Supervised Machine Learning for Biomolecular Condensate Quantification and Measurement

AM91r: Supervised Reading and Research
Rodney Lafuente Mercado
lafuentemercado@college.harvard.edu

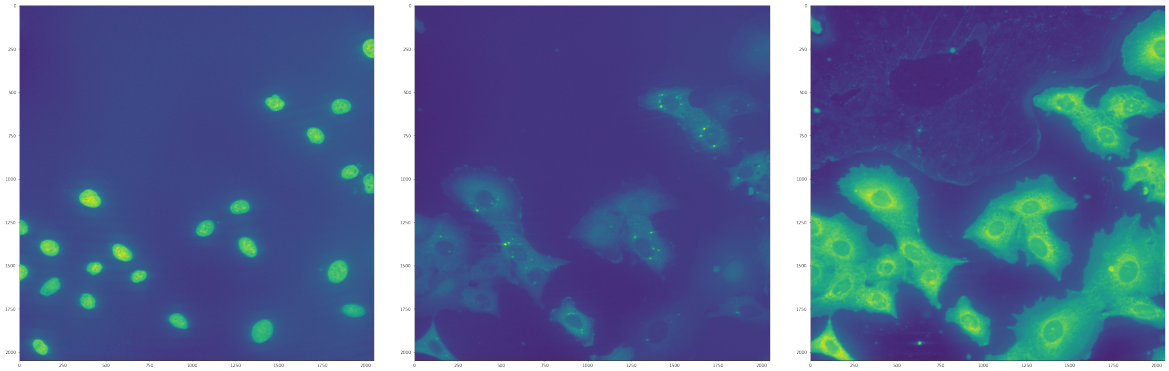
May 4, 2022

Abstract

This report covers an application of supervised machine learning for to the identification of biomolecular condensates in SARS-CoV-2 nucleocaspids. The report covers feature and model selection as well as masking and segmentation for differentiation of condensates respective to the cells in which they are located, their intensities, and their areas.

1 Background

The motivation behind this project is to quantify the effect of different compounds on condensate formation in SARS-CoV-2 nucleocaspids. Alternatives to machine learning for blob detection consist of filtering and thresholding techniques that alter cell images depending on desirable properties. The method outlined in this paper is inspired by Ilastik[1], a pixel classification tool using that uses ensemble methods and differing image filters. Images analyzed in this project were of cells in different stained environments. The data set analyzed in this project contained three different wavelengths for each image, each corresponding to a different goal. These are visualized in Figure 1 for an example well site.



(a) Hoechst stain (b) GFP (c) CellMask

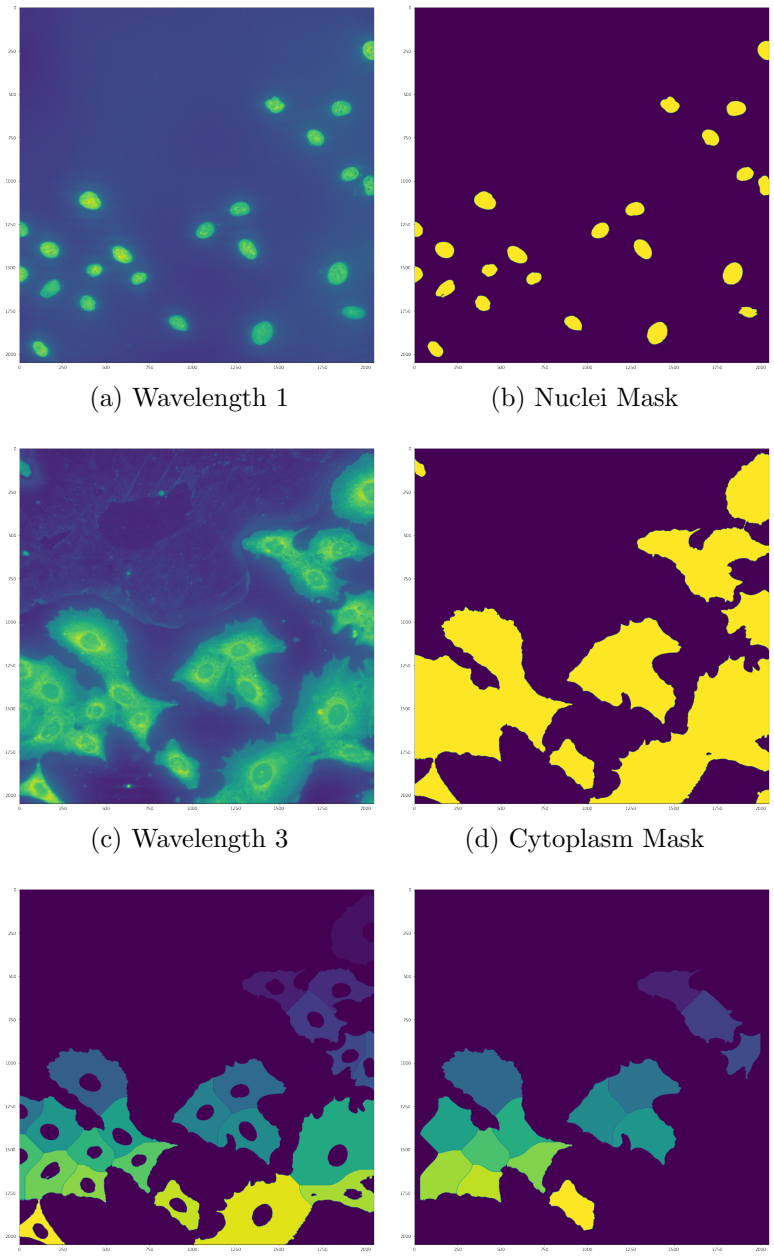
Figure 1: Example of Well Site Wavelengths

