Optimization of Mesmer, A Deep Learning Model for Cytoplasmic Segmentation Using LIVECell Dataset

Liang Yi
chnlyi@stanford.edu

Wei Da
wda0510@stanford.edu

Zhaoqiang Bai
baizq871@stanford.edu

Abstract

In this project, we are targeting to resolve the cell segmentation problem on LIVECell dataset.

We build an end-to-end pipeline on cloud for Mesmer to take cell images in LIVECELL as input. With tuning optimization hyper-parameters and selecting different semantics transformation and backbone models, we are able to get a workable model and study how the actual segmentation problem defined and resolved.

1. Introduction

Cell segmentation plays a crucial role in understanding, diagnosing, and treating diseases. Despite the recent success of deep learning-based cell segmentation methods, it remains challenging to accurately segment densely packed cells in cell membrane images. There are two critical challenges for solving the segmentation accuracy problem: 1) establish a scalable approach for generating large volumes of pixel-level training data in tissue images and 2) set up an integrated deep learning pipeline that utilizes these data to achieve human-level performance. As for the first challenge, recent advances in multiplexed imaging techniques have drastically expanded the amount of high-quality, open-source datasets that characterize full breadth of human tissues with annotated location, function, and phenotype of the cells therein. Outstanding among these datasets is the LIVECell [1] – a large, high-quality, manually annotated and expert-validated dataset of phase-contrast images with more than 1.6 million cells. Regarding the second challenge, we have recently witnessed a variety of visual recognition algorithms that facilitate the location and segmentation of single cells in images. Mesmer, [2] – a deep learning-enabled segmentation algorithm, is a cutting-edge model that has been demonstrated to achieve human-level performance. It is interesting to combine the two pieces of the state of the arts and explore whether and how we can optimize the Mesmer algorithm so that it works well on the Livecell dataset, through tuning backbone models, hyper-parameters, semantic transformations, among other knobs. In this paper, we apply Mesmer on the LIVECell dataset in an attempt to resolve the segmentation problem for cells.

The trained Mesmer models take the whole cell images as input and return predicted segmentation as output. We build an training-predicting end-to-end pipeline for Mesmer and apply it to different size of LIVECell data with tuning all hyper-parameters (e.g inner/outer distance distance for semantic transformation) and model architectures (e.g. backbone models for image classifications).

To accurately gauge the models' performances, we reuse the evaluation method provided by Mesmer with multiple metrics (e.g precision/recall/ average pixel-wised Intersection over Union (IoU)) and counts of different error types) between the predicted segmentation and the ground-truth labels.

The final results shows that, with pre-processing, Mesmer can work on the bigger cell-image dataset with different data format . Also, with tuning the models, we get better understanding about the importance of different transforms/features/parameters and how it will affect the final segmentation results. We also able to generate bounding boxes on the examples of cell images to visualize the model output.

This project can be used as benchmarks and compared with other segmentation models on LIVECell. It also helps us to better understand how segmentation works on real world.

2. Related Work

Traditional methods. Before deep learning became popular, a variety of techniques, such as the intensity thresholding method [3] and the ACME filtering method [4], were applied to cell segmentation. These methods suffered from lower segmentation accuracy as compared to cutting-edge deep-learning based cell segmentation methods and many of them were very labor-intensive as well.

Deep-learning methods. Deep convolutional-neural-network (ConvNet) based approaches present a higher adaptability and higher segmentation accuracy across different cell morphologies and categories. [5] One flavor of the deep-ConvNet based methods take the "object detection-based" strategy - bounding boxes are first predicted for all cells captured in an image, followed by mask generation for each cell. Belonged to this strategy are methods such as Retinanet [6], R-CNN and its series of derivatives [7–9], keypoint bounding box [10], and week annotation [11]. Among all of these methods, Mask R-CNN has achieved the highest accuracy [12], and hence been widely applied in various segmentation tasks. The main drawback of these models, however, is that they rely too much on the setting of the number of anchor boxes, whose performance varies on different datasets. Also, they tend to fail in situations where objects poorly approximated/identified with bounding boxes.

