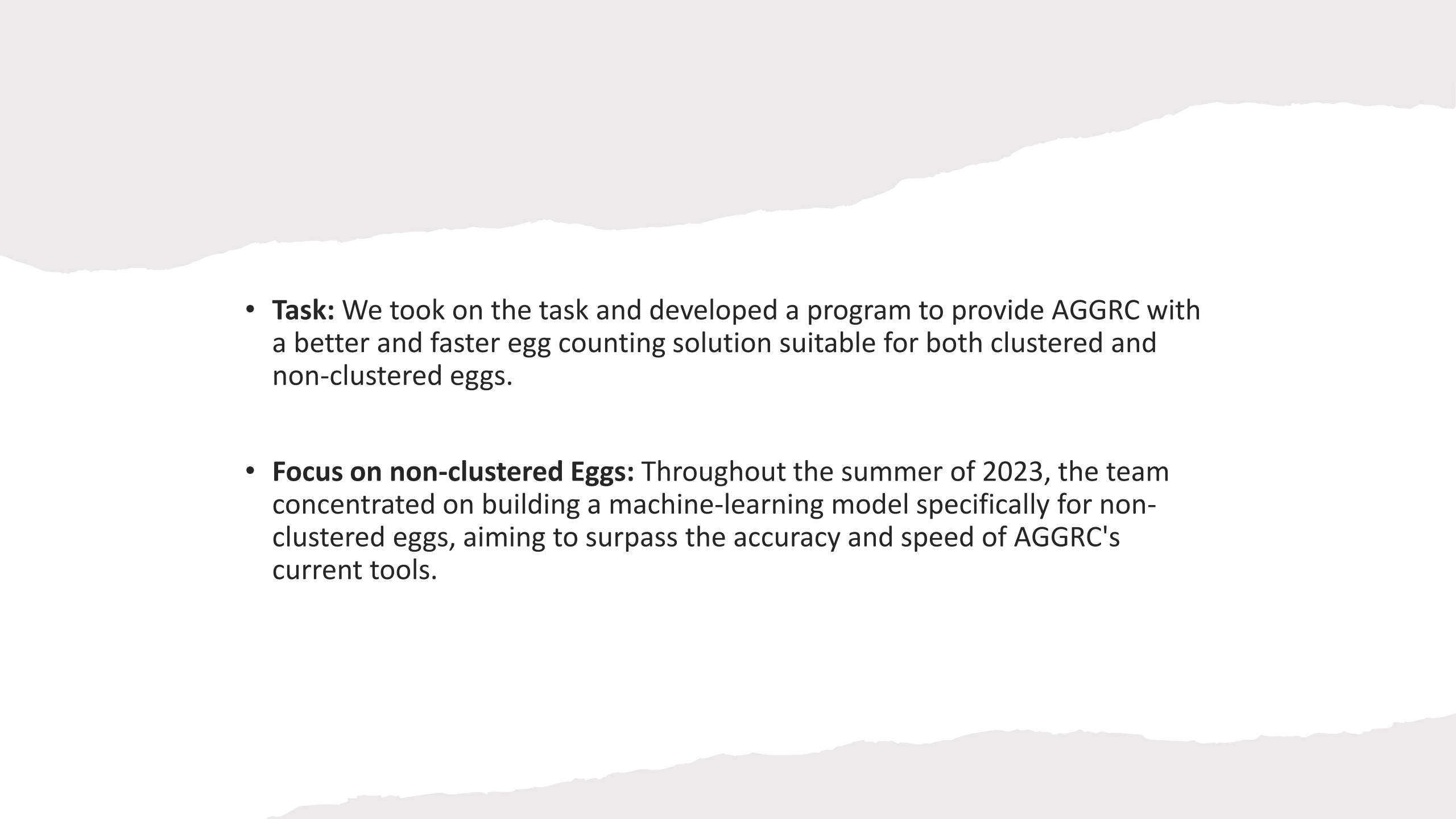


Main Goal



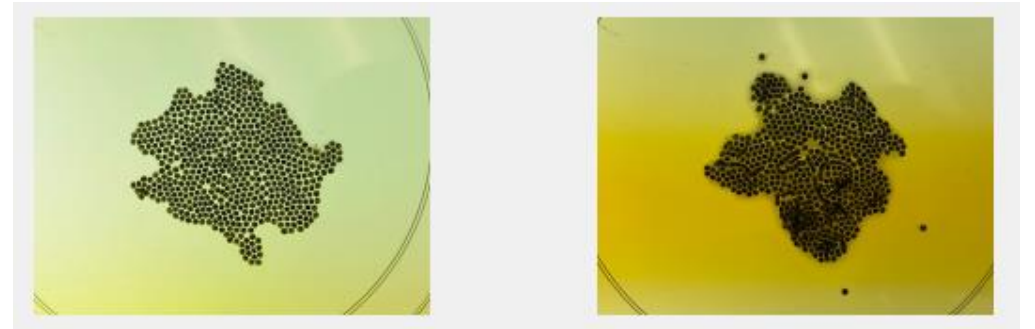
OUR MISSION

- **Project Goal:** The goal of the project is to count frog eggs in a Petri dish.
- **AGGRC's Challenge:** AGGRC sought MC²'s help to address their main challenge, which was accurately counting frog eggs, especially in a clustered environment. Existing methods were not performing well in such cases.

- 
- **Task:** We took on the task and developed a program to provide AGGRC with a better and faster egg counting solution suitable for both clustered and non-clustered eggs.
 - **Focus on non-clustered Eggs:** Throughout the summer of 2023, the team concentrated on building a machine-learning model specifically for non-clustered eggs, aiming to surpass the accuracy and speed of AGGRC's current tools.

- **Stardist Machine Learning Package:** The new model is based on the Stardist machine learning package, which utilizes image segmentation and star-convex polygons to count eggs by identifying cell nuclei.
- **Dataset Preparation:** To ensure good quality training data, the team carefully preprocessed the dataset using the FIJI application and lab kit plugins. Each egg was annotated to establish the total egg count.

- **Impressive Training Results:** During the evaluation and training process, the team conducted experiments using a dataset of frog eggs. The model achieved an impressive mean accuracy of over 90 percent.
- **Quality Assurance:** The meticulous dataset annotation and preprocessing were done to guarantee the model's training data's reliability and accuracy.



Tackling the problem



Brainstorming

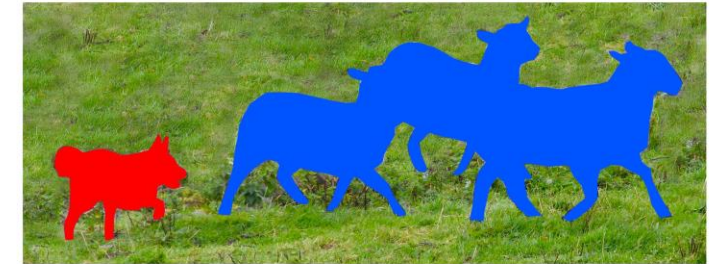
- What can help us solve the image detection problem?
 - Convolutional Neural Network used for computer vision
 - One can use the idea of Image segmentation specifically, instance or semantic segmentation to approach this problem.

Image Segmentation (Instance and Semantic)

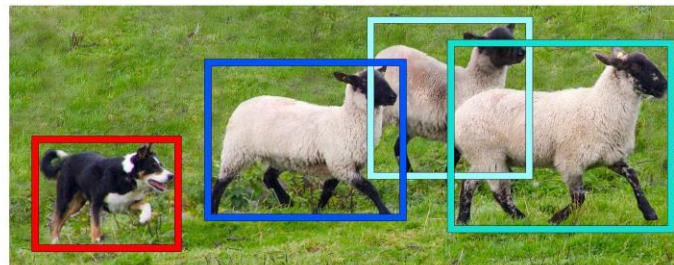
- **IS** - Identifying each object and providing them with a unique label
- **SS** - classifying each pixel in an image into predefined categories or classes



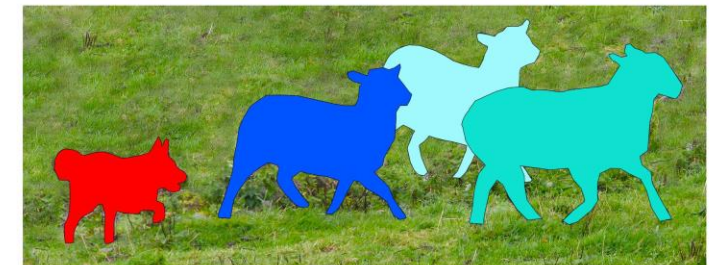
Image Recognition



Semantic Segmentation



Object Detection



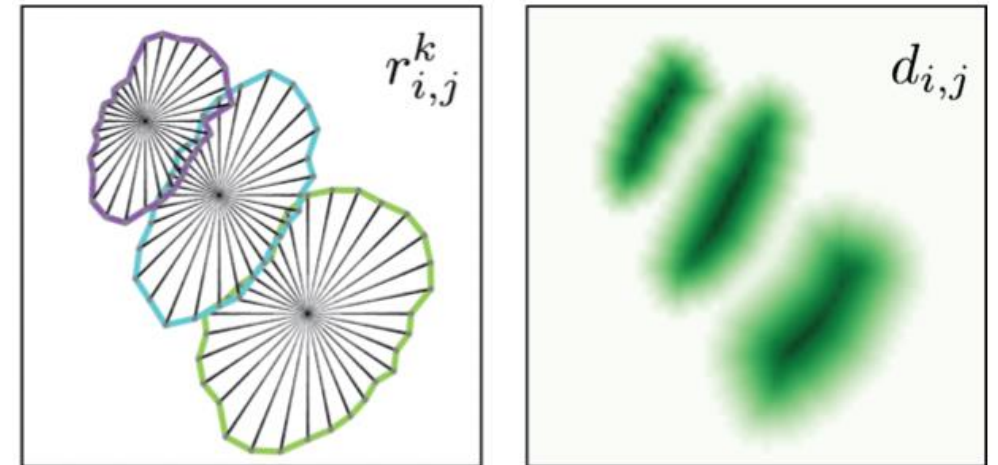
Instance Segmentation

Challenges in Cell Image Segmentation

- When dealing with objects too close to each other.
 - Misclassification of Pixels
 - Merging surrounding cell together
 - Overlapping Boundaries
 - Hard to distinguish individual cells
 - Small and Irregular Cells
 - wrong localization of cell
 - High Sensitivity to Initial Segmentation
 - Error in cell detection due to initial segment

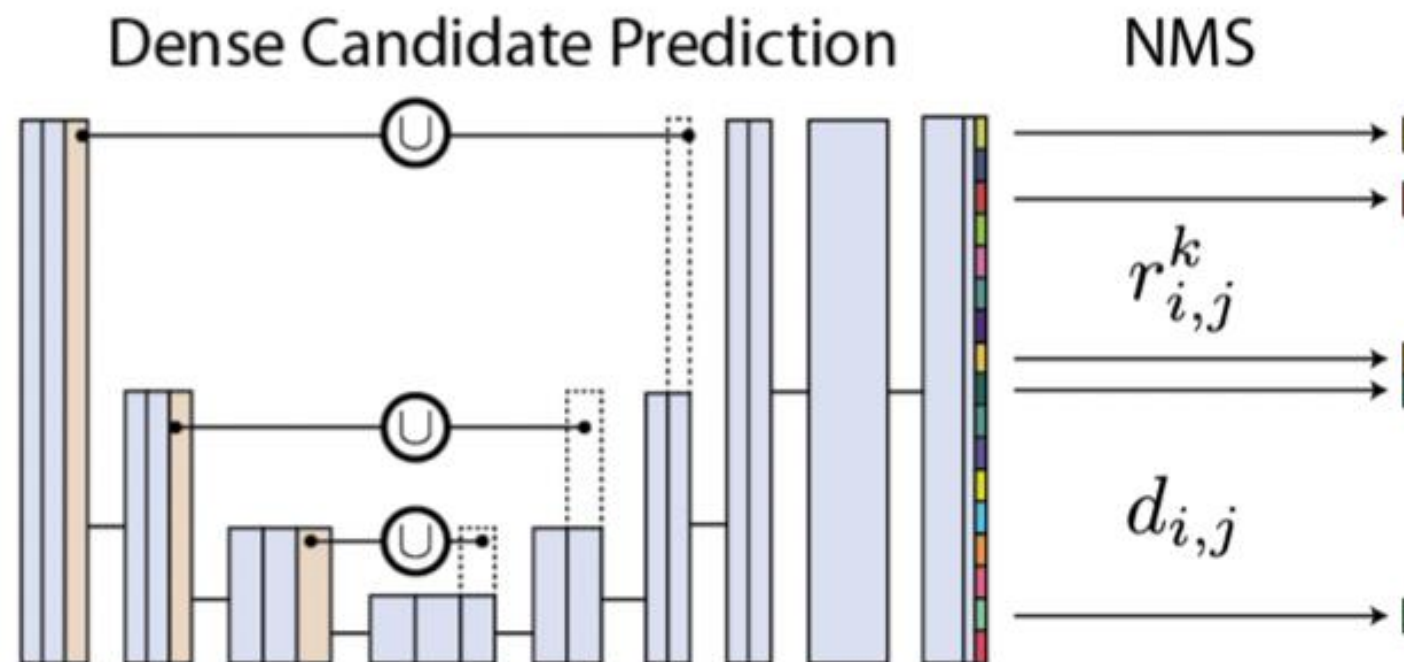
Stardist

- Uses star-convex polygons, well-suited to approximate the round shapes of cell nuclei in microscopy images.
- Utilizes a lightweight neural network based on U-Net, making it easy to train and use.
- Predicts flexible shape representations



Stardist architecture in a Nutshell

- U-net Backbone- Dense polygon prediction.
- Post processing-Polygon selection



Different models for Cell Detection

U-Net (2 class):

- Predicts "cell" and "background" classes. (bottom up)
- Approx. 1.4 million parameters.
- Uses thresholding on the cell probability map for the final result.

U-Net (3 class):

- Similar to U-Net (2 class). (bottom up)
- Adds "cell boundary" class to differentiate crowded cells with touching borders.

Mask R-CNN:

- State-of-the-art instance segmentation.
- Combines bounding-box region proposals and mask segmentation (Top-down approach).
- Approx. 45 million parameters.
- Involves grid search for hyper-parameters.

StarDist:

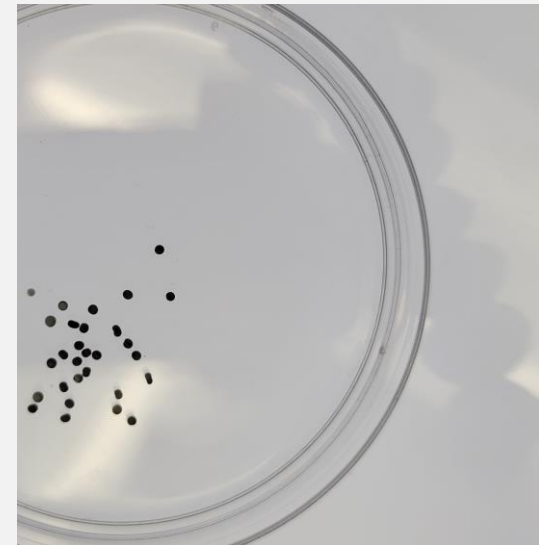
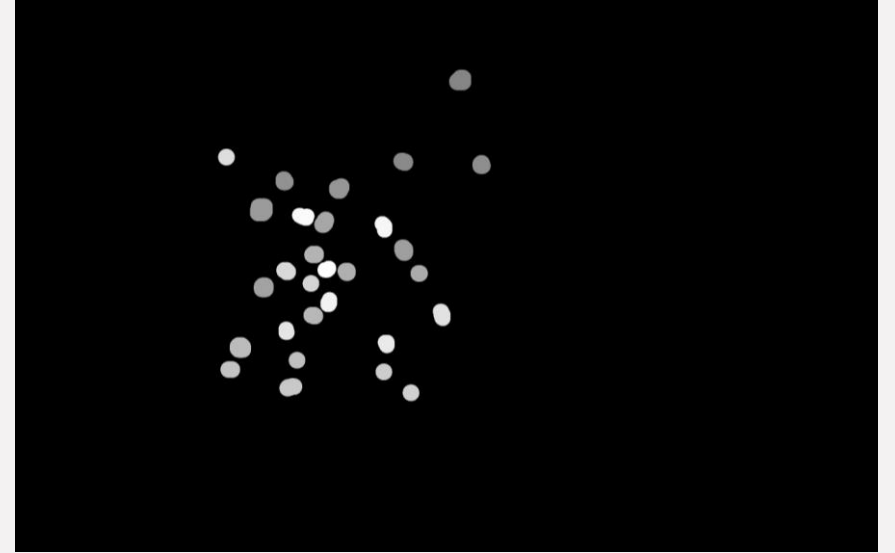
- Uses 32 radial directions with U-Net backbone.

