Cell Segmentation for MERFISH Human Heart Experiments

Jackie Eschbach

Abstract

Cell segmentation is crucial for analyzing spatial transcriptomics data in Multiplexed Error-Robust Fluorescence in situ Hybridization (MERFISH) experiments. This study addresses challenges in segmenting cells in MERFISH images of human heart tissues, exploring limitations of existing methods like thresholding. The research aims to enhance accuracy using machine learning algorithms such as k-means and convolutional neural networks. Initial experiments with baseline thresholding and mini batch k-means clustering on DeepSea's diverse phase contrast microscopy dataset show mixed results. While baseline thresholding is less effective with fewer cells and more background noise, mini batch k-means struggles with noise and accurate boundary identification. Using average Intersection over Union (IoU) scores, mini batch k-means exhibits improved performance compared to baseline on the DeepSea dataset. To overcome current limitations, a U-Net convolutional neural network (CNN) was developed, however, the results were worse than anticipated. Like with thresholding and k-means, the CNN had difficulty distinguishing between the background noise and cell bodies. While results are less than optimal, these findings will still contribute to developing robust cell segmentation methods applicable across staining techniques and tissue types in MERFISH experiments.

1. Introduction

In recent years, there has been a push to map the transcriptome of human tissues to characterize coding and noncoding transcriptional activity. Many methods to do so have been employed, such as single-cell RNA sequencing (scRNA-seq), single-molecule Fluorescence in situ Hybridization (smFISH), and Multiplexed Error-Robust Fluorescence in situ Hybridization (MERFISH). My lab at UC San Diego employs all three of these technologies to characterize the human heart transcriptome, however, my focus has been on the MERFISH technology.

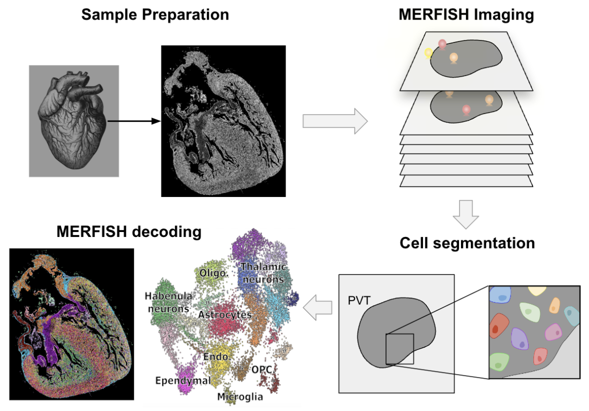

Figure 1. MERFISH experiment pipeline. First, the sample is sliced and prepped, then it is sequentially imaged. After imaging is completed, the images are analyzed to pick out each individual cell boundary. Then the MERFISH fluorescence data is decoded to determine where each mRNA transcript is spatially located.

MERFISH is a spatial transcriptomics imaging method that utilizes fluorescently labeled oligonucleotide probes that bind to specific RNA species, imaged over multiple hybridization rounds (**Figure 1**). Each target RNA species is given a unique barcode of zeros and ones, and a high resolution microscope with multiple wavelength lasers is used to sequentially image the samples. If a fluorescence is present during that round of imaging, a 1 is recorded for the corresponding bit position; if no fluorescence is present, a 0 is recorded. In this way, the multiple rounds of imaging come together to generate the barcodes of the RNA species, allowing for the spatial localization of many genes. Cell boundaries are determined through cell segmentation, and the transcripts are assigned to cells. Once the genes are spatially located, cell types can be determined.

One issue we consistently run into with our MERFISH experiments is with cell segmentation. While we have multiple methods to find the cell boundaries in images, they do so with varying accuracy that cannot always be predicted before testing them on the data. Different tissues have different cell densities, shapes, and sizes, which makes it difficult to have a consistently high-performing cell segmentation

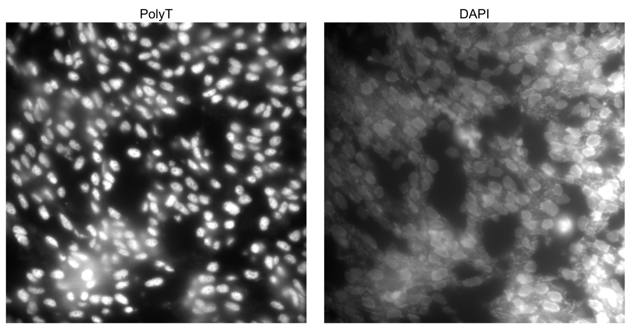

Figure 2. Sample field of view of heart experiment PolyT (left) and DAPI (right) stainings. These images are what are used to segment the cells.

method. Additionally, we utilize different kinds of stainings, which can lead to less accurate cell segmentation (**Figure 2**). We utilize three different stainings: Poly(A)/Poly(T), which stain the cytoplasm; DAPI, which stains the cell nuclei; and more recently, Lamin A, which stains the nuclear lamina. In this paper, I will be investigating how I can utilize machine learning algorithms to create a more robust cell segmentation method that can be used on all types of staining.