Each of these wavelengths is used for a different goal, with the GFP wavelength being the most important for the purposes of pixel classification, as it contains information on condensates as visibly brighter, circular blobs. The other two wavelengths are used in segmenting the cells and removing anomalies in the images posterior to pixel classification.

2 Cell Segmentation

Cell segmentation in this project was required in order to determine what blobs belonged to which cells. A mask is required to accomplish this, which is an array of the same size of the images (2048 by 2048) with zeroes representing background areas (e.g., pixels not part of any cells) and integers for each different cell. After pixel classification is performed on the wavelength containing information on condensates (GFP), the condensates are filtered and counted using this mask.

Figure 2 demonstrates the masking process, which consists of finding nuclei, findingn cytoplasm, and combining these two images to segment the cells using a watershed algorithm. The watershed algorithm takes as input binary images that it treats as topological maps such that they are flooded in order to find the proper separation between regions. This method, along with others in this project, was implemented using the Scikit-Image[2] and Scikit-Learn[3] opensource projects. The nuclei masks were used to signal the location and count of the number of cells in an image, and the cytoplasm masks were used to perform the watershed algorithm following the marking of locations of cells.



(a) Wavelength 1 (b) Nuclei Mask

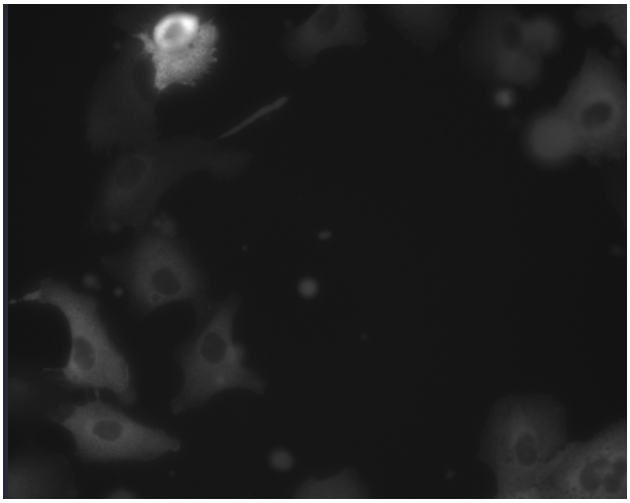
(c) Wavelength 3 (d) Cytoplasm Mask

(e) Segmented Cells with Nuclei (f) Mask

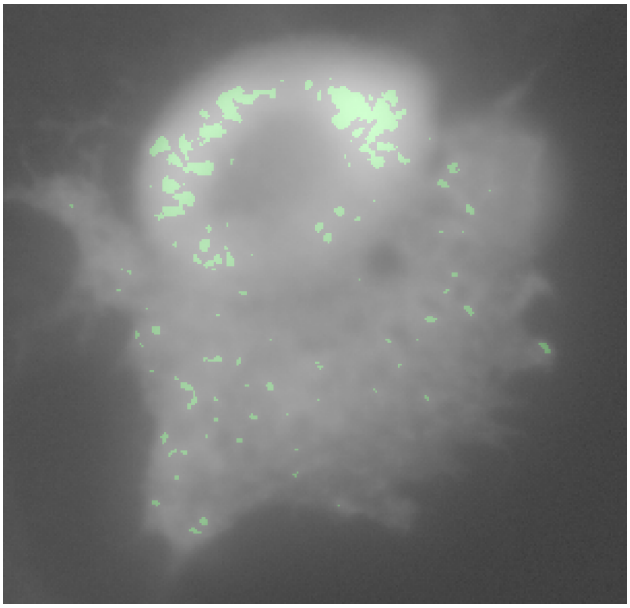
Figure 2: Full Masking Process

A major impediment to accurate blob detection was the presence of overly bright cells in images that would get recognized as containing multiple blobs due to their high pixel intensity despite not containing blobs. An example of this is shown in Figure3. In order to prevent this from happening, pixels with intensity greater than 5 standard deviations from the mean of the image were selected and a morphological opening was performed on the resulting mask.

The result was a mask that contained an area indicated where pixels were abnormally bright but did not include condensates that were overly bright. The detection of the overly bright areas but not overly bright condensates was accomplished using morphological opening; morphological opening removed regions from the outlier mask that were smaller than a constant such that they would not be identified as abnormalities. The full outlier removal process is shown in Figure 4.