Performance comparison of the models

- Outperforms other methods in predicting more plausible cell shapes (e.g., no holes or ridges).
- Competitive with state-of-the-art Mask R-CNN, while having significantly fewer parameters and being easier to train and use.
- Requires minimal hyper-parameter tuning for good results.

More on stardist

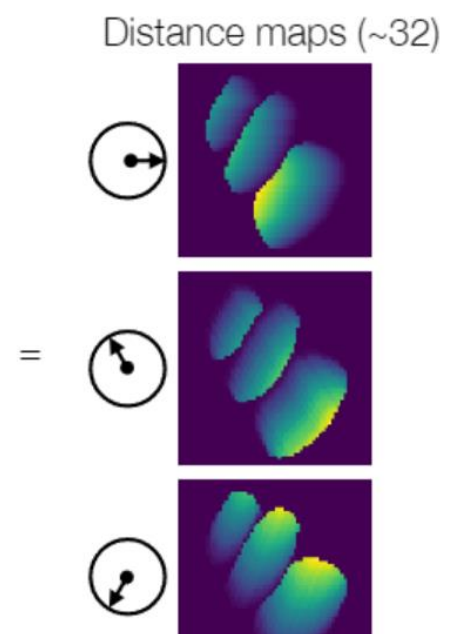
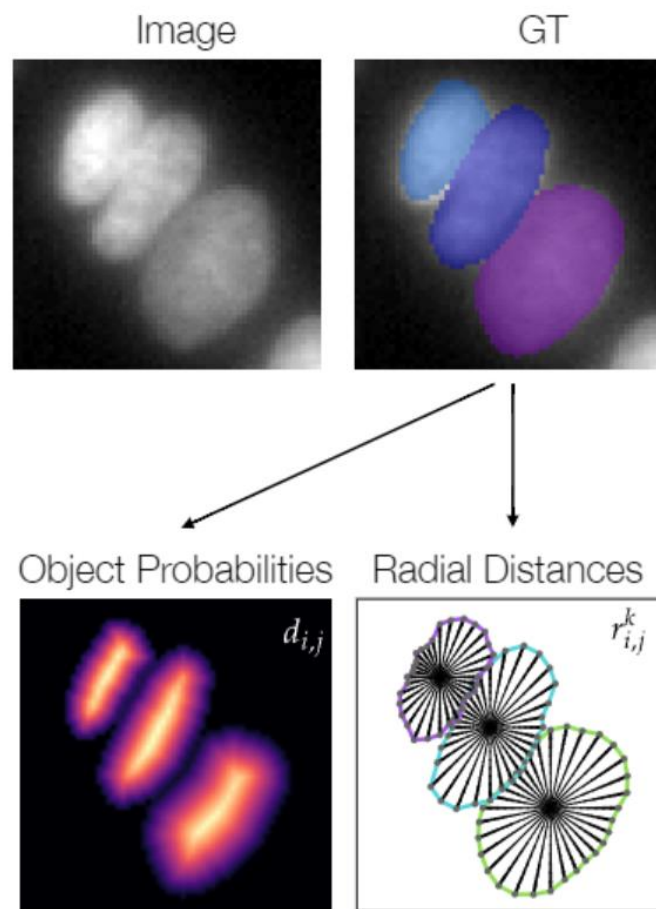
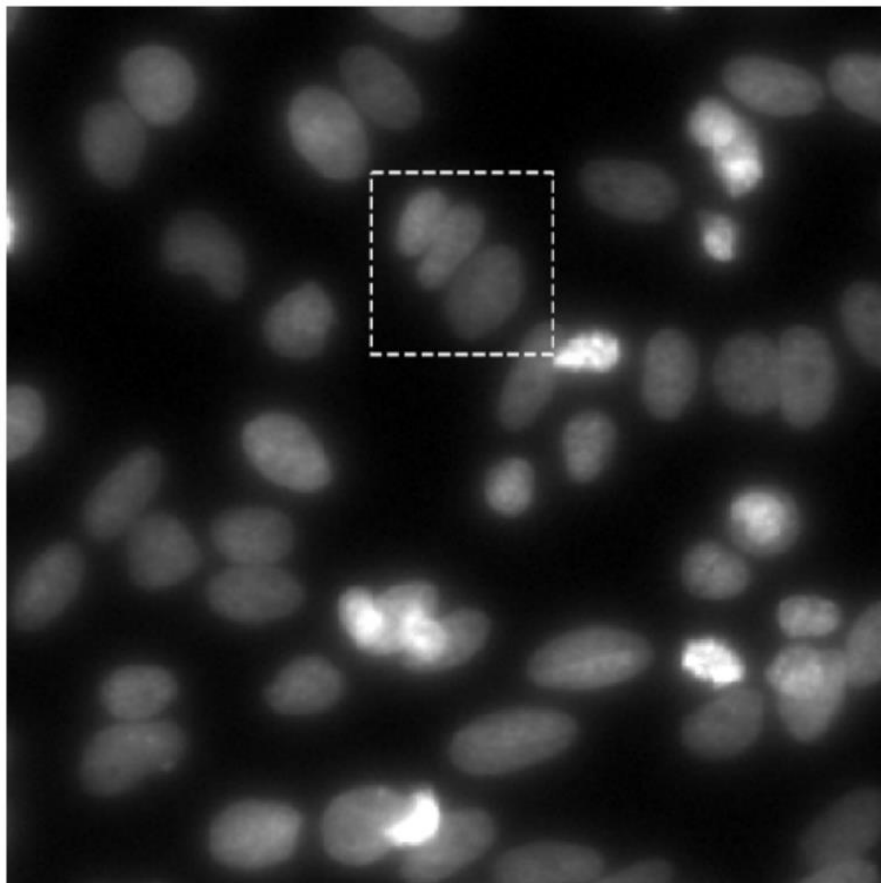
- It is a supervised learning model
- To use stardist we need image and mask pair. (Input and output values)
- Stardist is a model and we will use image J to create such masks for our images.
- post-processing step involved



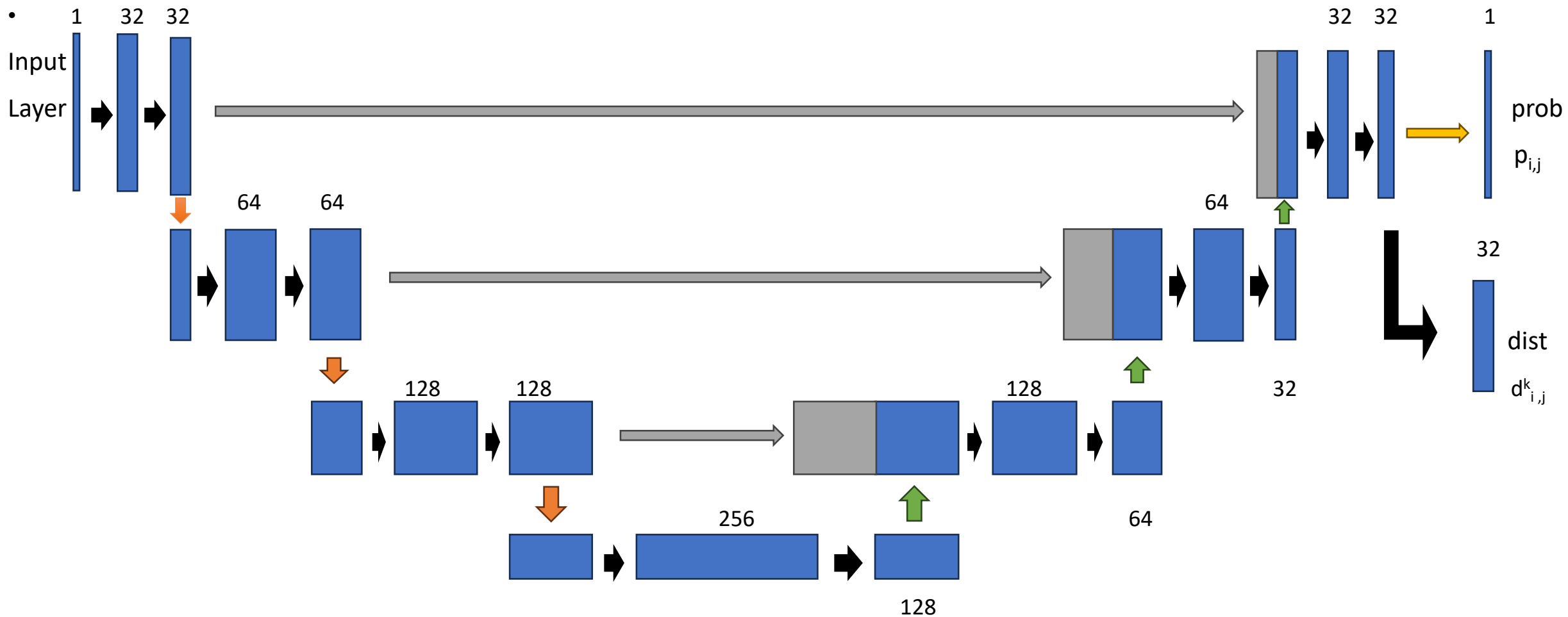
Stardist: Understanding the Working Process

Process

- Stardist model predicts a star shaped polygon for each pixel of the image.
- For each pixel i,j , it predicts two things as follows:
 1. Object probability ($d_{i,j}$):- It is the normalized Euclidean distance to the nearest background pixel.
 2. Star-convex polygon distances ($r^k_{i,j}$):- The Euclidean distances in 32 radial directions till a pixel with different identity is encountered.



Architecture



➡ Conv2D , (3,3), "relu"

➡ Conv2D , (3,3), "sigmoid"

↑ Upsampling, (3,3)

↓ MaxPool2D, (2,2)

➔ Concatenation

Training loss function

- The loss function is the combination of binary cross entropy loss and distance loss (mean absolute error) weighted by the true object probabilities:

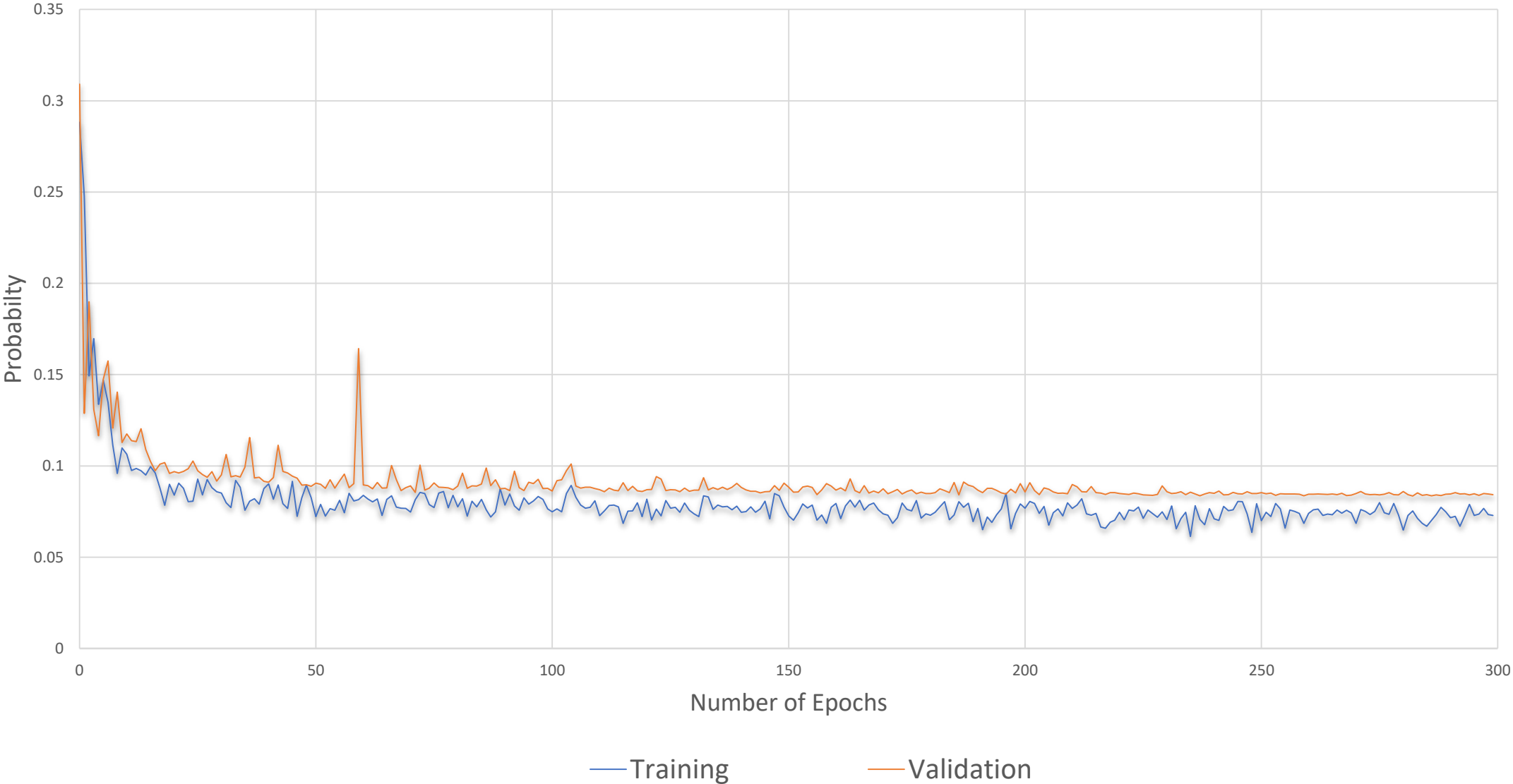
$$L(d, r_k, d', r'_k) = L_{\text{prob}}(d, d') + d' L_{\text{dist}}(d, d', r_k, r'_k)$$

where, (d, r) are predictions and (d'_k, r'_k) are ground truth.