My baseline thresholding algorithm, which maps high values to 1 and low values to 0, and then selects the borders of cells based on the binary mapping, has a low accuracy (IoU of 0.08). K-means clustering, specifically with mini batches to improve runtime, also had relatively low accuracy (IoU of 0.13). This low accuracy is possible due to using mini batches rather than the normal k-means clustering, though it has higher IoU scores than the baseline method. Lastly, a U-Net convolutional neural network (CNN) was developed, which also had a low IoU score (0.12). The results look very similar to the mini batch k-means results, where background noise is often picked as cells in the images.

Overall, it is clear that the k-means and thresholding algorithms are incapable of adapting to diverse datasets for cell segmentation. While the CNN also had low IoU accuracy, with a larger and more diverse dataset, I believe it is capable of generating higher accuracy cell segmentations.

2. Related Work

2.1. Watershed Algorithm

Many methods have been proposed to segment various cell types. One popular method is the Watershed algorithm, which creates a topological surface from grey-scale images, then determines the lines along the top of the ridges formed (Bieniek & Moga, 2000). While this method works well in theory, it underperforms when cell fluorescence signals don't smoothly decay from the center of the cell to the cell boundary (which is the case for most nucleic boundary stainings). It also has difficulty with cells that overlap in the images.

2.2. Thresholding Algorithm

Another common method to segment cells is the thresholding algorithm. Thresholding is the mapping of grey-scale image into the binary set 0,1 where 0 is the background and 1 is the cell (Perez & Gonzalez, 1987). The issue with this method is that picking a threshold value can be difficult; similar to the Watershed algorithm, different fluorescence intensity from various cell-types can present issues. However, thresholding is an important step in most cell segmentation algorithms as it allows for clear definition of boundaries, the images just must be preprocessed to maximize the thresholding algorithm's accuracy.

After testing the thresholding algorithm on the DeepSea dataset, it is clear that the method is insufficient for diverse datasets. It often selects the background as part of the cells, especially if there is glare in the image. However, the method is still useful when used in combination with other methods, such as machine learning methods.

2.3. Machine Learning Approaches – Cellpose

More recently, machine learning algorithms have been implemented to segment cells. Deep neural networks have been relatively successful, with their only major downside being the inflexibility of the segmentation (parameters cannot be adjusted as with aforementioned algorithms) (Rune, 2023). One such deep learning-based algorithm is Cellpose, which my lab has utilized in the past. Cellpose generates topological graphs, as Watershed does, through a process of stimulated diffusion that uses manual masks (Stringer

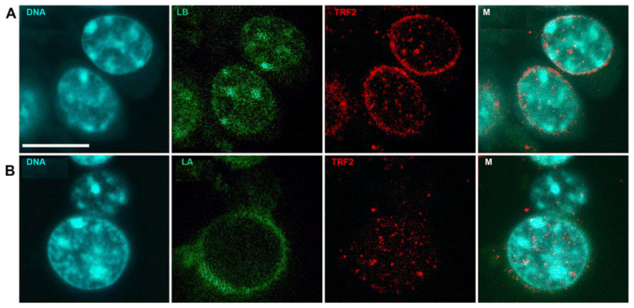

Figure 3. Cumulus cells stained with Lamin B (a) and Lamin A (b). The staining is indicated on each panel: DNA - DAPI; Lamin A (LA) and Lamin B (LB); TRF2 is red. Merged image is marked (M). Bar 10 um for all images (Pochukalina & Podgornaya, 2016)

& Pachitariu, 2021). A neural network is then trained to predict the gradients of those topological maps. Cellpose uses gradient tracking to route all pixels belonging to a cell to the center of that cell, which allows each cell to be identified. While this algorithm works well when we use DAPI, Poly(A), and Poly(T) staining for cell segmentation, it underperforms when we use Lamin A staining. I believe this is due to the fact that Lamin A stains the nuclear lamina, which would give more of a ring of fluorescence rather than the solid fluorescence as shown in **Figure 3**.

After testing my CNN method, it seems that a much more complicated neural network is necessary, one more like Cellpose, to complete this difficult task. My CNN had difficulties handling background noise, which Cellpose does not have difficulty with. Cellpose's training dataset wasn't larger than my own, so it seems that it is likely a difference in complexity of the network rather than just insufficient training (however, Cellpose included some images of random objects rather than just cells to increase the model's ability to handle diverse images, which is something I did not do when training my CNN).