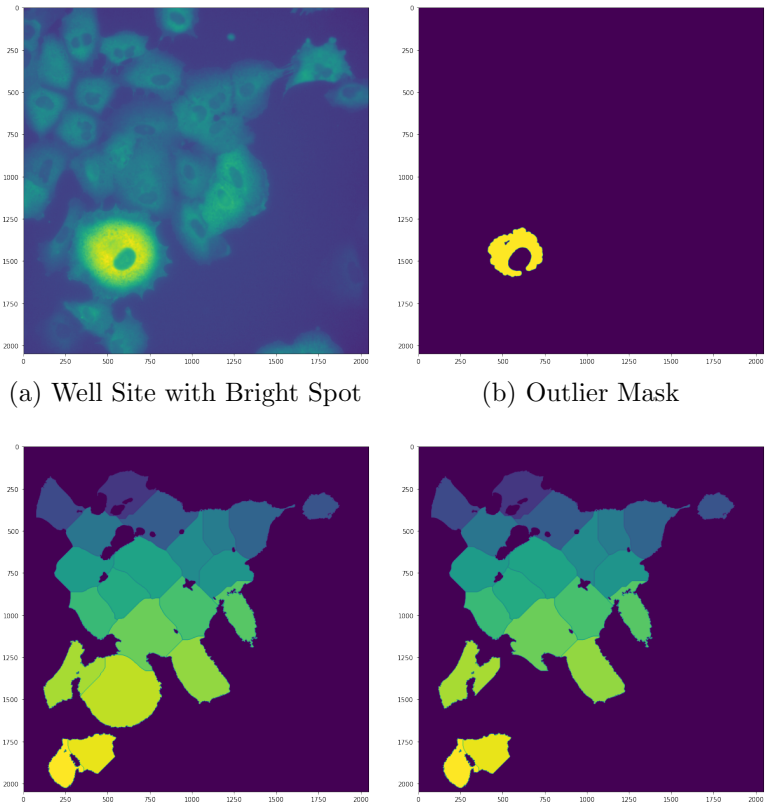


(a) Wavelength 2 with Bright Cell



(b) Effect on Predicted Labels

Figure 3: Bright Spot and Effect on Prediction



(a) Well Site with Bright Spot

(b) Outlier Mask

(c) Mask Prior to Removal

(d) Final Mask

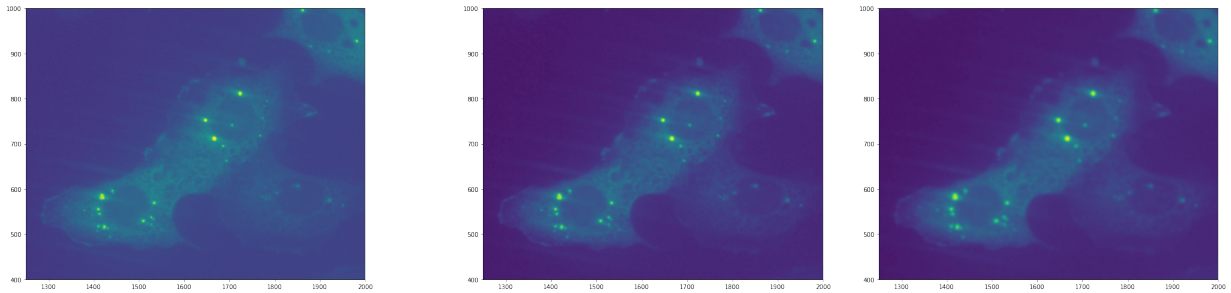
Figure 4: Outlier Removal Process

3 Feature Selection

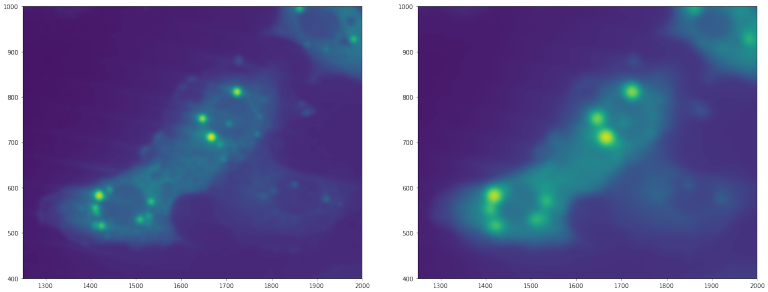
Labeling of images was accomplished with Napari,[4] a Python library for image presentation and labeling. An example of a labeled image portion is shown in Figure 8. Following the labeling of a portion of the full data set, features were generated from the labeled pixels in two ways. The first way was to apply Gaussian blurring, which gave each pixel information on the intensities of nearby pixels in differing ranges. The second was to, in addition to Gaussian blurring, calculate the eigenvalues of the Hessian matrix of the image and append their sum for each set of eigenvalues calculated. This is equivalent to the calculation of the Laplacian of the Gaussian blurred image. These two feature extraction methods were done with for varying variances in the Gaussian blur. Through experimentation, four values for the variance in the Gaussian filter were used.

Figure 5 shows the application of Gaussian blurring to a zoomed-in portion of an image. We refer to these as the intensity features of the image, as the information they provide on the pixel is concerned mainly with intensity at the pixel and in pixels surrounding it.

Figure 6 shows the component gradients of the Hessian matrix and their respective eigenvalues. The sum of these was calculated for each variance in the Gaussian blur such that there were four features for each pixel. Figure 7 shows the difference in these eigenvalues for differing values of the variance of the Gaussian blur. We refer to these features as texture features as they are concerned with how rapidly intensity is changing at and around a pixel, which gives insight into the relative brightness and texture of a pixel.