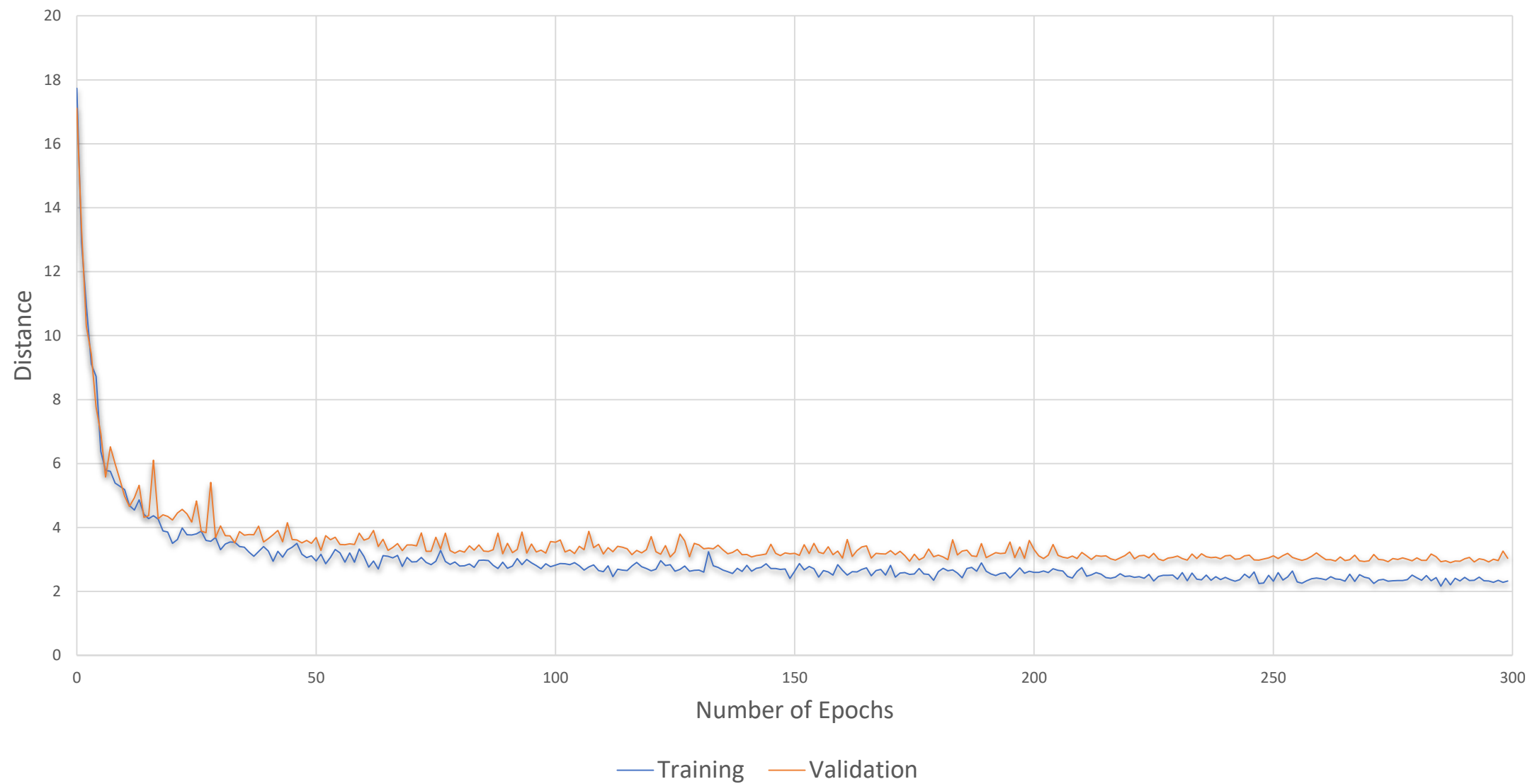
$$L_{\text{prob}}(d, d') = -d' \log(d) - (1-d') \log(1-d');$$

$$L_{\text{dist}}(d, d', r_k, r'_k) = 1/32 \sum_{k=1}^{32} |r_k - r'_k|.$$

Probability loss vs Number of Epochs



Distance loss vs Number of Epochs



Post Process – Interference

IOU

- IOU (Intersection over Union) is an evaluation metric calculated as the ratio of the area of intersection between the predicted and ground truth regions to the area of union between the two regions.

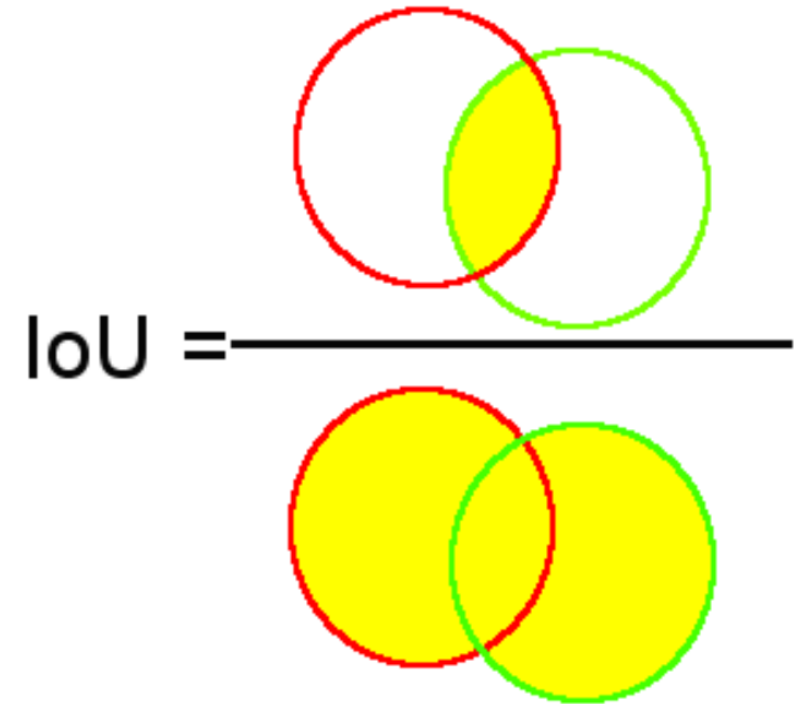


Fig. 4: Intersection-over-Union: ground-truth objects (red) and predicted objects (green)

Probability Threshold

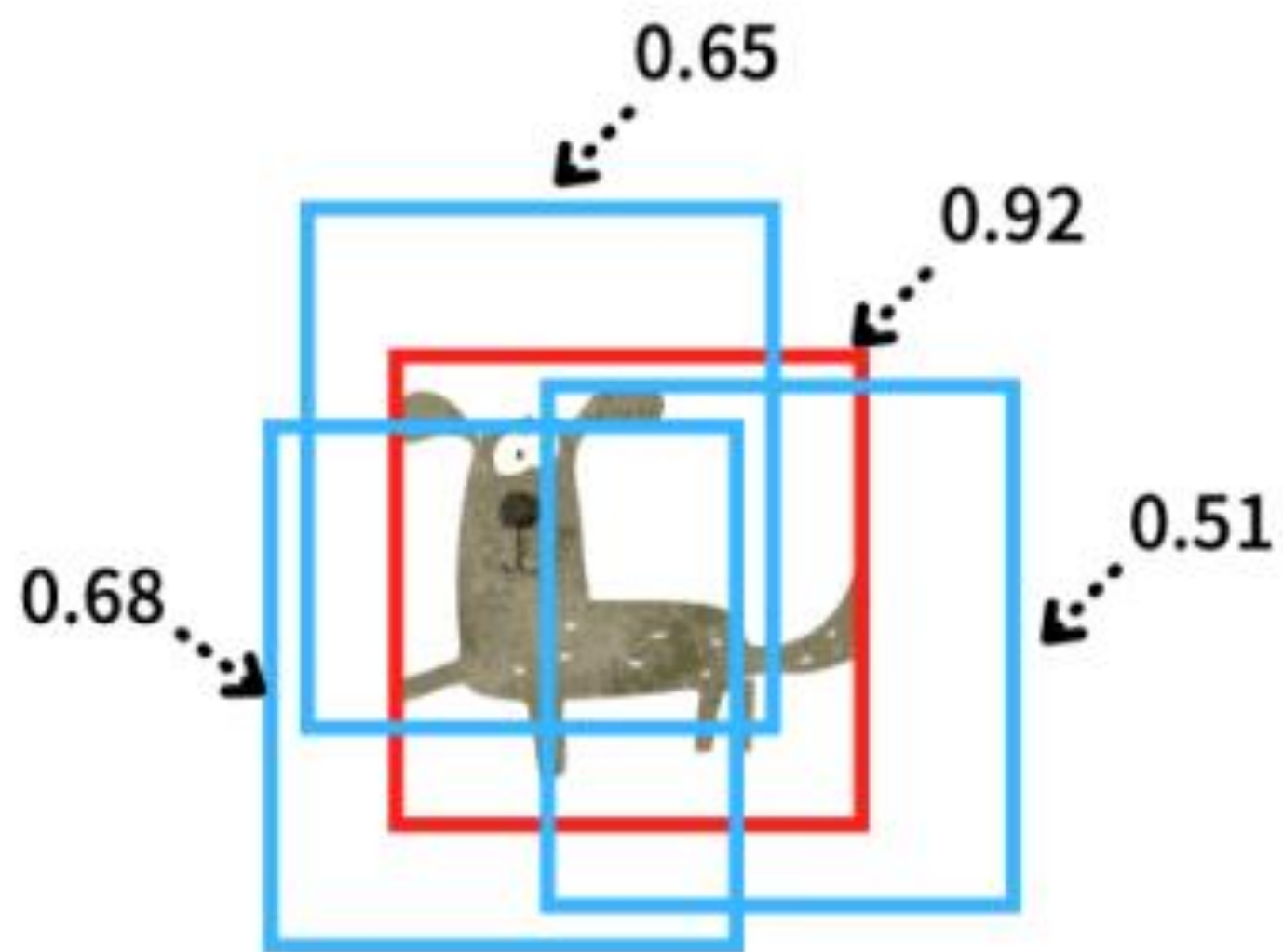
- After the image runs through the Neural network, the algorithm considers the only pixels which has probability ($d_{i,j}$) greater than a certain threshold.

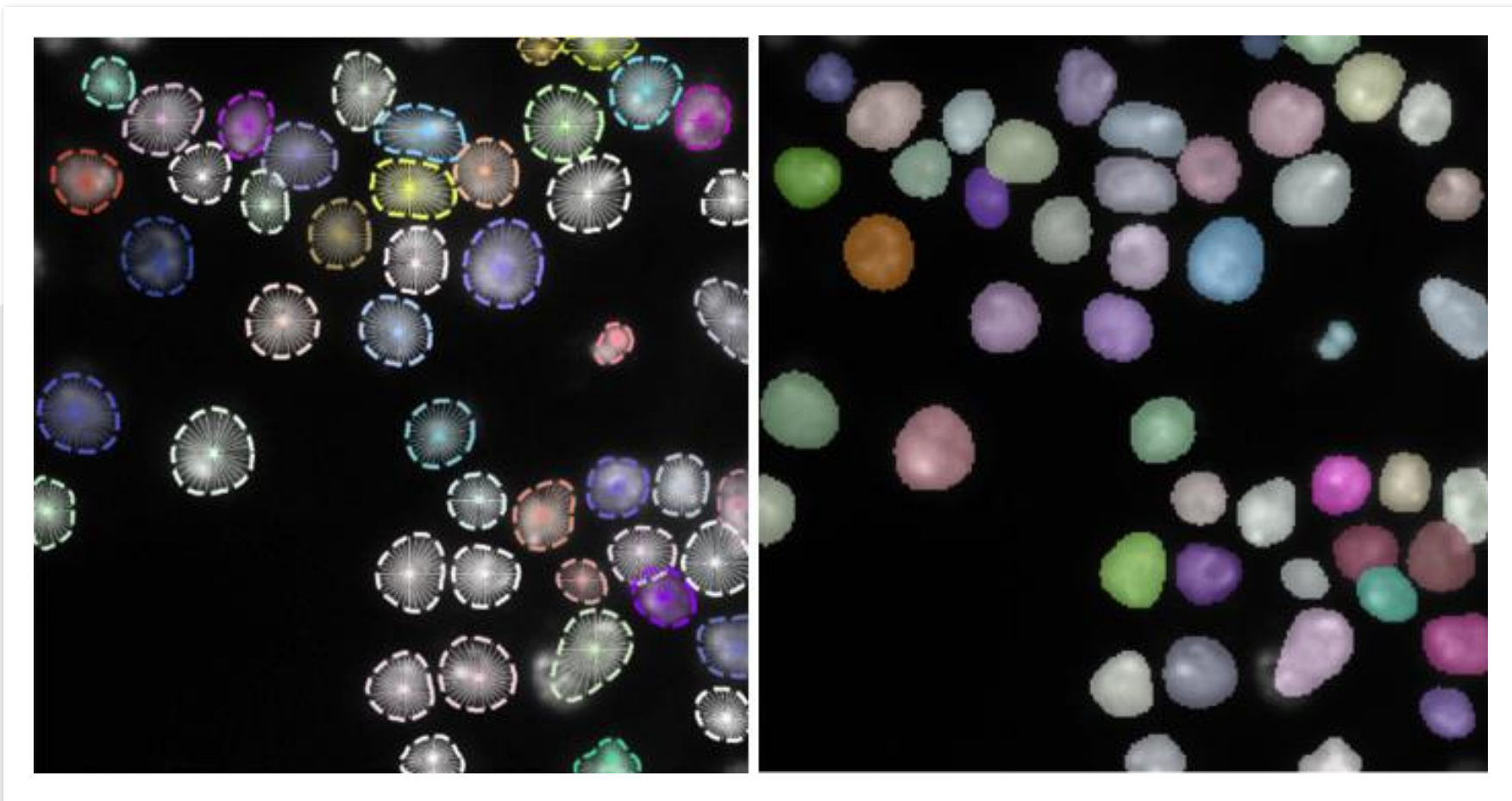




Non-Maximum Suppression

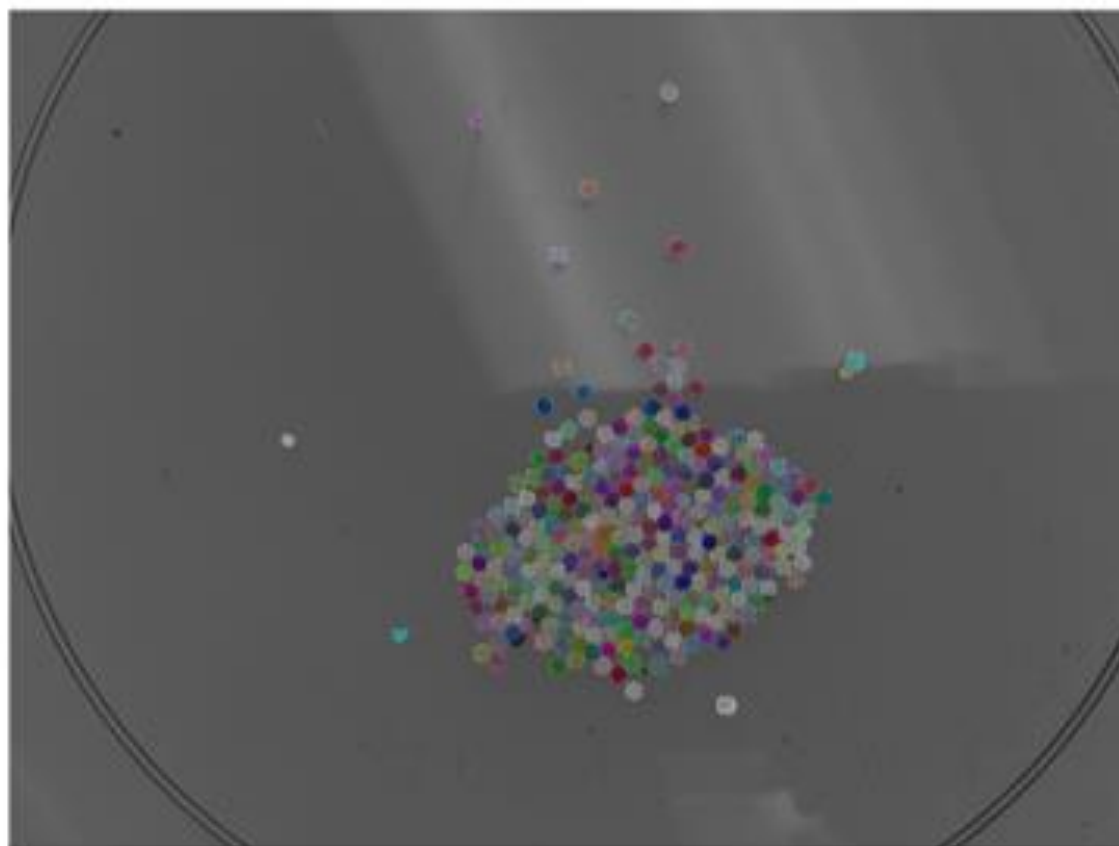
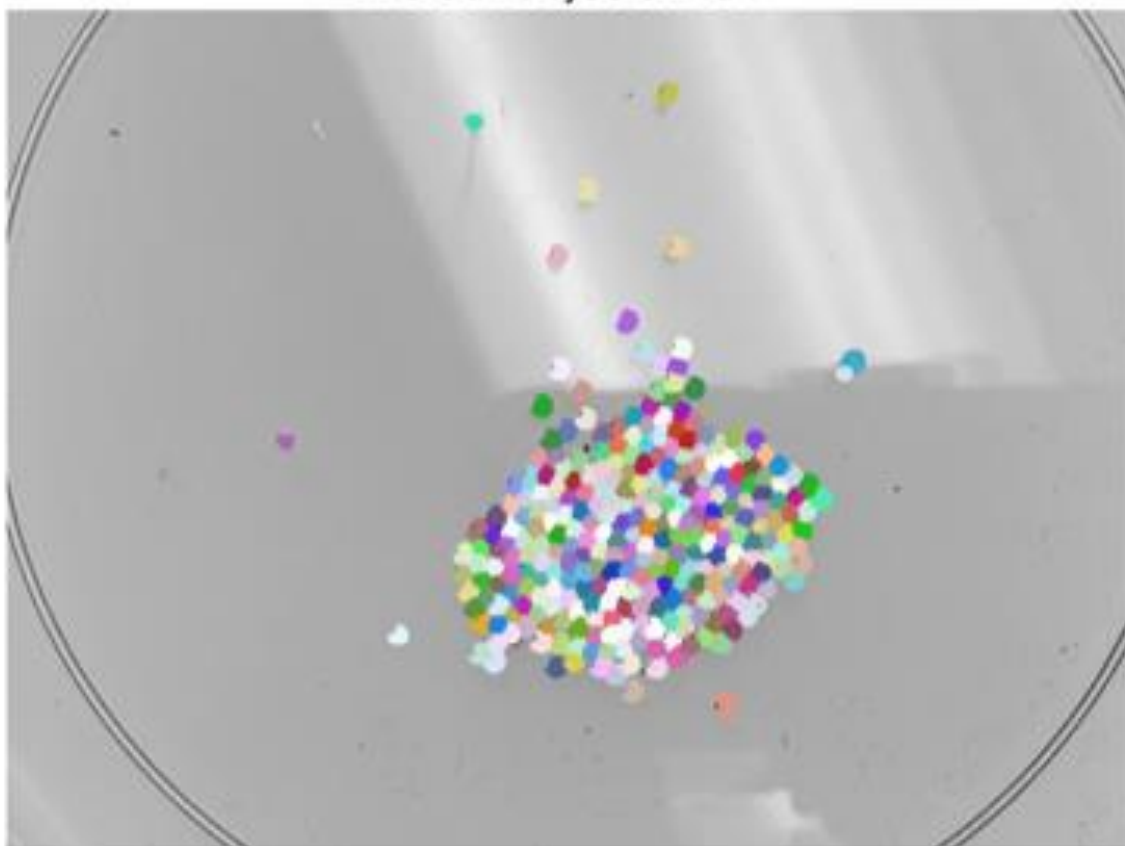
- The NMS algorithm works as follows:
- Calculating Confidence Scores
 - Sorting by Confidence in descending order
 - Suppresses all the polygons with IOU greater than a certain threshold called **NMS** threshold.
 - Iterate