2.4. Machine Learning Approaches – U-Net CNN

In 2015, a group in the Computer Science Department of the University of Freiburg implemented a CNN called the U-Net, which consists of a decoder, a middle layer, and an encoder, specifically for biomedical image segmentation, which is what drew me to use one myself. As shown in **Figure 4**, the encoder consists of convolutional layers with ReLU activation functions and max-pooling to reduce spatial dimensions, the middle layer further reduces spatial dimensions, and the decoder is responsible for upsampling and reconstructing the segmented image (Ronneberger et al., 2015).

After testing the method myself, it doesn't seem to work that well for cell segmentation. I think the U-Net model is usually used for segmentation of electron-microscopy images, which usually have stark boundaries and are therefore much easier to segment than images of cell stainings.

3. Dataset and Evaluation

3.1. Dataset – MERFISH Human Heart Samples

Previously, I intended to use Cellpose's training and test data, which utilizes DAPI staining and is open for public use (Stringer & Pachitariu, 2021). Cellpose's training data includes 540 images, and their test data includes 68 images; however, their masks are blank images and cannot be used. Instead, I am using DeepSea's dataset, developed by UC Santa Cruz (Abolfazl Zargari, 2023). This dataset has 1,853 cell images along with annotated cell masks for each. In particular, it has 444 test images and 1,409 training images, which I split into a dev set (283 images for the dev set and 1,126 for the training set). While this dataset consists of images of phase contrast microscopy, I believe that the task of segmenting cells for these images is similar enough to the task of segmenting cells with Lamin A staining because phase contrast microscopy creates images with a ring border of the cells, which is very similar to Lamin A staining.

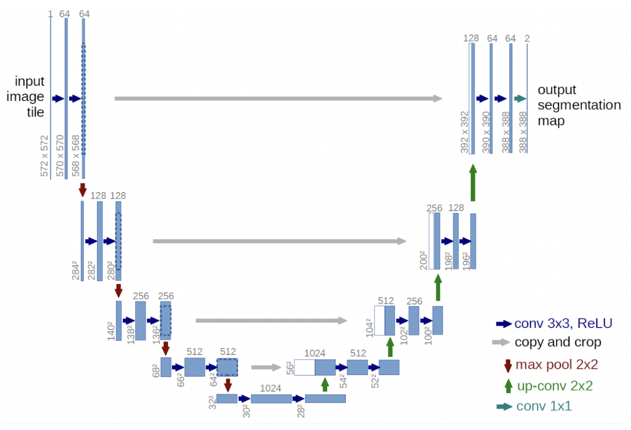

Figure 4. Diagram of the U-Net CNN model architecture (Ronneberger et al., 2015). (See **4.3 Machine Learning Method 2**)

3.2. Evaluation

DeepSea's dataset comes with manually segmented images, which can be used to evaluate the accuracy of my algorithms. These masks can be compared with my model's predictions by using the IoU (intersection over union) score, which is commonly used to evaluate the accuracy of image segmentation tasks. IoU measures the degree of overlap between a predicted bounding box or segment and a ground truth bounding box or segment. Specifically, it is defined as the intersection area divided by the union area of the two regions. Typically, an IoU score of 0.5 or above is considered acceptable, and a 0.7 or higher is considered good for image segmentation.

4. Methods

4.1. Baseline Method

As a baseline, I used simple thresholding. First, the input image is loaded and converted into greyscale. Next, the contrast is enhanced with histogram equalization (Mustafa & Kader, 2018), contrast stretched between the 2nd and 98th percentile to stretch the range of intensity values, and then Otsu's thresholding is applied to create a binary image (Otsu, 1975). Once the image has been converted to binary, the cells are selected as any region with a value of 1, and then the selected cells are filtered for size (greater than 100