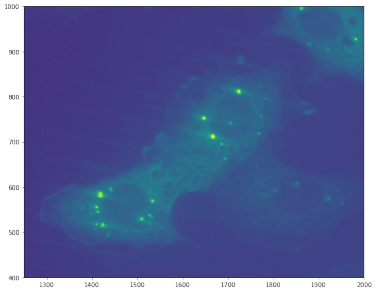


(a) Regular Image (b) $\sigma = 1$ (c) $\sigma = 2$

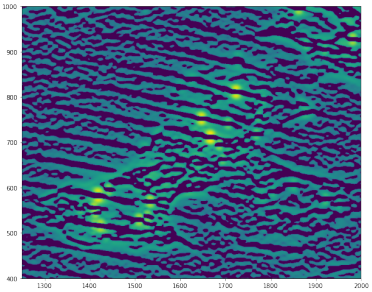


(d) $\sigma = 4$ (e) $\sigma = 8$

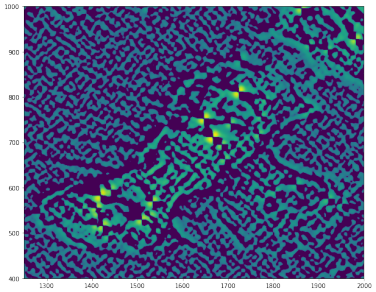
Figure 5: Gaussian Blurring Filters



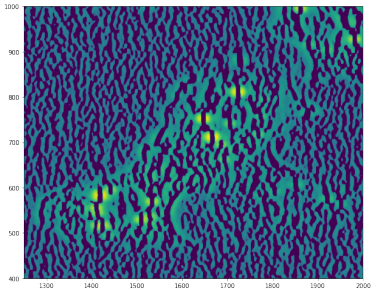
(a) Regular Image



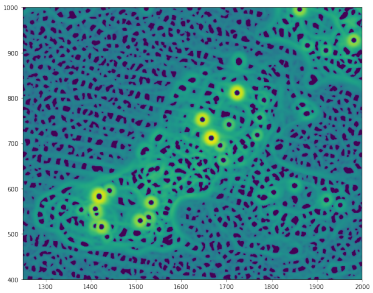
(b) $\sigma = 4$, Hxx



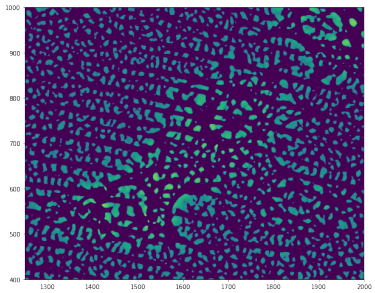
(c) $\sigma = 4$, Hxy



(d) $\sigma = 4$, Hyy

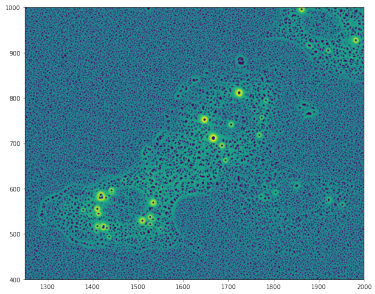


(e) $\sigma = 4$, Smaller Eigenvalues

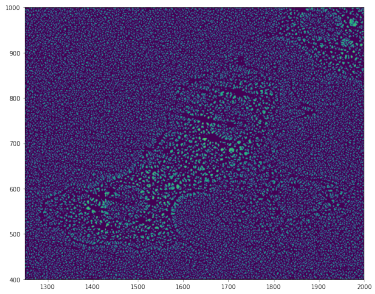


(f) $\sigma = 4$, Larger Eigenvalues

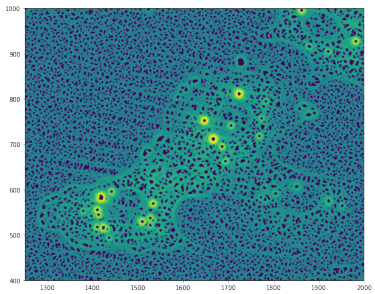
Figure 6: Gradients and Hessian Eigenvalues