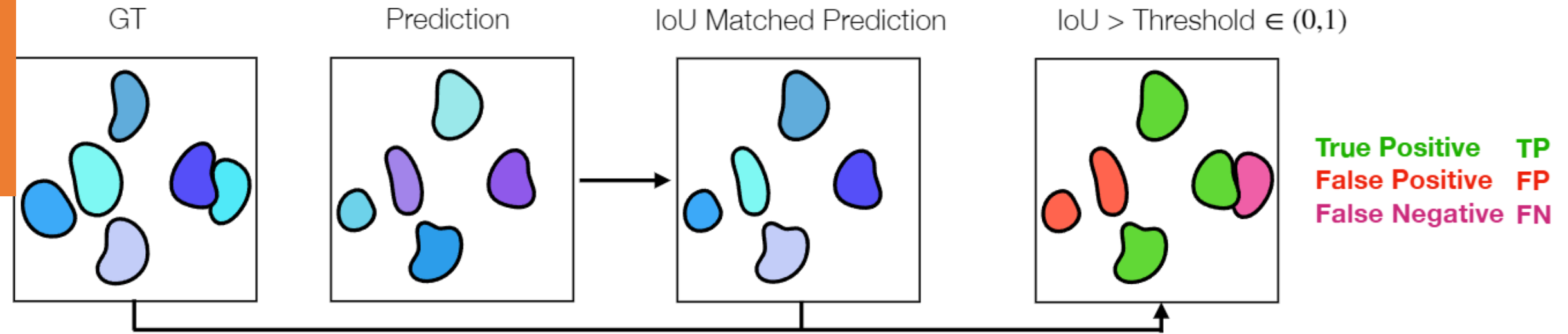


Output

Predicted Objects: 288
Actual Objects: 284



Accuracy Measures



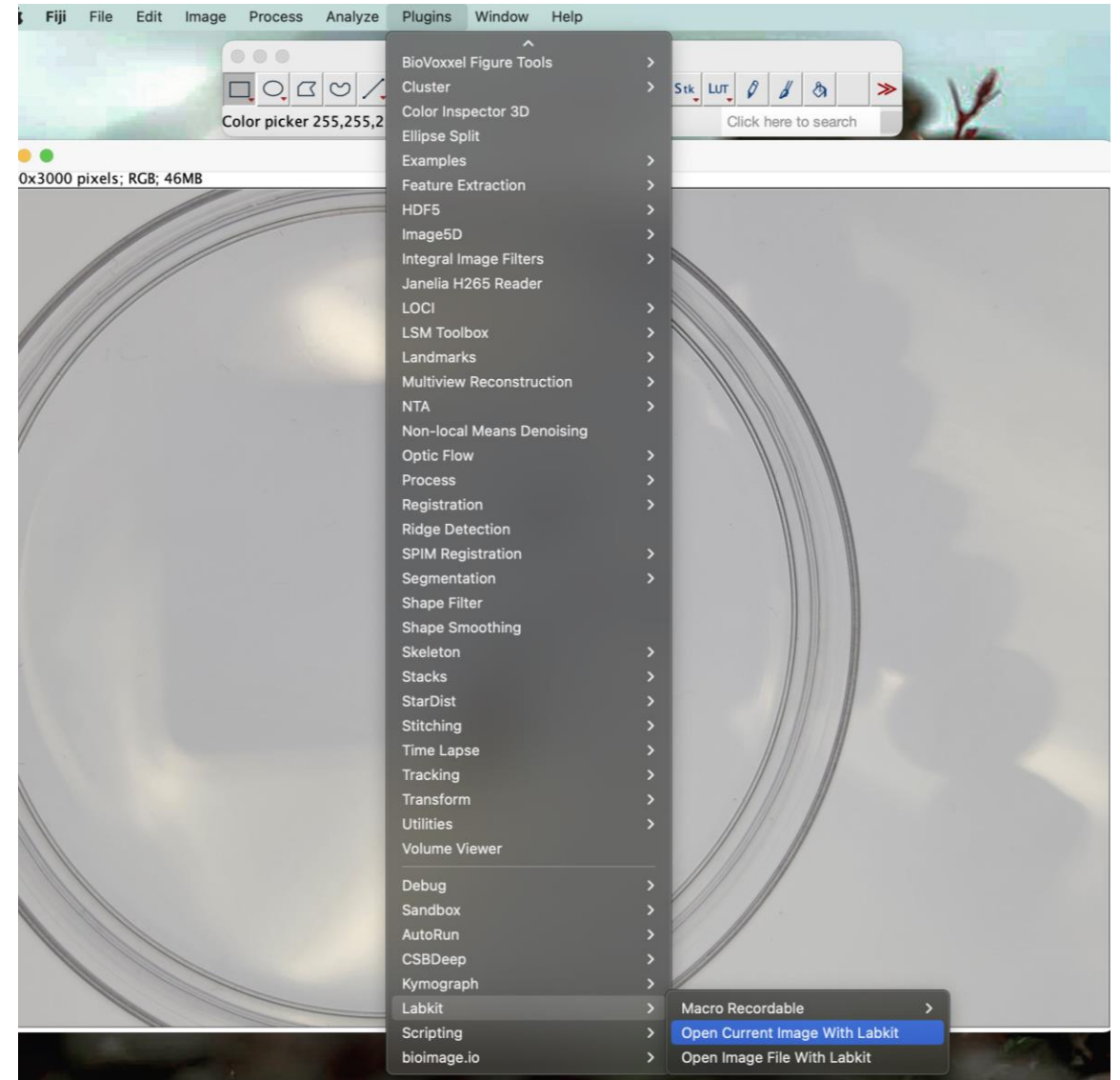
Evaluation

- We evaluate the accuracy based on IOU metric of predicted polygons and ground truth mask.
- Accuracy is calculated as the ratio of true positives to the total number of labels in the image. It measures the overall correctness of the model's segmentation compared to the ground truth.
- To summarize: Average Accuracy = $(TP) / (TP + FP + FN)$

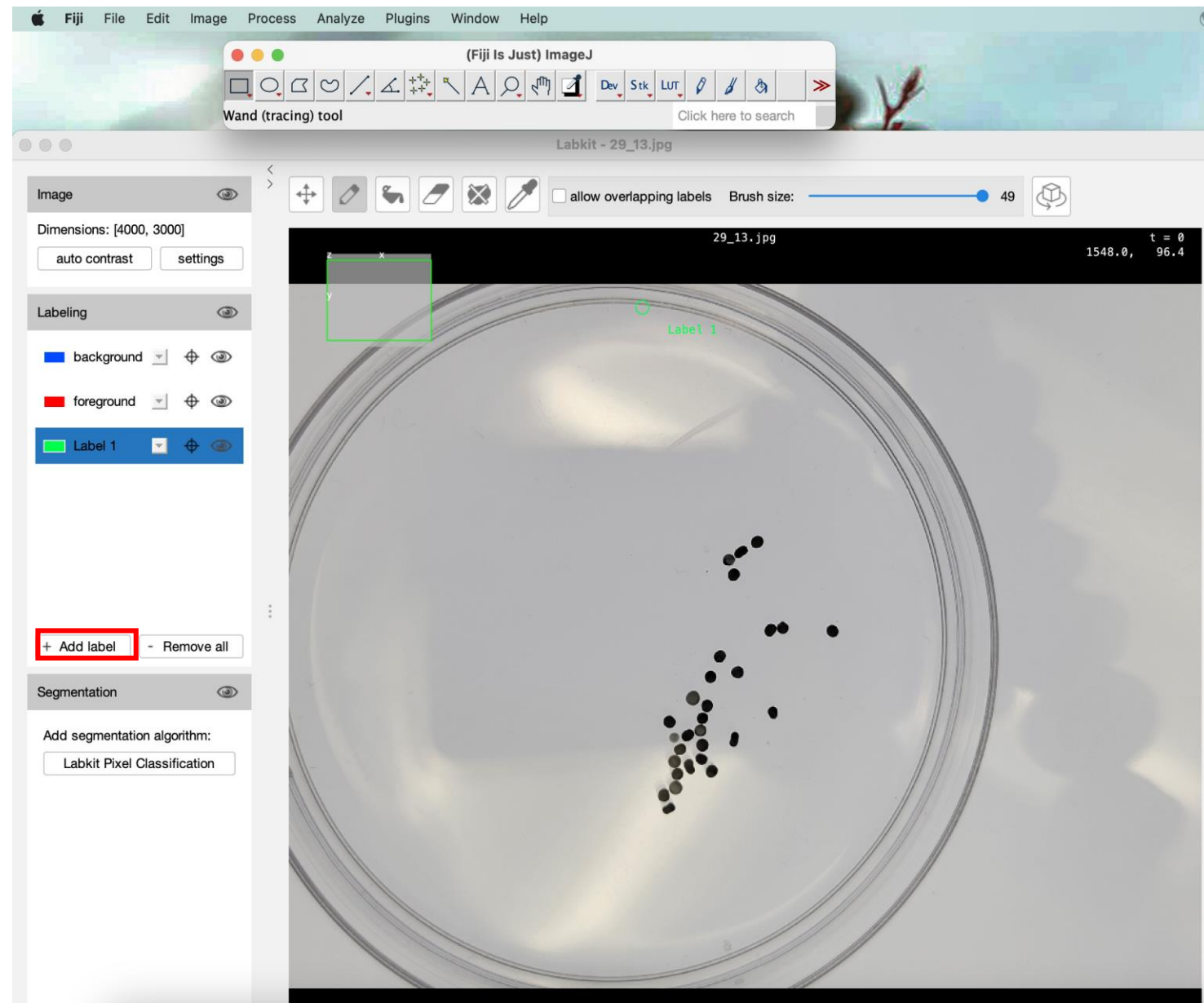
Annotation of image using LabKit

Opening Image via **FIJI** for annotation

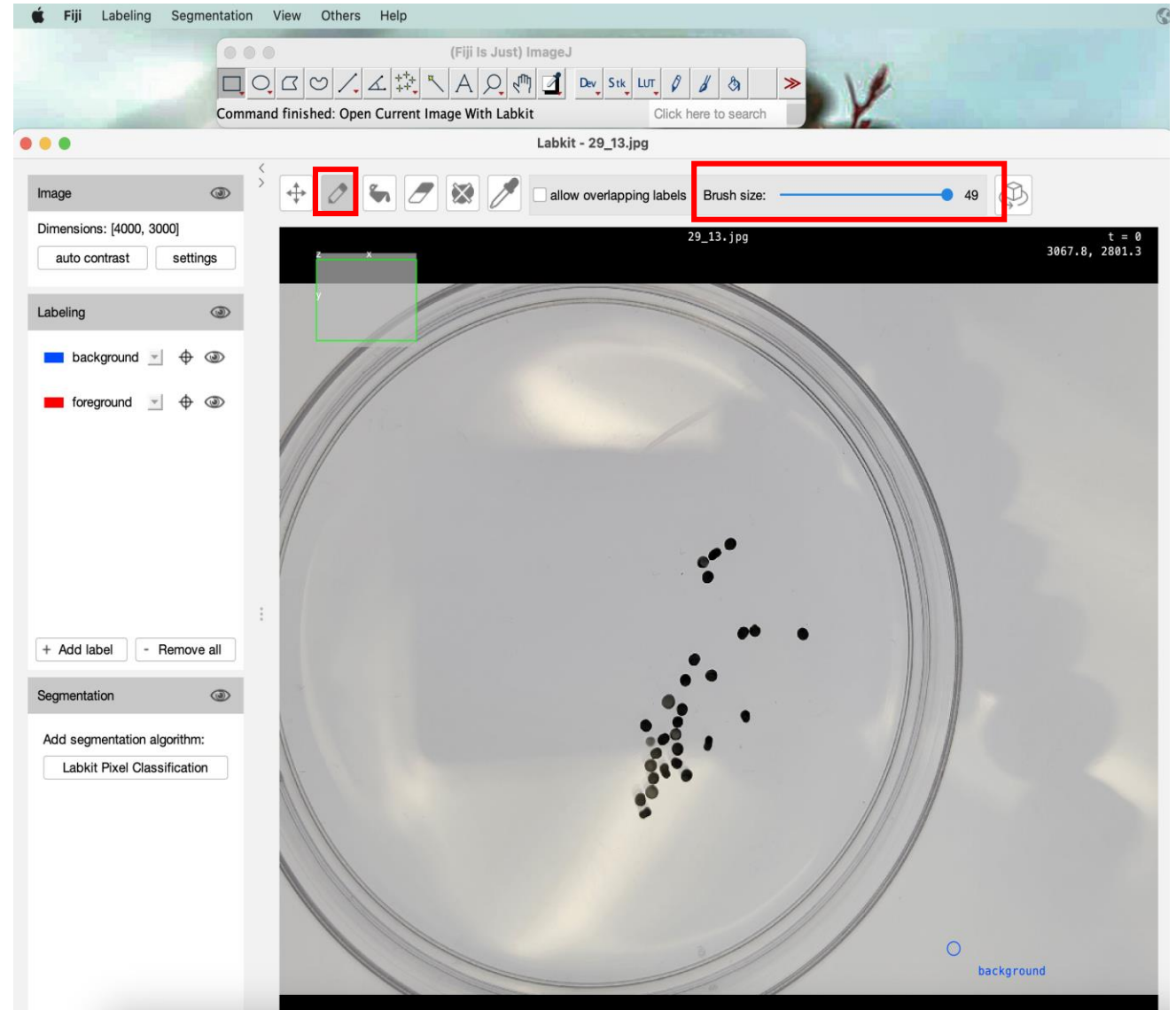
1. Then go on to plugins and select **LabKit**
2. Open current image with **Labkit**
3. Choose the highest resolution to open image (3000 x 4000)