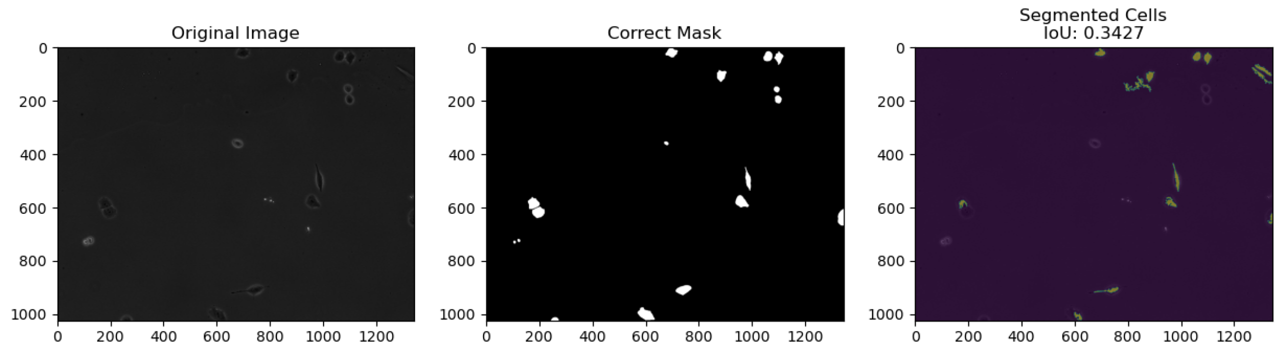

Figure 5. Original cell image (left) and correct mask (middle) from DeepSea's test set (Abolfazl Zargari, 2023), and cells segmented by my baseline algorithm (see **4.1 Baseline Method**).

pixels) so no noise is marked as a cell. Next, the initial cell mask is created. Since the cells have strong borders, these initial cell masks end up selecting the center of the cells as background. Therefore, I invert the cell mask and apply another threshold to select only regions with high enough area (greater than 300 pixels). This is turned into the final binary cell mask prediction.

4.2. Machine Learning Method 1 – Mini Batch K-Means

My first attempted machine learning program to perform cell segmentation on raw images was with k-means clustering. After loading and preprocessing the data (just changing images from RGB to greyscale), I used sklearn's mini batch k-means clustering with a batch size of 800, maximum iteration of 100, and 2 clusters. Mini batch k-means is a variation of k-means that uses random subsets of the dataset to perform updates (Thankachan, 2023). Similar to k-means, mini batch k-means randomly selects points (2 since 2 clusters) to be centroids of the clusters, and calculates the distance of all other points in the batch (also randomly chosen) to those centroids. The data points are assigned to the cluster closest to them. Once all the data points in the batch are assigned, the centroids are recalculated. This process is repeated until convergence, or until the maximum number of iterations (100) is reached.

I had previously attempted to use sklearn's regular k-means clustering (which uses the whole dataset instead of subsets), however, the runtime was far too long (almost an hour) because of the size of my dataset of large-sized images.

4.3. Machine Learning Method 2 – Convolutional Neural Network (CNN)

After implementing mini batch k-means, I developed and trained a U-Net convolutional neural network (CNN) using PyTorch. My CNN first loads in the data, and then has an encoder, middle layer, and then decoder (a U-Net model, which is commonly used for cell segmentation). Similar to the architecture shown in **Figure 4** (though my image sizes were different and I had less layers), the encoder consists of two convolutional + ReLU layers (number of output channels is 64, kernel size of 3, padding of 1, for both), followed by max pooling (kernel size of 2, stride of 2). The middle layer has the same layers as the encoder, except the

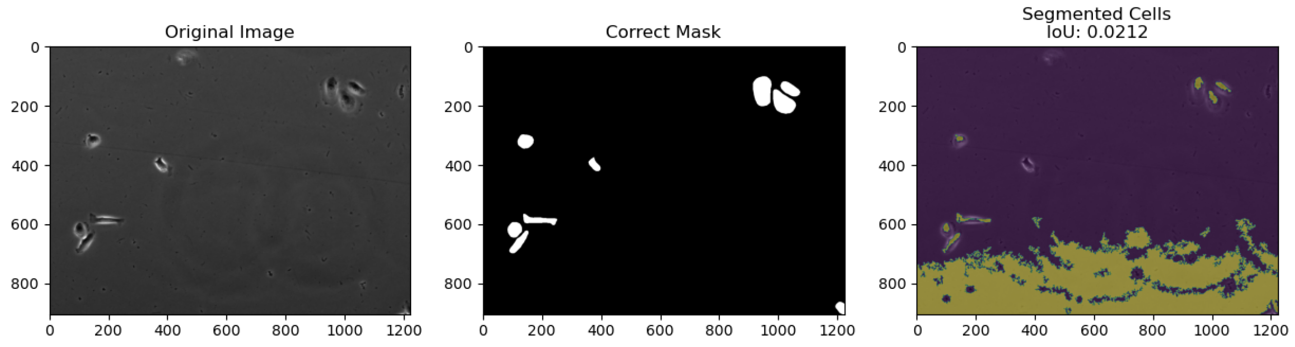

Figure 6. Original cell image (left) and correct mask (middle) from DeepSea's test set (Abolfazl Zargari, 2023), and cells segmented by my baseline algorithm (see **4.1 Baseline Method**).