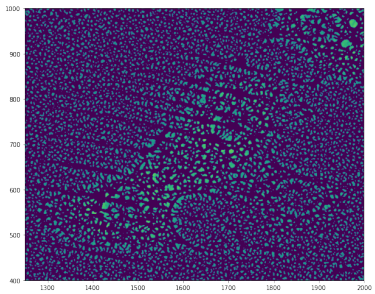
(a) $\sigma = 1$, Smaller Eigenvalues



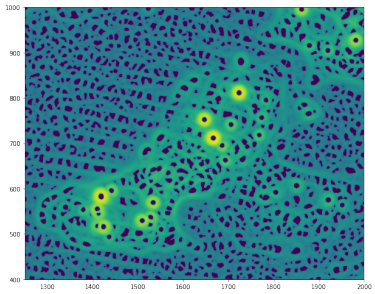
(b) $\sigma = 1$, Larger Eigenvalues



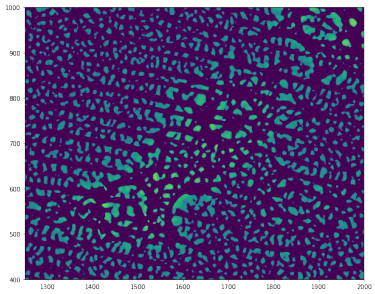
(c) $\sigma = 2$, Smaller Eigenvalues



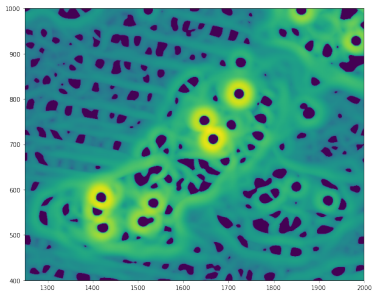
(d) $\sigma = 2$, Larger Eigenvalues



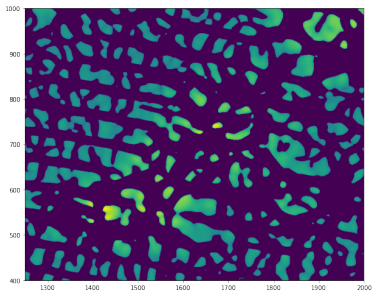
(e) $\sigma = 4$, Smaller Eigenvalues



(f) $\sigma = 4$, Larger Eigenvalues



(g) $\sigma = 8$, Smaller Eigenvalues



(h) $\sigma = 8$, Larger Eigenvalues

Figure 7: Full Range of Texture Filters

4 Model Selection and Results

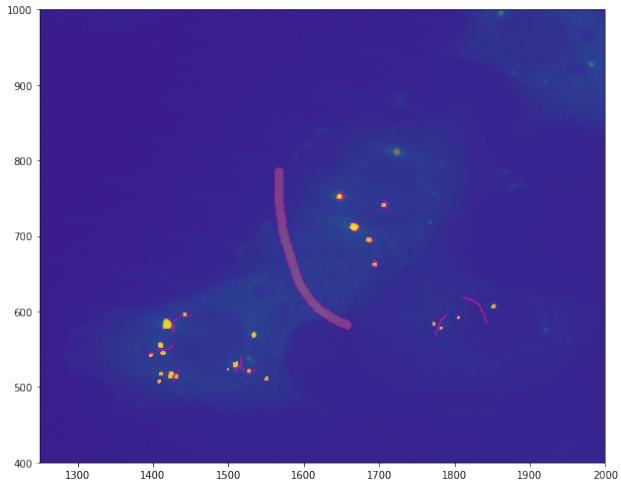
Two models were used for classification, Support Vector Machines and Random Forest Classifiers. These two were chosen for their ability to perform well given smaller training datasets. Differing parameters were tested for each; the parameters varied were the number of Estimators for the random forest classifiers and the C Value for the Support Vector Machine. The resulting balanced accuracies are displayed in Table 1 and Table 2. A visualization of true vs. predicted labels is shown in Figure 8.