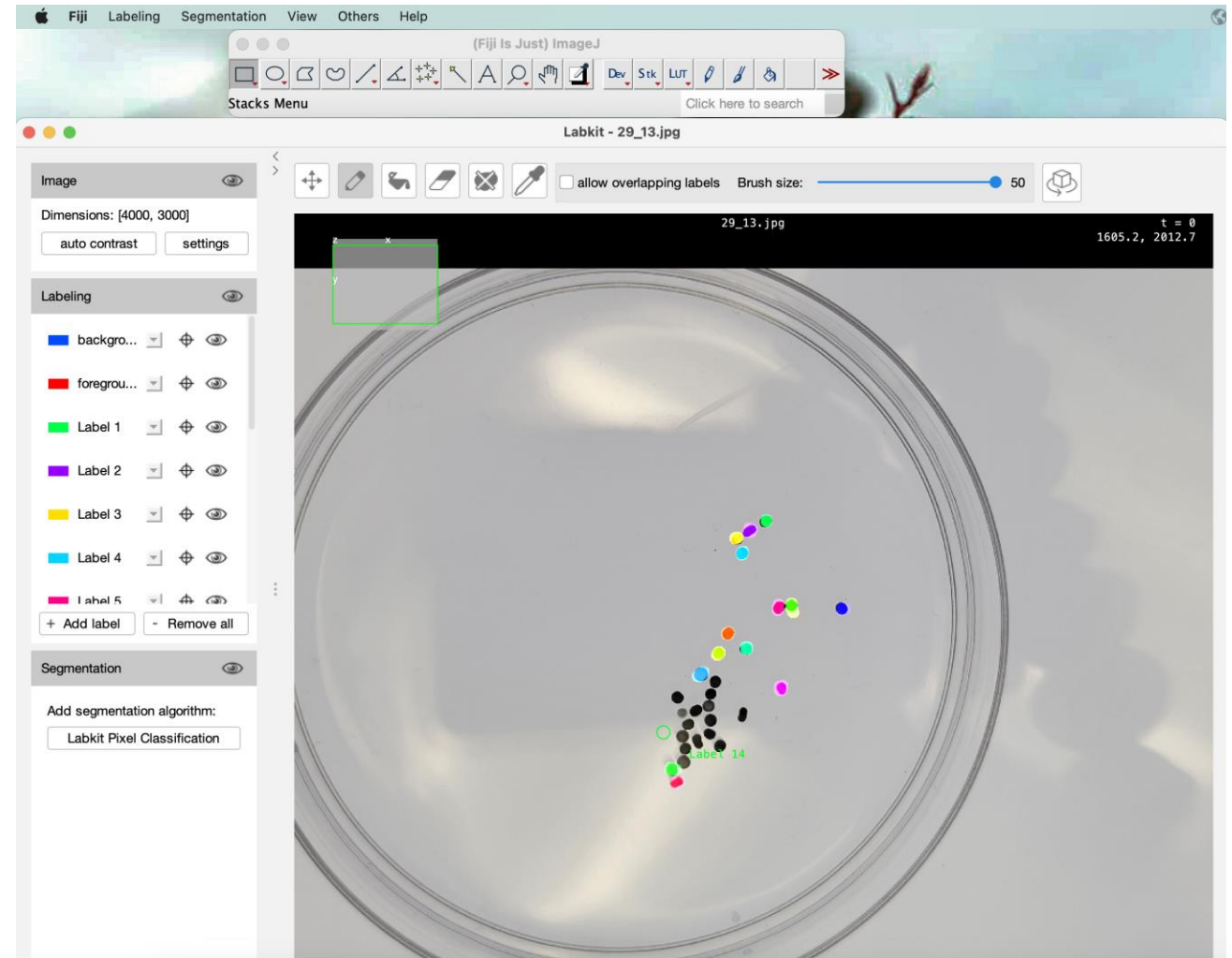
- Once image opens through **LabKit**; Select add label to annotate an egg in the image



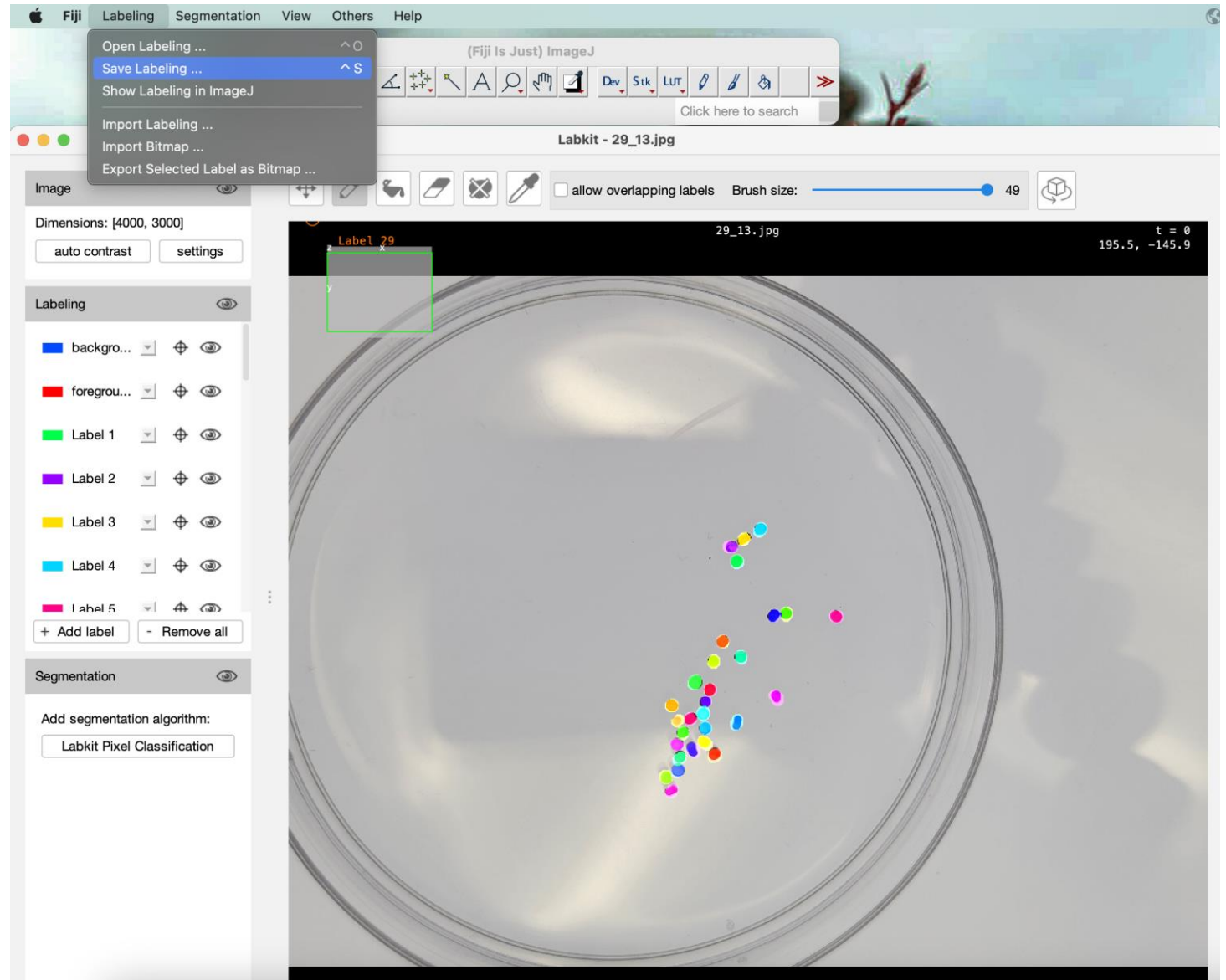
- Select the draw icon and adjust brush size according to egg size in the image to capture egg properly

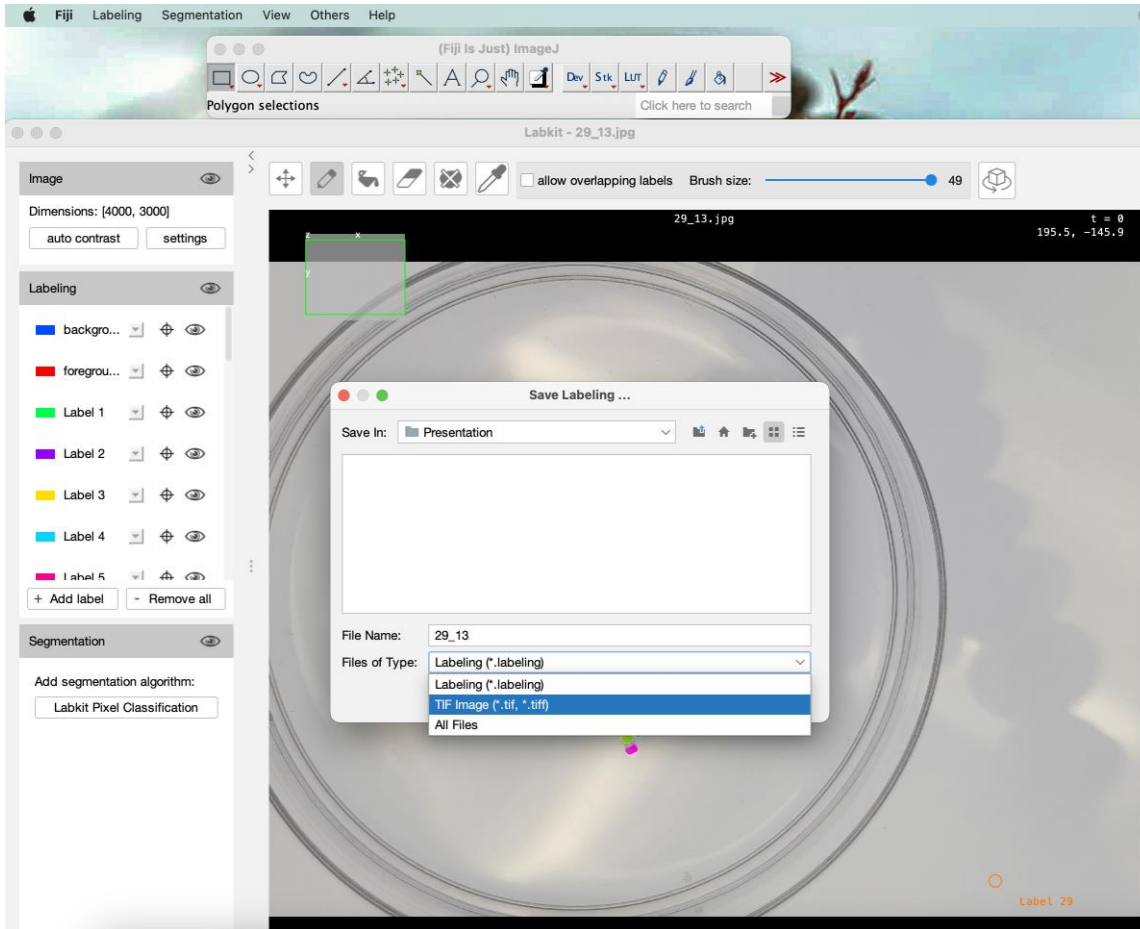


- Start labelling image following previous step
- One color can only be used to label one egg
- At the end, number of labels equals to number of eggs



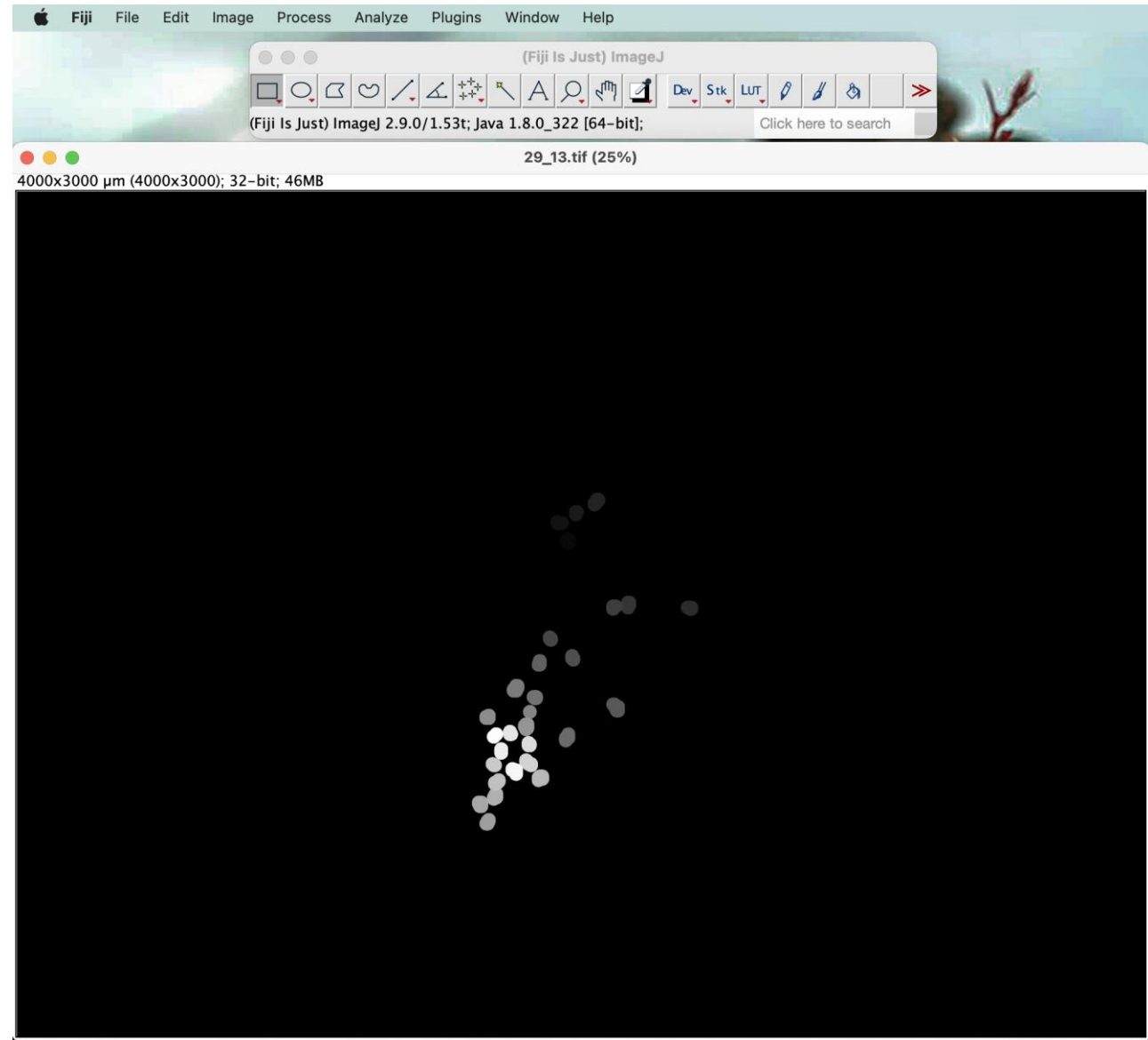
-
- Once labelling is done, select “**Labelling**” from the taskbar and then “**Save Labelling**”