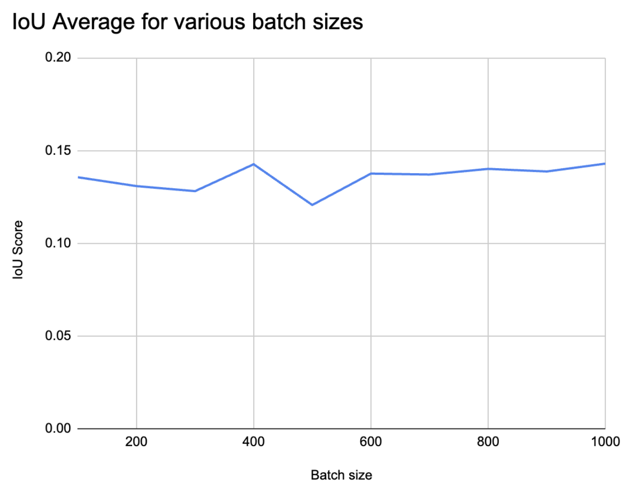

Figure 7. Average IoU scores for mini batch k-means with various batch sizes.

convolutional layers have 128 output channels. Lastly, the decoder has the same layers, but with 64 output channels for the convolutional layers, and instead of pooling, there is a convolutional transpose layer with 1 output channel, kernel size of 2, and stride of 2. After the decoder, the dimensions of the output are adjusted to match the input size by using PyTorch's interpolate function.

Hyperparameters for a CNN include the number of layers, which non-linearity to use, kernel size, padding, and output channel size for the convolutional layers. Because of the results shown in **5.3 Machine Learning Method 2 Results**, I decided to use padding of 1 and kernel size of 3, which is typical for many CNNs, and I used ReLU for my non-linear layers.

The CNN method performs worse than the mini batch k-means algorithm because while the CNN is able to learn a lot more features than a k-means clustering algorithm is capable of learning, I think the amount of training data was insufficient and there wasn't enough diversity in my training set. Also, including more layers like the original U-Net model may improve the algorithm's performance.

5. Experiments

5.1. Baseline Results

Overall, my baseline algorithm had an average IoU of 0.08, which is extremely low. Looking closer, it seems that the algorithm worked well for some images, but poorly for others. For example, **Figure 5** shows a good prediction of where the cells are (IoU of 0.3427), however, **Figure 6** shows an example of a poor prediction (IoU of 0.0212) when the background is selected along with the cells.

5.2. Machine Learning Method 1 (K-Means) Results

In order to determine the best batch size for the mini batch k-means algorithm, I tested on the dev set for batch sizes varying from 100 to 1000. As shown in **Figure 7**, IoU scores were relatively similar and plateau off after batch size of 600. Therefore, I chose to go with batch size of 800 because it had a low runtime (12 seconds for 283 images) and a relatively high IoU (0.1401029687). Adjusting the maximum iterations did not make significant changes to the IoU values, so I stuck with the standard of 100 max iterations. To determine the best number of clusters, I calculated the average IoU for the dev set for various numbers of clusters (using 800 batch size and 100 max iterations), as shown in **Figure 8**. Since 2 clusters had the highest IoU (0.1401029687), I used 2 clusters for the rest of my analyses.

Initial results are mixed, similar to the baseline results. On

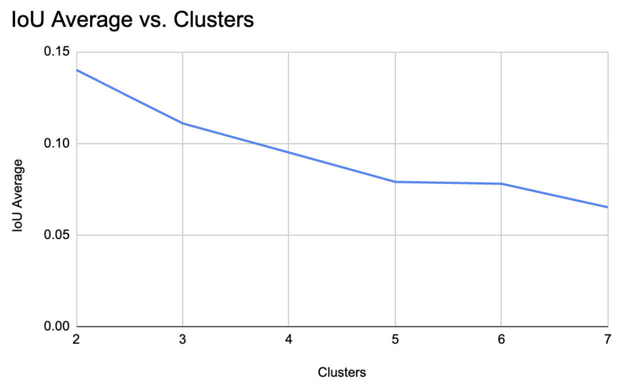

Figure 8. Average IoU scores for mini batch k-means with various cluster sizes.