# Estimators	Intensity Features	Texture Features
360	0.9813	0.9811
400	0.9816	0.9813
440	0.9818	0.9814
480	0.9807	0.9816
520	0.9814	0.9811

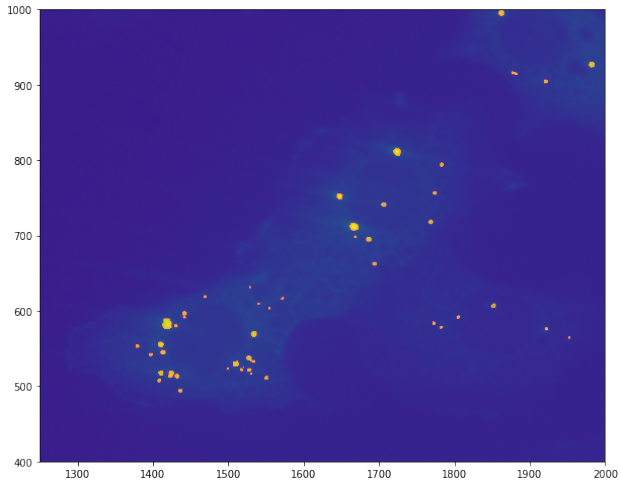
Table 1: Balanced Accuracies of Random Forest Models of Varying Estimator Counts

C Value	Intensity Features	Texture Features
0.001	0.8388	0.8839
0.01	0.8865	0.9330
0.1	0.9257	0.9513
1	0.9513	0.9611
10	0.9701	0.9677

Table 2: Balanced Accuracies of Support Vector Machine Models of Varying C Values



(a) Drawn Labels



(b) Prediction Result

Figure 8: True and Predicted Labels

The results suggest that there is not much benefit to applying filters intended to identify edges and texture like we did above. This is a surprising result, as virtually all traditional methods (e.g., no machine learning) in image processing that are used to identify blobs and other features in images rely on heavy filtering techniques. As for the accuracy of the results, the pixel classification suggests that the methods are accurate in differentiating between pixels belonging to and not belonging to condensates.