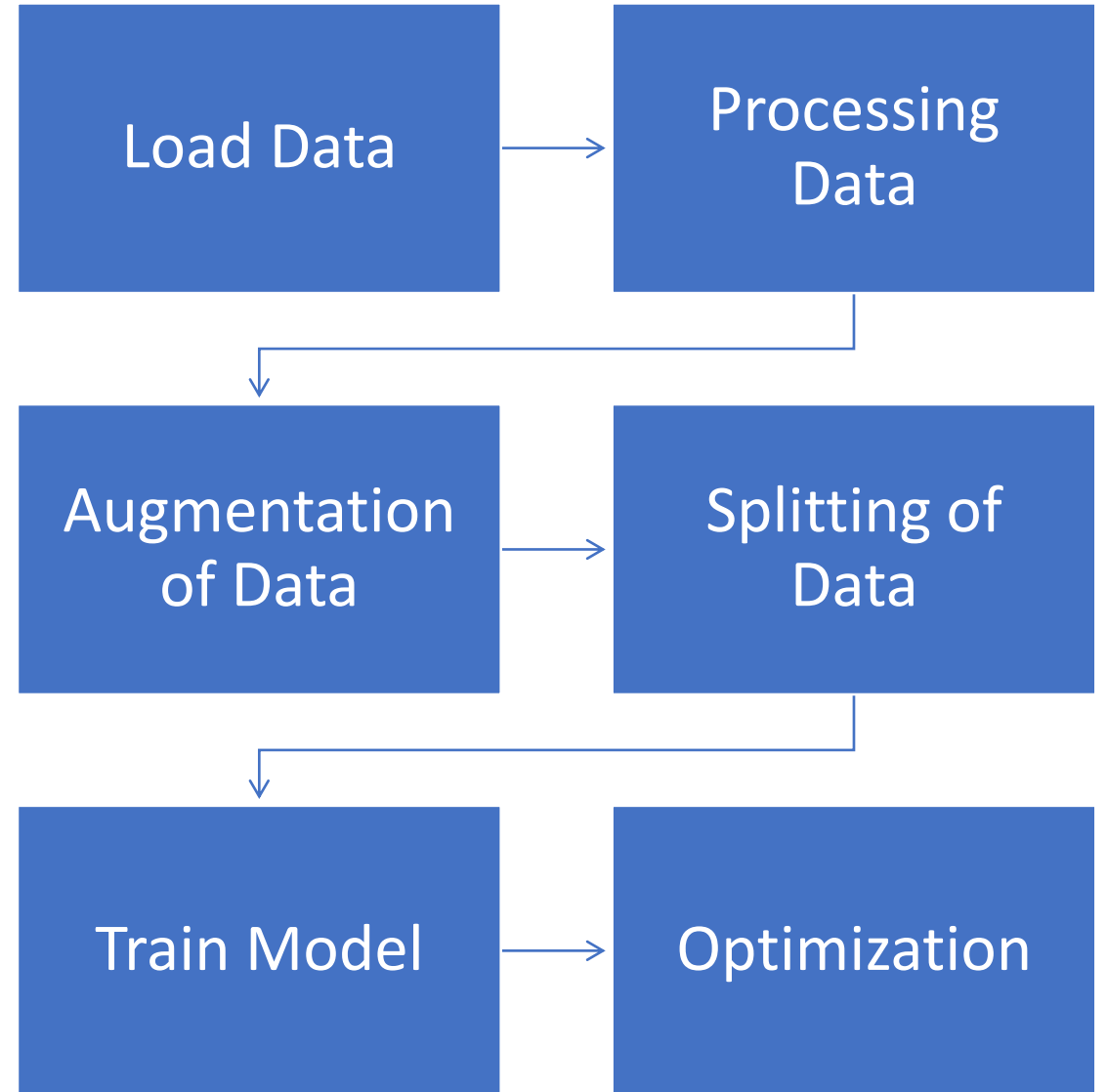
-
- Save image as “**TIF**”
 - The labelled image will automatically be saved in same folder as original image.

-
- Masked image can be previewed by opening the labelled image in **Fiji**
 - Image in the left gives the example of masked image



Overview of Training Process

Training flowchart

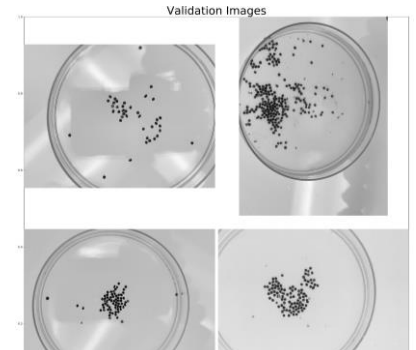
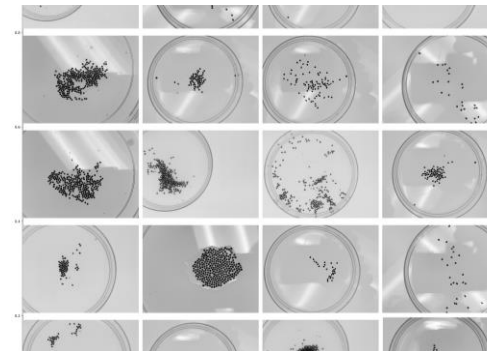
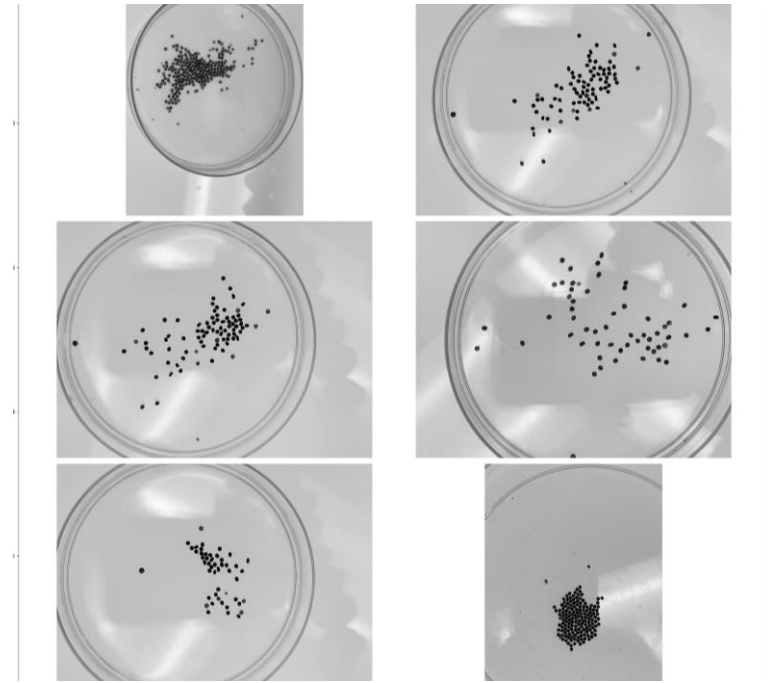


Loading, Processing and Augmenting Data

- Data Loading
 - Loading Data into the program
- Processing
 - Conversion to grayscale
 - Normalization
 - Adjusting image pixels to uniform intensity scale.
- Augmentation
 - Permuting dimensions of the image
 - Flipping, rotation, brightness, contrast
 - Adding noise
 - “Static” or “graininess”
 - Benefits
 - Increases dataset size
 - Boosts model robustness
 - Aides in better generalization to new data

Splitting data

- We need three subsets of images:
 - Training
 - Validation
 - Testing



Training and Optimizing Model

- Training the Model
 - Adding augmenters and defining epoch
- Optimization
 - NMS(Non-Maximum Suppression)
 - Optimizing probability threshold for 0.3, 0.4, and 0.5
 - Algorithm tests each threshold and selects the best

Epochs

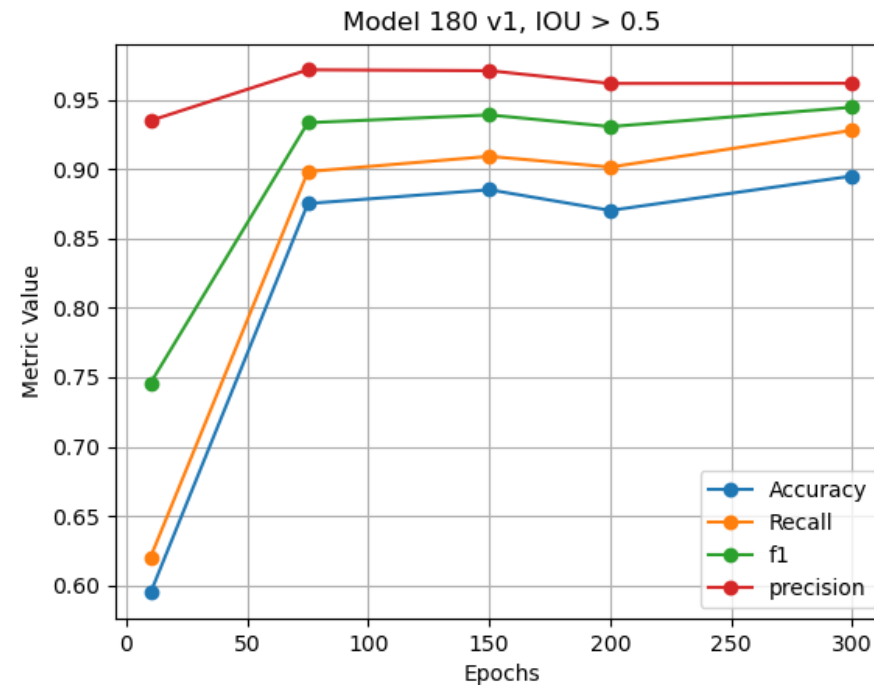


Epoch

- In machine learning, an epoch is a sequence of training steps that makes use of the entire training dataset one time.
- We were interested in observing how the performance of our neural network varied based on the number of epochs allowed for training.
- For each choice of dataset size, we trained models on that size of dataset with the following numbers of epochs: 10, 75, 150, 200, 300
- Generally, too small a number of epochs will not give enough time to train the model. After a while, the model will plateau in terms of performance with respect to training epochs.

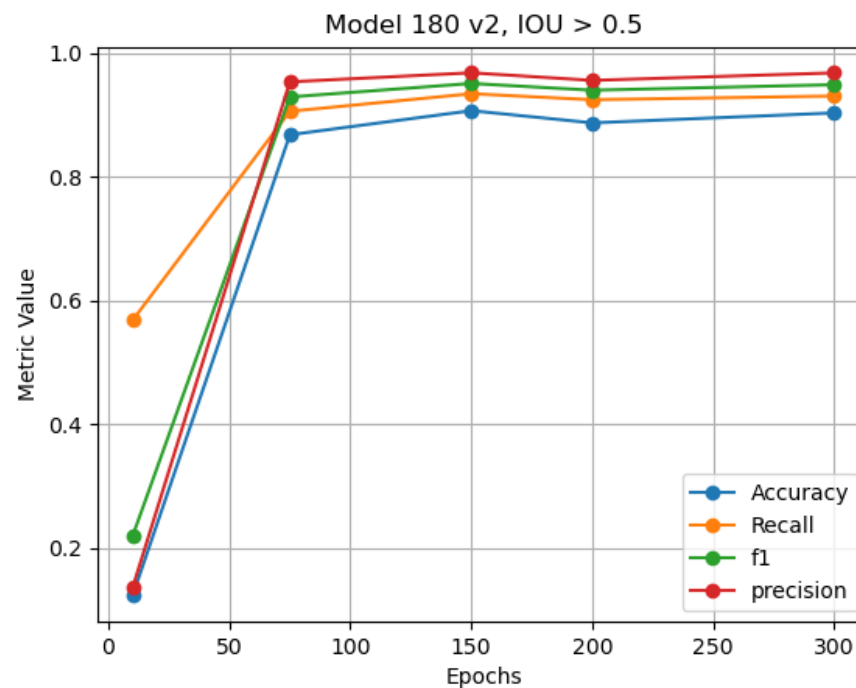
Training Set of 180 Images, v1

- $\text{Accuracy} = \text{TP} / (\text{TP} + \text{FP} + \text{FN})$
 - $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$
 - $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$
- Observe that precision was high after only 10 epochs, though recall was low. So, there were many false negatives, but not many false positives.



Training Set of 180 Images, v2

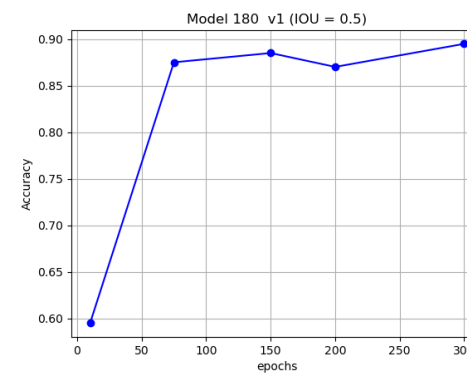
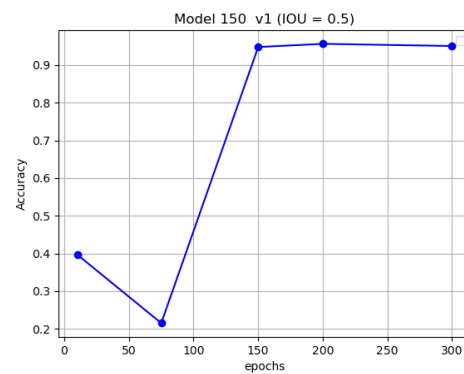
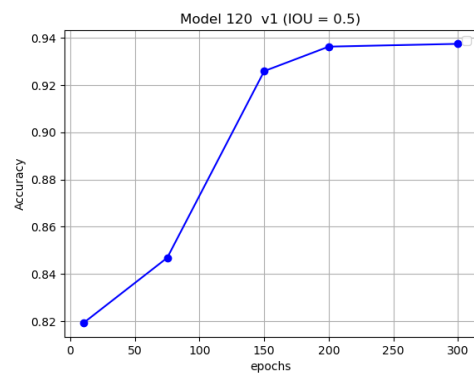
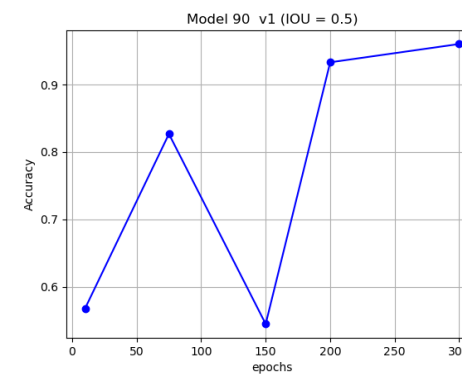
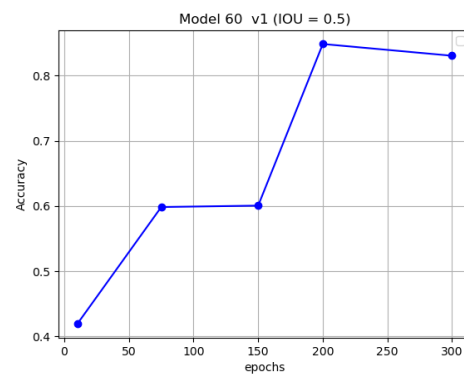
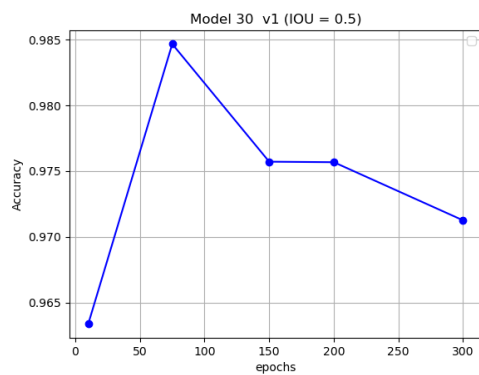
- $\text{Accuracy} = \text{TP} / (\text{TP} + \text{FP} + \text{FN})$
 - $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$
 - $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$
- In this version, precision was very low after just 10 epochs, so there must have been many false positives.