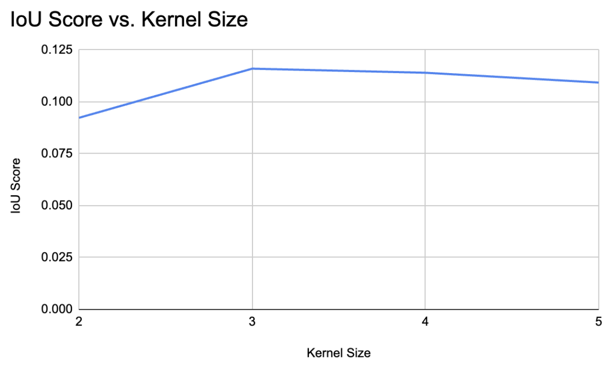

Figure 9. Average IoU scores for CNN with various kernel sizes.

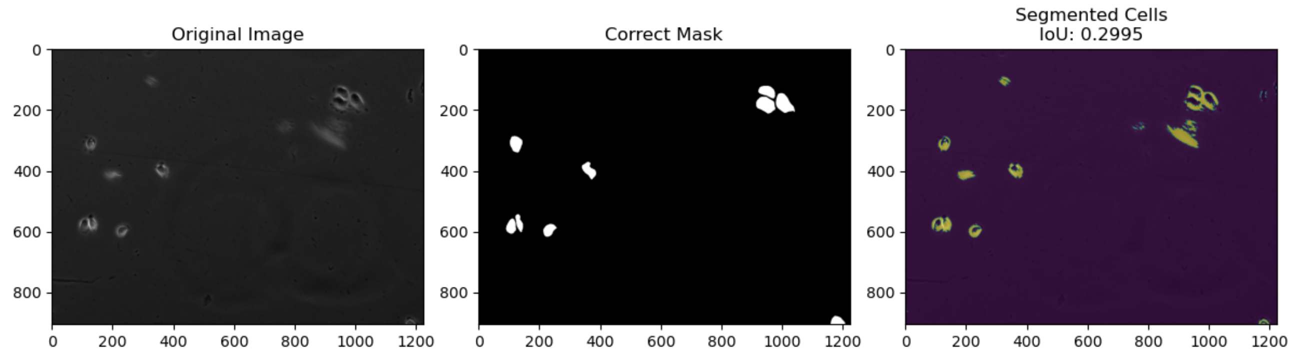

Figure 10. Original cell image (left) and correct mask (middle) from DeepSea's test set (Abolfazl Zargari, 2023), and cells segmented by sklearns' mini batch k-means clustering algorithm (see **5.2 Machine Learning Method 1**).

the one hand, **Figure 10** shows an example good quality result (IoU of 0.2995) for the mini batch k-means, whereas **Figure 11** shows an example of a low quality result (IoU of 0.0222) on the test set. Overall, this algorithm had an IoU average of 0.1267 on the test set, which is higher than the baseline algorithm (see **Table 1**). I believe the mini batch k-means algorithm is better than baseline thresholding algorithm because it can handle varying levels of intensity for cells, whereas thresholding depends on an intensity value to identify cells.

5.3. Machine Learning Method 2 (CNN) Results

In order to determine the best hyperparameters for the CNN, I tested on the dev set. Hyperparameters for a CNN include the number of layers, which non-linearity to use, kernel size, padding, and output channel size for the convolutional layers. Because of the large number of possible hyperparameters, I kept the number of layers and the number of output channels constant (to prevent excessive runtime, which is already exceeding 13 hours). I tested kernel sizes of 2 to 5, results shown in **Figure 9**. Because of the results, I decided to use a kernel size of 3, which is typical for many CNNs (such as the original U-Net model) anyways.

The overall results for the CNN were mixed, similar to the k-means results. **Figure 12** shows an example of a visually successful segmentation, however, its IoU score was very low (0.06). This is because the algorithm selected the wrong shapes for the cells, even though it picked the right location for the cell to be in. **Figure 13**, on the other hand, shows a poor segmentation mask with a higher IoU. The CNN chose almost all of the background to be labeled as a cell, which gave it a higher IoU score because of the sheer number of cells in the image. It is clear that the glare in the image caused the algorithm to select the entire background.