An important thing to note is that these accuracies represent the percentage of correctly classified pixels weighted by each class (condensate vs. not condensate). These accuracies do not represent the accuracy of the resulting condensate count. This is because small misclassifications have a large effect on the resulting count. A single pixel misclassified as a condensate will get counted as a small condensate, which will heavily alter the condensate

count, especially in cells with low/no condensates.

The full process, including pixel classification and counting using the mask calculated through cell segmentation, is shown in Figure 9. Corresponding output data, which includes measurements on the condensates such as intensity and area, are depicted in part in Figure 10. The resulting condensate counts of a data set of multiple wells is shown in Figure 11. These results demonstrate the difference in condensate counts in two different types of wells, ones in which a compound was added and one in which one wasn't.

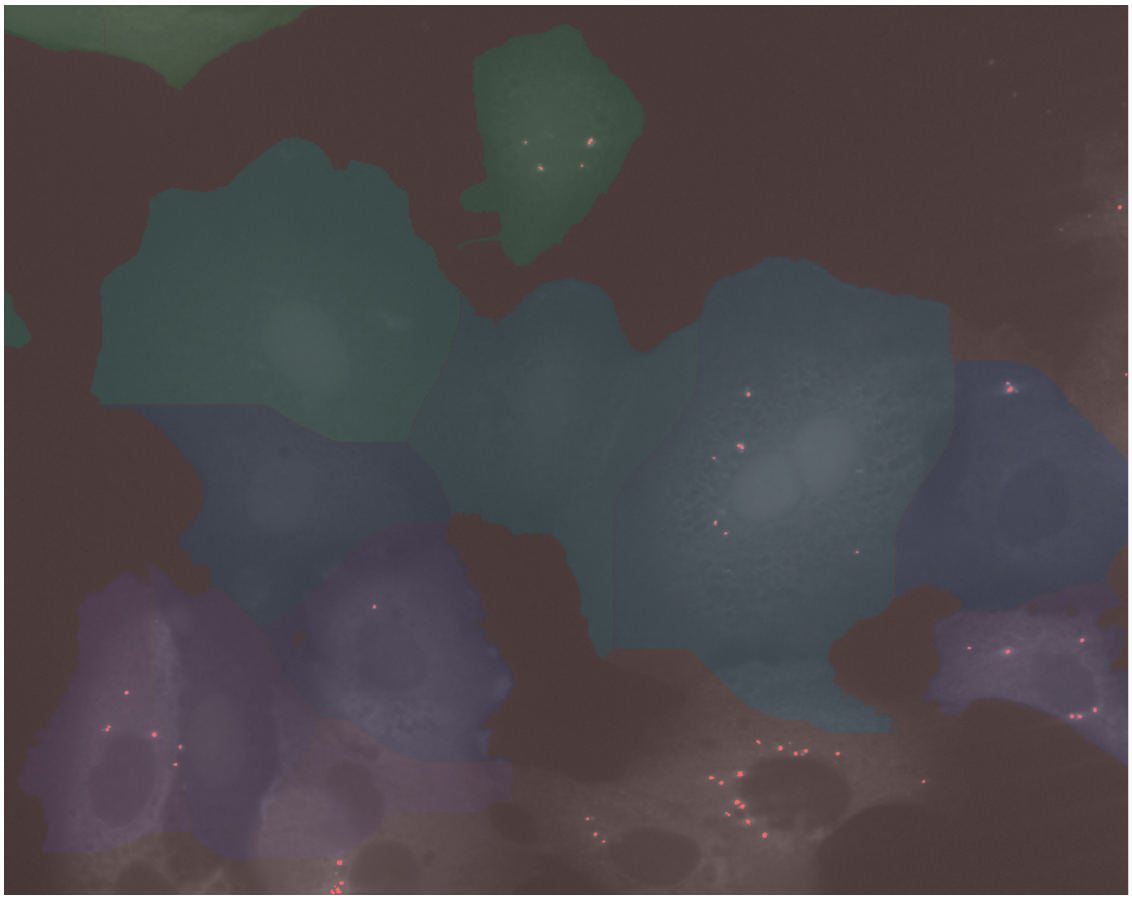


Figure 9: Mask Overlapped with Predicted Labels

well	num_cells	num_cells_with_condensates	num_condensates	avg_condensate_area	avg_condensate_intensity
KRD-211028-p5_A24	33	24	149	2.797000	0.075000
KRD-211028-p5_B24	35	30	239	2.706000	0.083000