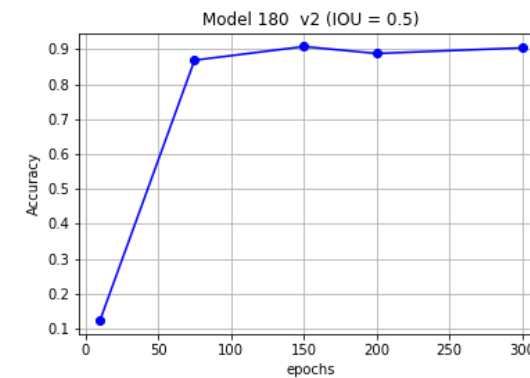
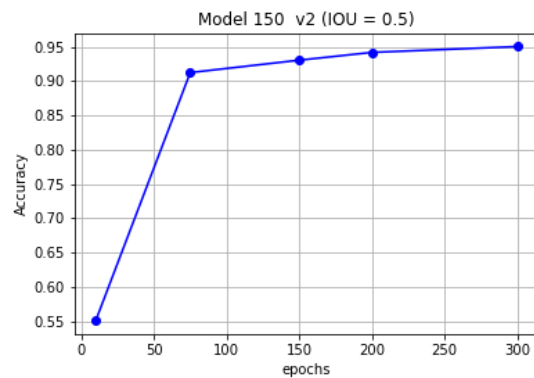
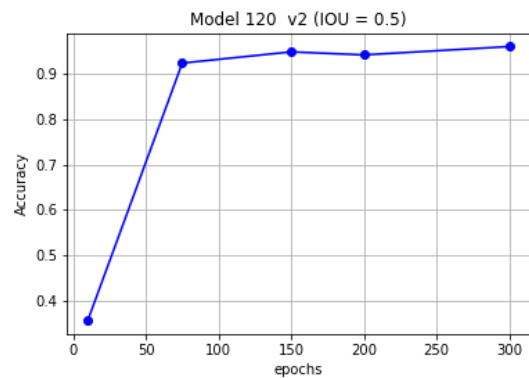
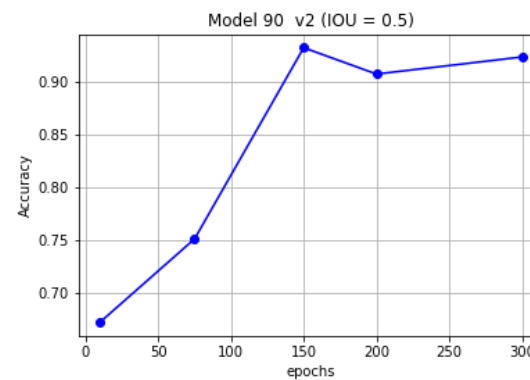
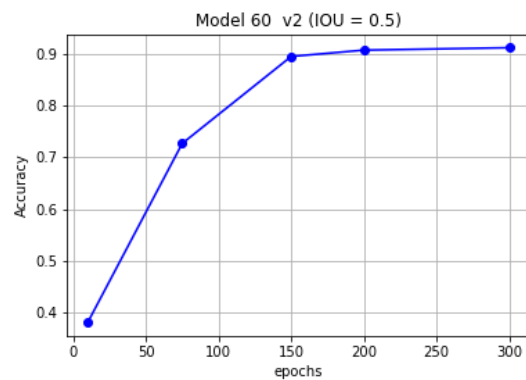
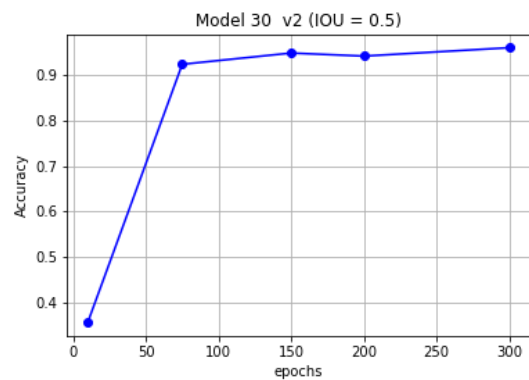
General Trends

- In the following slides, we have graphs of average model accuracy with respect to training epochs for every model that we trained.
- Typically, 200 epochs was sufficient to reach the maximum accuracy. In some cases, we observed decreased performance on more epochs of training.

General Trends, v1



General Trends, v2



Dataset sizes

Dataset Sizes

- 180 image-mask pairs to work with
- Train multiple models with varying data set sizes at a range of number of epochs
- Ensure that we are not overfitting our data either by providing excess data or running with too many epochs

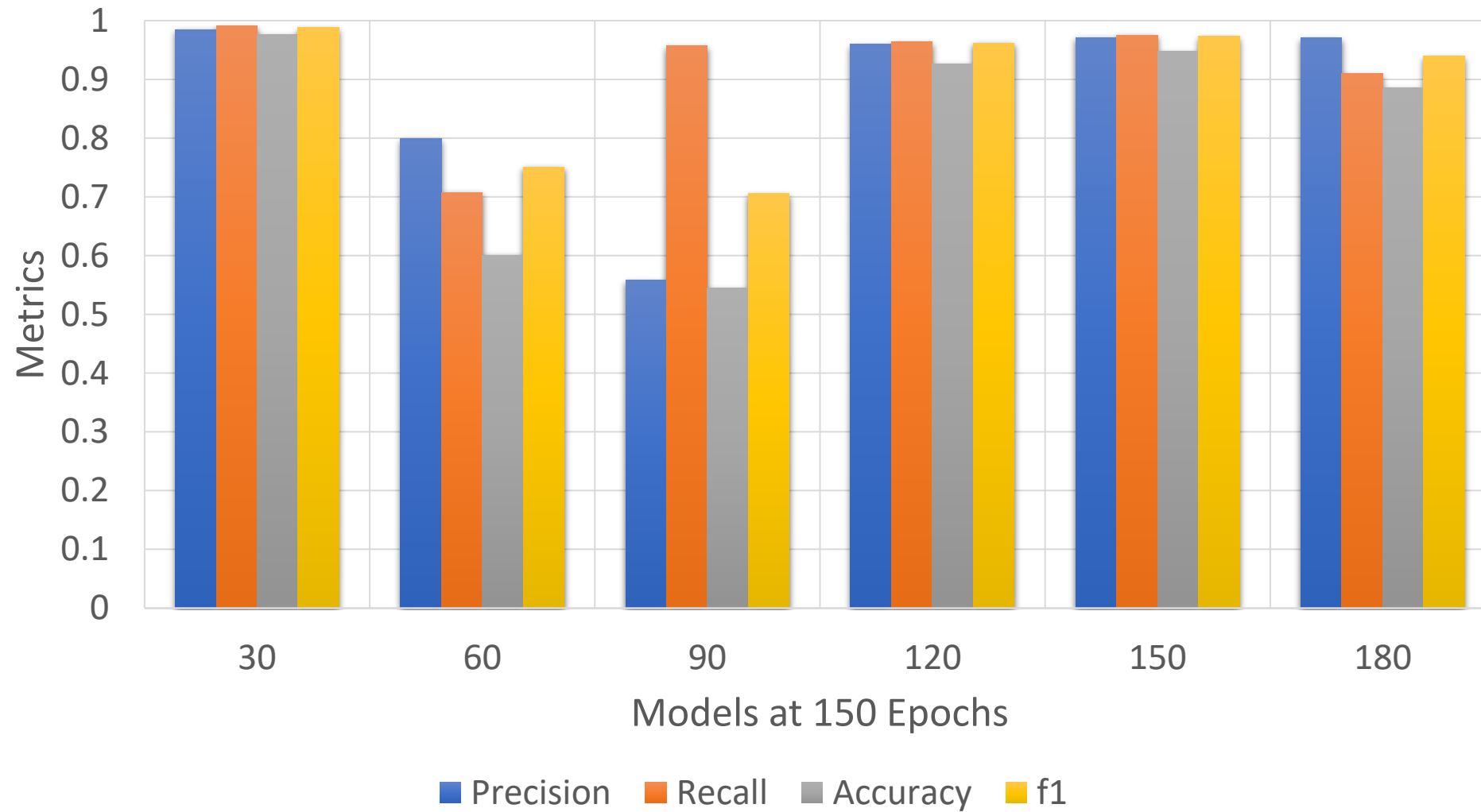
How the Data was Split

80-20 Split of Images		
Images	training	testing
30	24	6
60	48	12
90	72	18
120	96	24
150	120	30
180	144	36

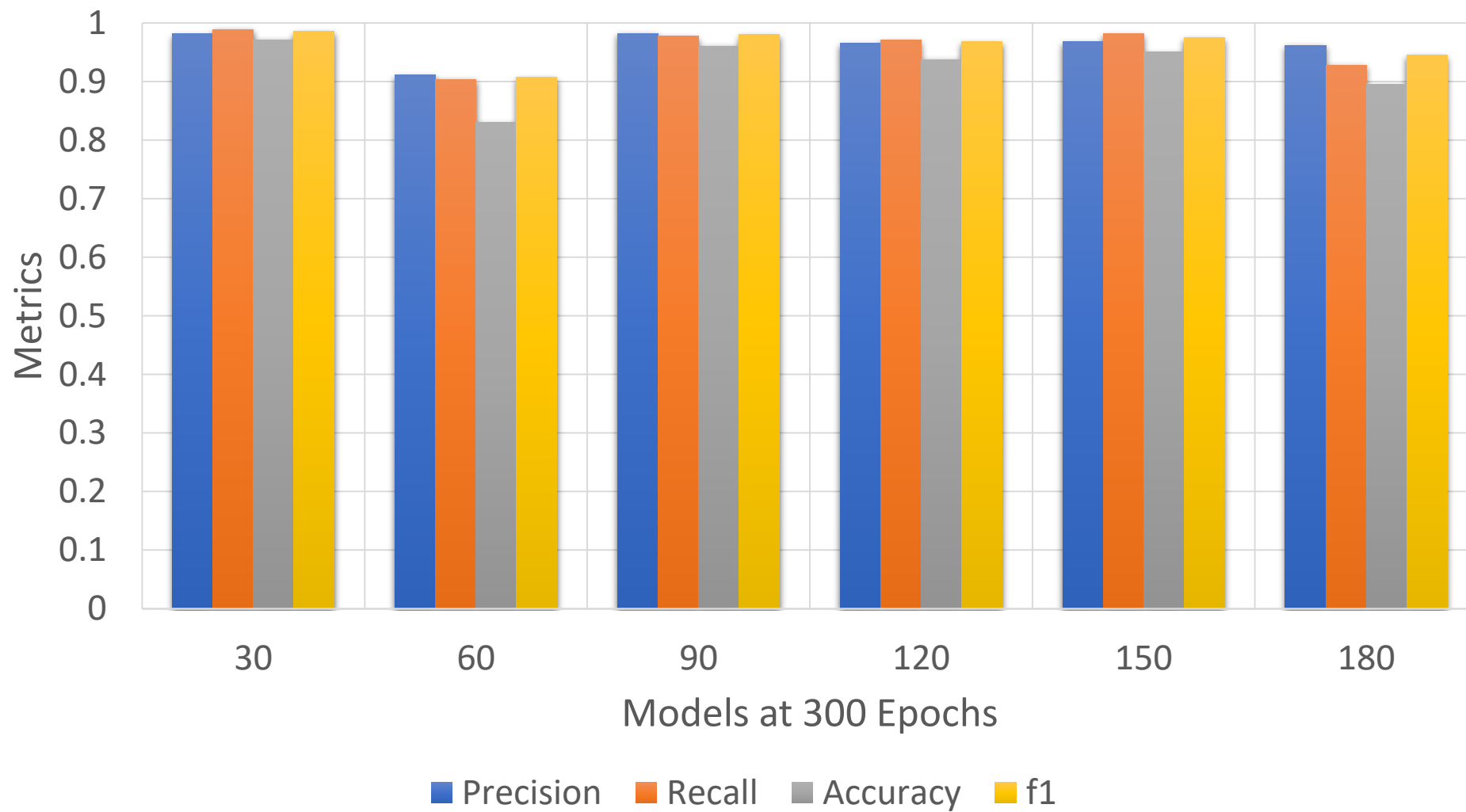
Epochs per Model
10
75
150
200
300

- This split was done twice using two different versions of each dataset (denoted v1 and v2)