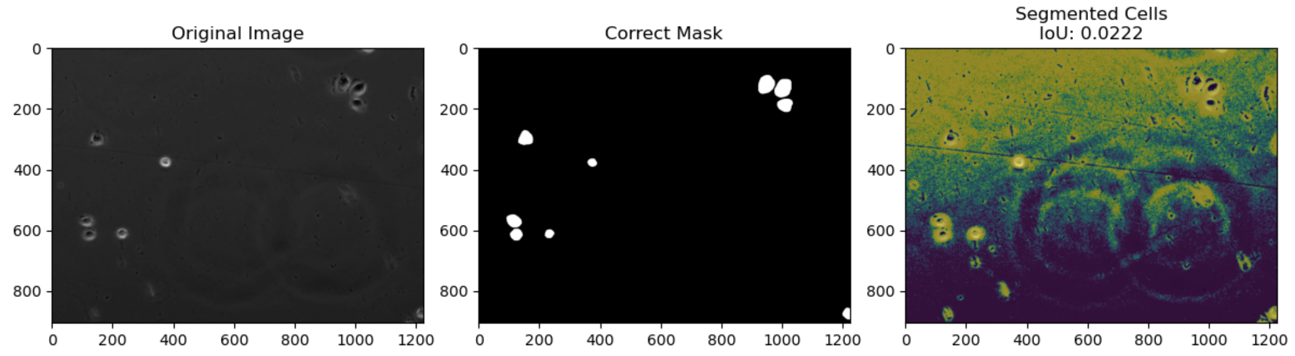

Figure 11. Original cell image (left) and correct mask (middle) from DeepSea's test set (Abolfazl Zargari, 2023), and cells segmented by sklearns' mini batch k-means clustering algorithm (see **5.2 Machine Learning Methods 1**).

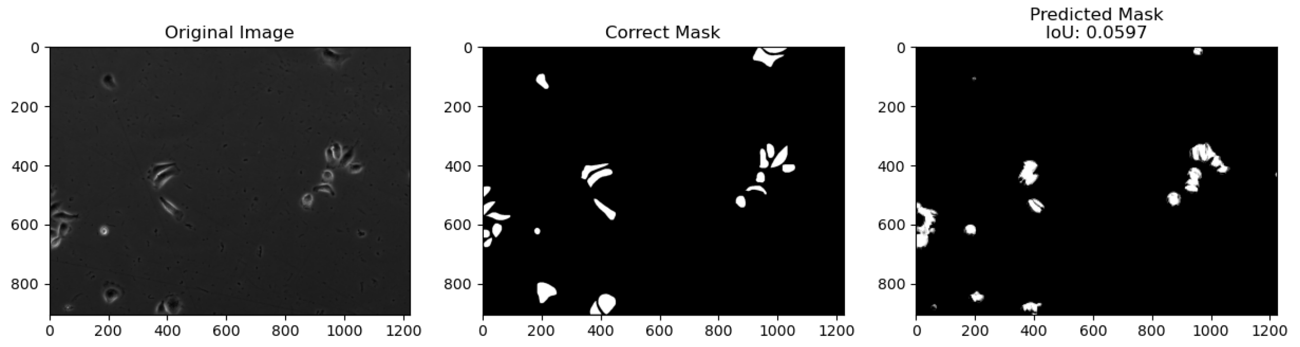

Figure 12. Original cell image (left) and correct mask (middle) from DeepSea's test set (Abolfazl Zargari, 2023), and cells segmented by U-Net CNN algorithm (see **5.3 Machine Learning Method 2**).

Table 1. Average IoU scores for baseline and mini batch k-means algorithms on DeepSea's dataset.

Data set	Baseline	K-Means	U-Net CNN
DeepSea	0.07982	0.1267	0.1160

While the CNN performed better than the baseline algorithm on average, it performed worse than the k-means algorithm (see **Table 1**). However, looking at the results manually, I think that the CNN and k-means algorithms performed very similarly. I think this is because while the baseline algorithm just selects anything above a certain threshold and therefore cannot discern between noise and cells, the k-means and CNN are learning some amount of information about the cells' properties. Additionally, the CNN has made some clear choices on what the cell shapes are like, where as k-means and thresholding just picks the bright spots. I think that the CNN didn't perform as well as expected because it wasn't trained enough with a diverse enough dataset.

6. Discussion

The baseline algorithm has low IoU, particularly for images with a low number of cells. The mini batch k-means clustering algorithm a higher IoU than the baseline algorithm, though it was still very low particularly because it had difficulty with images with background noise (like a glow from the phase contrast microscopy).

The mini batch k-means clustering just selected any bright spots. For example, in **Figure 10**, the bright spot at around (900, 300) is clearly glare to the human eye, but it is identified as a cell by the k-means clustering. This result was expected for this algorithm, as it just grouping similar points from the image together.