KRD-211028-p5_C24	27	23	191	3.829000	0.139000
KRD-211028-p5_D24	65	53	418	3.535000	0.102000
KRD-211028-p5_E24	65	47	232	3.121000	0.070000
KRD-211028-p5_F24	48	42	438	3.304000	0.113000
KRD-211028-p5_G24	59	52	428	3.163000	0.102000
KRD-211028-p5_H24	86	70	526	2.953000	0.086000
KRD-211028-p5_I24	25	22	199	2.874000	0.085000
KRD-211028-p5_J24	54	47	396	3.135000	0.108000
KRD-211028-p5_K24	57	50	387	3.130000	0.094000
KRD-211028-p5_L24	72	62	434	2.853000	0.084000
KRD-211028-p5_M24	36	32	353	2.830000	0.075000

Figure 10: Example Output Data

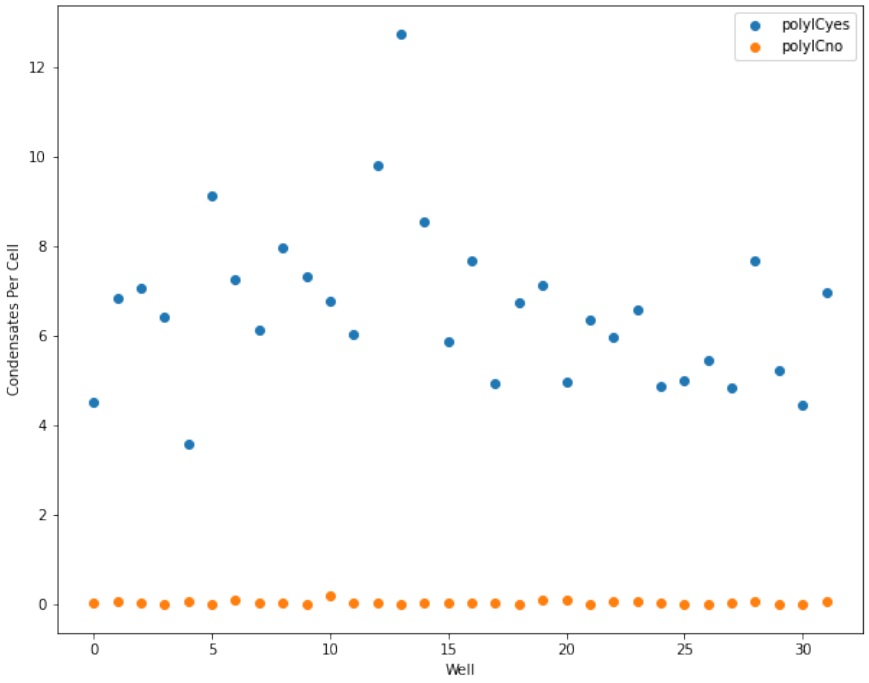


Figure 11: Counts of Wells of Different added Compounds

These results suggest that the machine learning methods were sufficient for robust qualitative assessment of compounds' effects on cells.

5 References

1. Berg, S., Kutra, D., Kroeger, T. et al. ilastik: interactive machine learning for (bio)image analysis. Nat Methods 16, 1226–1232 (2019). https://doi.org/10.1038/s41592-019-0582-9

2. Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. scikit-image: Image processing in Python. PeerJ 2:e453 (2014) https://doi.org/10.7717/peerj.453

3. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

4. napari contributors (2019). napari: a multi-dimensional image viewer for python. doi:10.5281/zenodo.3555620