Metrics vs Models at 150 Epochs



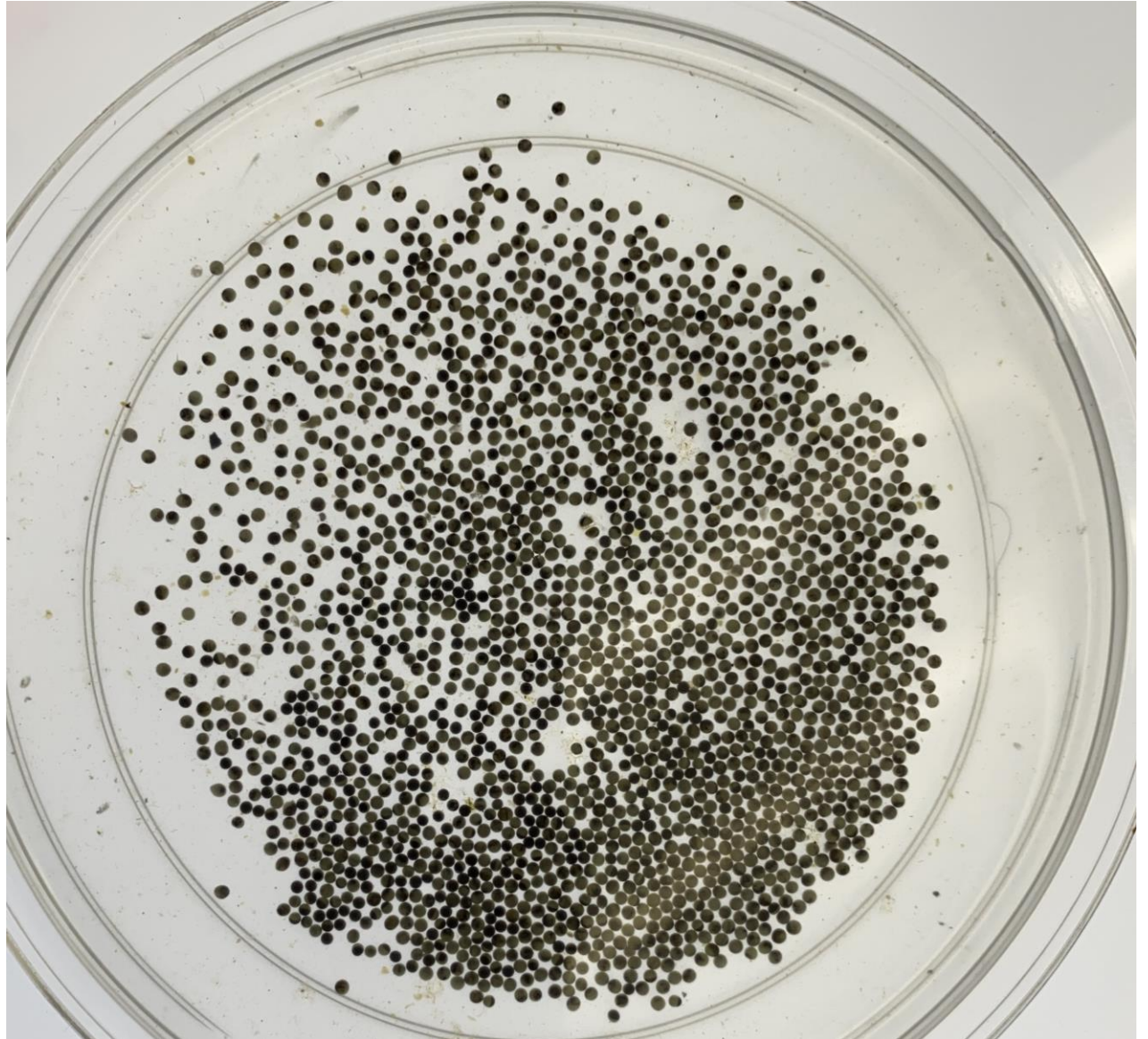
Metrics vs Models at 300 Epochs



Results

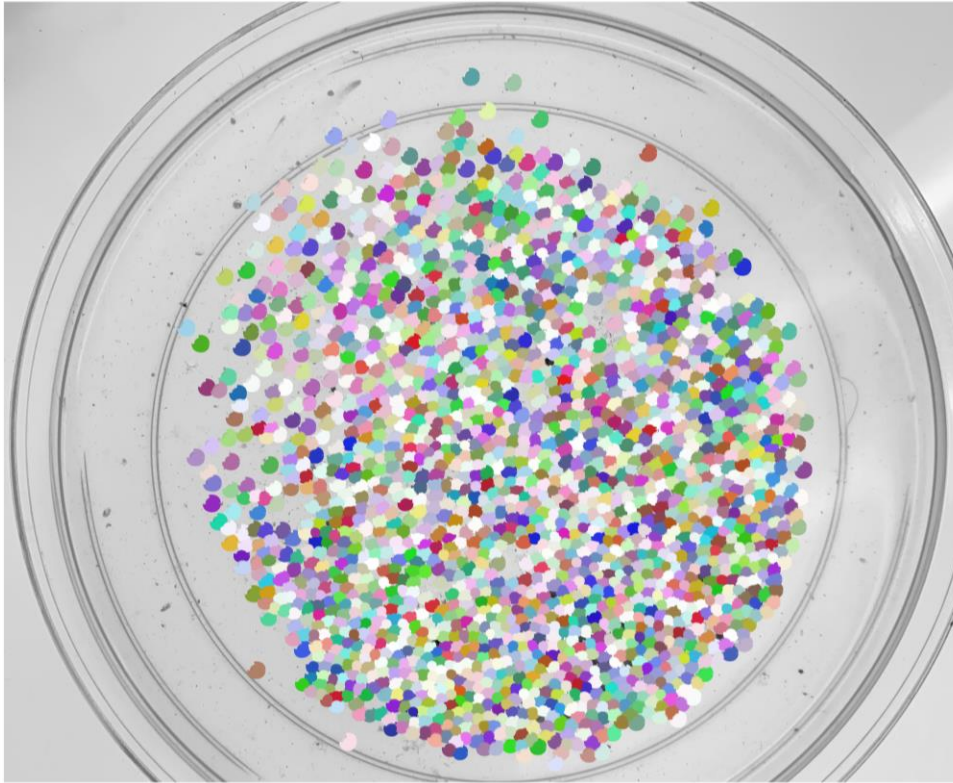


Example Image of 2006 Eggs

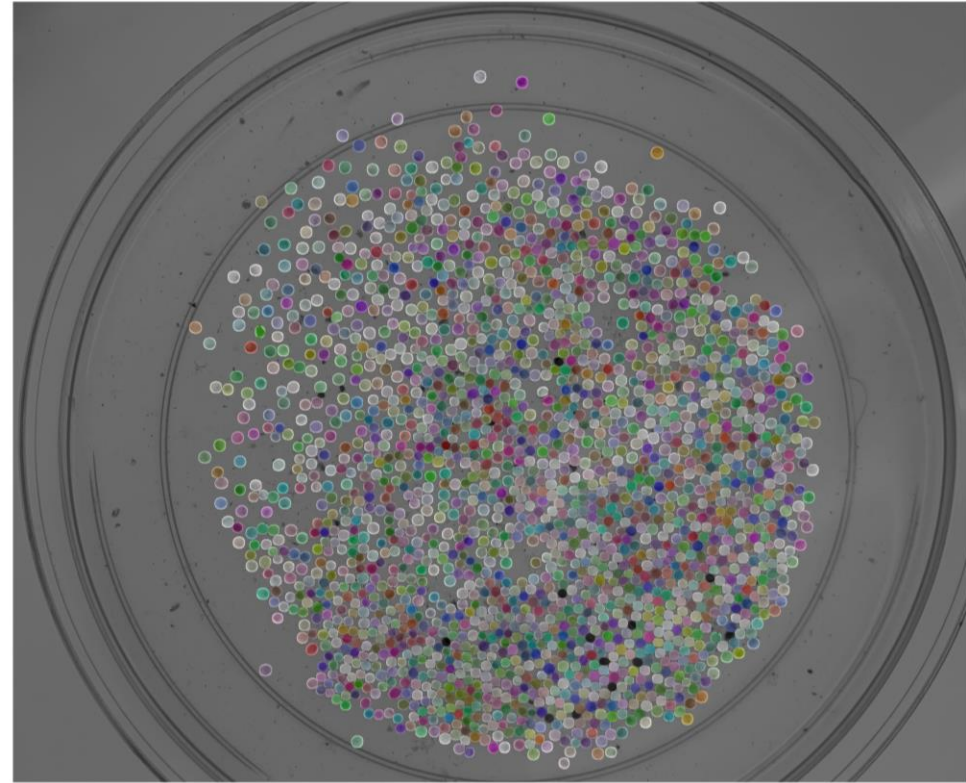


Example Image After Running StarDist

Predicted Objects: 1992
Actual Objects: 2006



Accuracy: 99.30%



Recall This Graph, but now with CI's

StarDist V2 with 180 Images &
300 Epochs

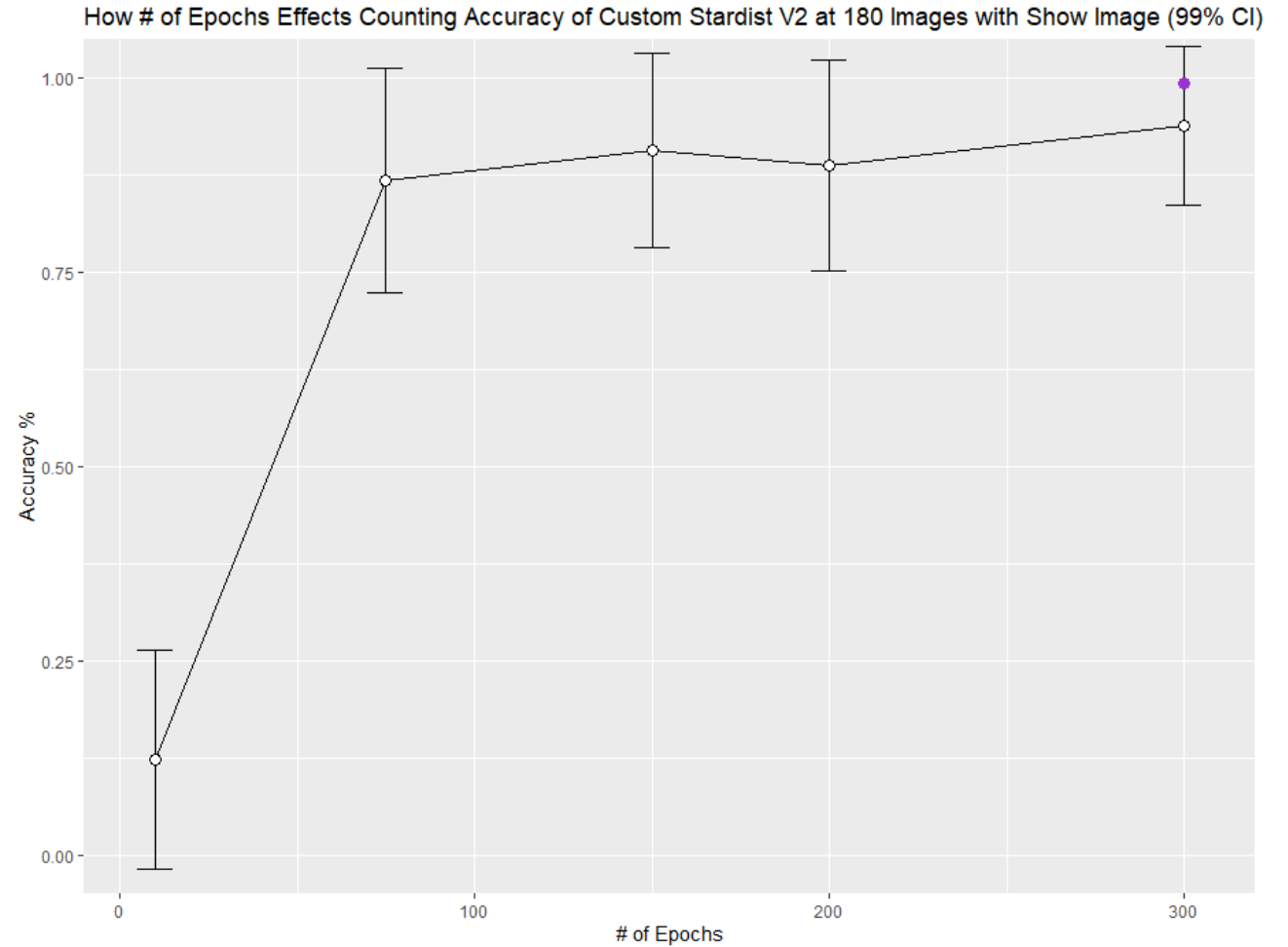
Mean Accuracy: 93.95%

Standard Error: 3.97%

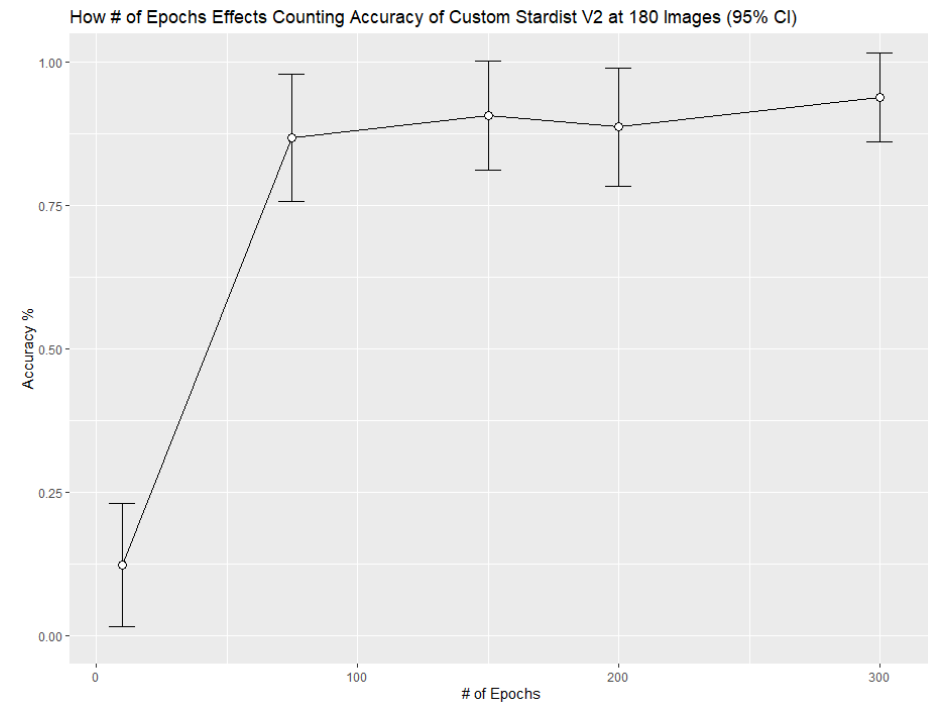
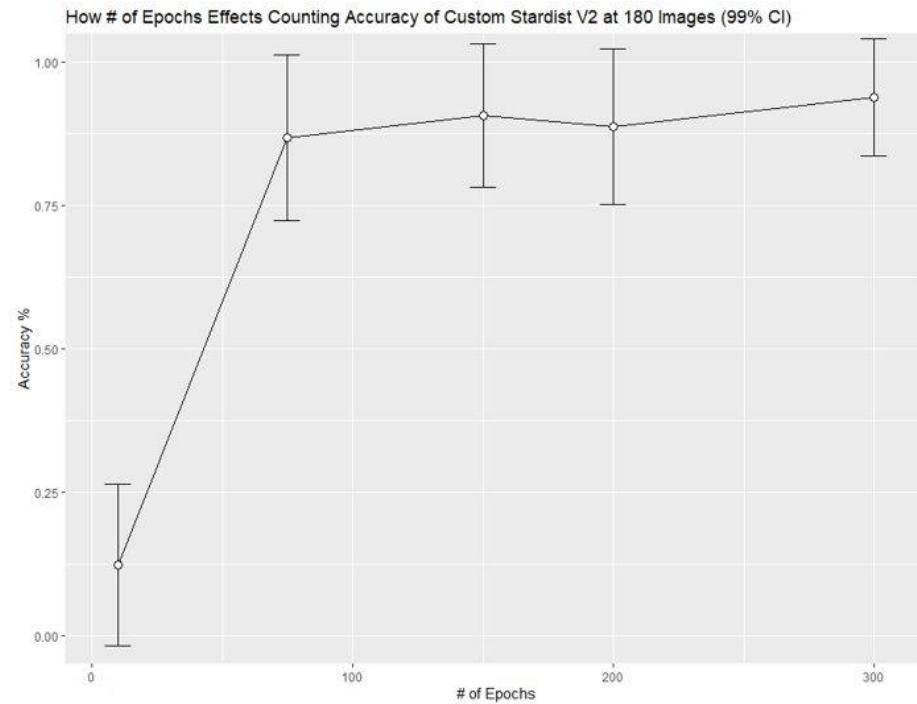
99CI: 83.13-100%

95CI: 85.89-100%

Predicted Image Accuracy:
99.30%



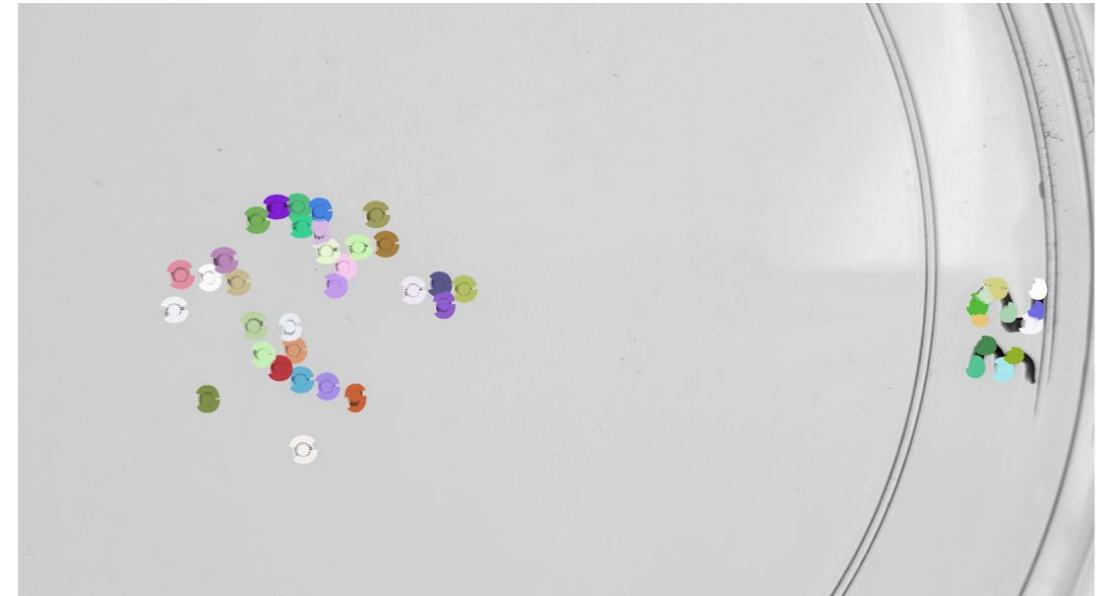
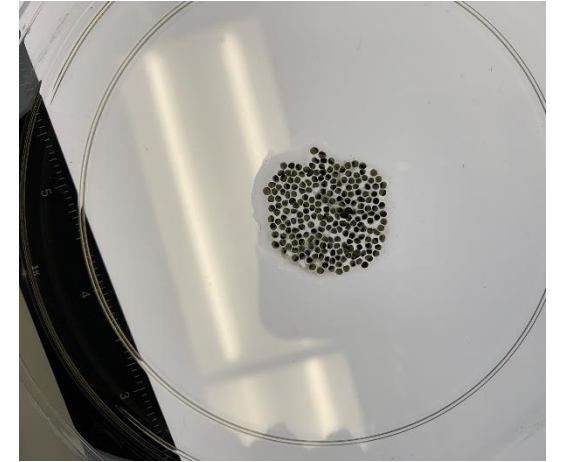
A Note of Redundant Epochs



Future work

Refined dataset

- Early stopping of epochs
- More images(180 total)
- Better quality of images(some fuzzy or low quality)
- Consistent dimensions (some images were small)
- Adding more noise(only two images had noise; note top images)
- Noise would help prevent an improper prediction like bottom image





- Clustering
- Multiclass predictions
 - Distinguishing stage of eggs
 - Multiple types of eggs
- Video
 - For using StarDist on another custom dataset for a different problem
 - Made by the DeVision Team

Thank You!