One way to improve this algorithm is to just use the regular/vanilla k-means, which will guarantee correct centroids for the clusters, whereas the mini batch k-means could have the wrong centroid because it is just picking a random subset of the data to calculate it. Another way to improve the algorithm is to include more preprocessing. If the background noise is significantly decreased, the k-means clustering will be less likely to pick out the noise as cells (like what hap-

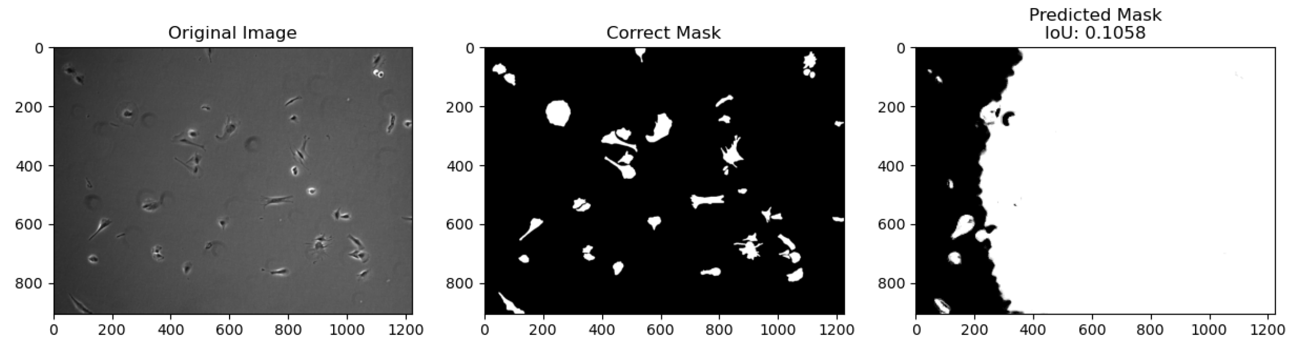

Figure 13. Original cell image (left) and correct mask (middle) from DeepSea's test set (Abolfazl Zargari, 2023), and cells segmented by U-Net CNN algorithm (see **5.3 Machine Learning Method 2**).

pened in **Figure 11**). This can be done with smoothing filters or background subtraction. Mini batch k-means also doesn't always select the entire cell, since the cell's borders are brighter than the center. This can be remedied with the watershed algorithm (see **2.1 Watershed Algorithm**).

The U-Net CNN I developed had similar issues to the mini batch k-means algorithm, especially with background noise. Therefore, I think a similar fix (using the watershed algorithm) would allow for less background noise and therefore cleaner segmentation. Also, more preprocessing (such as using some form of thresholding) could allow the glare in the background, like in **Figure 13**, to be dealt with properly.

Overall, I think that this algorithm performed worse than I expected, as I thought the CNN would be able to learn features that the k-means wasn't able to learn (like that the cells are outlines and not circles, that there is glare in the images, and the relative size of the cells); however, it seems it wasn't able to learn much. This could be because my training dataset wasn't diverse enough or large enough, or that I didn't have enough layers in my CNN.

7. Conclusion

Overall, the baseline method and mini batch k-means performed poorly, due to their inability to adapt to diversities in the datasets. The U-Net CNN also performed poorly on images with high background noise because it wasn't able to learn what is a cell and what is not even though it had extensive training. While it performed better than the baseline thresholding method, I would still like to continue to improve the CNN to be able to handle even more diversity in the images.

Link to GitHub repo:

https://github.com/eschbachj/CSCI467FinalProject

Link to DeepSea Dataset:

https://deepseas.org/datasets/

Note: The DeepSea dataset has a lot of images, but not all of them have a correct mask pair, so I only used the images with a paired mask (so my training dataset was smaller than what they provided).

References

Abolfazl Zargari, Gerrald A. Lodewijk, N. M. A. B. L. H. S. A. S. Deepsea is an efficient deep-learning model for single-cell segmentation and tracking in time-lapse microscopy. *Cell Reports Methods 3*, 2023.

Bieniek, A. and Moga, A. An efficient watershed algorithm based on connected components. *The Journal of the Pattern Recognition Society*, 2000.

Mustafa, W. A. and Kader, M. M. M. A. A review of histogram equalization techniques in image enhancement application. *Journal of Physics: Conference Series*, 2018.

Otsu, N. A tlreshold selection method from gray-level histograms. *IEEE*, 1975.

Perez, A. and Gonzalez, R. An iterative thresholding algorithm for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1987.

Pochukalina, G.N., I. N. and Podgornaya, O. Nucleolus-like body of mouse oocytes contains lamin a and b and trf2 but not actin and topo ii. *Molecular Cytogenetics*, 2016.

Ronneberger, O., P.Fischer, and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pp. 234–241. Springer, 2015.

Rune, J. *Developing Automated Cell Segmentation Models Intended for MERFISH Analysis of the Cardiac Tissue by Deploying Supervised Machine Learning Algorithms*. PhD thesis, Royal Institute of Technology, 2023.

Stringer, C., W. T. M. M. and Pachitariu, M. Cellpose: a generalist algorithm for cellular segmentation. *Nature Methods*, 2021.

Thankachan, K. Data scientists' interview guide: k-means. *Medium*, 2023.