Another category of deep-ConvNet based methods use a 2-step strategy: 1) First treat the task as semantic segmentation by using deep ConvNet models to provide pixel-wise prediction of cell interiors and edges, and backgrounds [13]; 2) Then generate predicted masks for different cells by applying clustering methods to the output of step 1. These methods are "contour-aware approaches", because the second step, which is essentially an instance segmentation, relies on accurately classifying all contour pixels of cells . Cutting-edge methods such as U-Net [14], DeepCell [15] and Mesmer are among this category.

As a side note, there recently emerges such model as NucleAIzer [16], which combines a Mask R-CNN model and a U-Net model and shows promising performance by taking advantages of the strengths of both the object detection-based strategy and the contour-aware strategy. In this project, we tried to implement this model as well, but were not successful due to the extreme complexity of the model and the time/resource constraint. We decide to put this Mask R-CNN + U-Net framework into the scope of our future work, as specified in Section 6.

3. Methods

As shown in Fig. 1, the Mesmer deep learning model we used for segmentation is essentially a Feature Pyramid Networks (FPN), which consists of an usual deep convolutional network (ConvNet) backbone, that is connected to a feature pyramid.

Feature Pyramid Networks. The FPN in the Mesmer architecture consists of a bottom-up pathway, a top-down pathway, and lateral connections [17] . Images are fed into a deep Convnet backbone coupled to a feature pyramid network. Two semantic heads produce pixel-level predictions. The first head predicts whether each pixel belongs to the interior, border, or background of a cell, while the second head predicts the centroid of each cell.

- **The backbone models.** ResNet50 [18] is our baseline backbone model. The ResNet includes skip connections to the otherwise "plain" ConvNet. These skip connections work in two ways. Firstly, they alleviate the issue of vanishing gradient by setting up an alternate shortcut for the gradient to pass through. In addition, they enable the model to learn an identity function. This ensures that the higher layers of the model do not perform any worse than the lower layers.

  DenseNet [19] is an alternative backbone model we used in this work. It also embraces a similar underlying principle that ConvNets can be made substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output. DenseNet utilises dense connections between layers, through Dense Blocks, where all layers with matching feature-map sizes are directly connected with each other. As illustrated in Fig. 2, the difference between ResNet and DenseNet is that ResNet adopts summation to connect all preceding feature maps while DenseNet concatenates all of them.

  EfficientNet is also another alternative backbone model we chose. EfficientNet is all about engineering and scale - Instead of navigating complex architectures, it starts with a relatively small baseline model and gradually scales in terms of network depth (more layers), width (more channels per layer), and resolution (input image), either individually or simultaneously. Unlike conventional approaches that arbitrarily scale network dimensions, such as width, depth and resolution, EfficientNet uniformly scales each dimension with a fixed set of scaling coefficients. In many applications, EfficientNet shows significant improvements in both accuracy and efficiency compared with ResNet50 under similar FLOPS. [20]

- **The bottom-up pathway.** The bottom-up pathway is used for feature extraction. It could in principle be any usual deep convolutional network. In our case, we used Resnet50 as the baseline, and tried DenseNet, and EfficientNet as alternatives. The bottom-up pathway consists of a number of convolution modules, each of which has many convolution layers. As data flow upstream, the spatial resolution decreases, while the semantic value for each layer increases as more high-level structures are detected. The output of each convolution module is labeled as Ci and later used in the top-down pathway.

- **The top-down pathway.** The top-down pathway, on the other hand, is used to construct higher resolution layers from the semantic rich layer(s). It extracts higher-resolution features by upsampling feature maps from higher pyramid levels that are spatially coarser but semantically stronger. Then each lateral connection merges feature maps of the same spatial size from

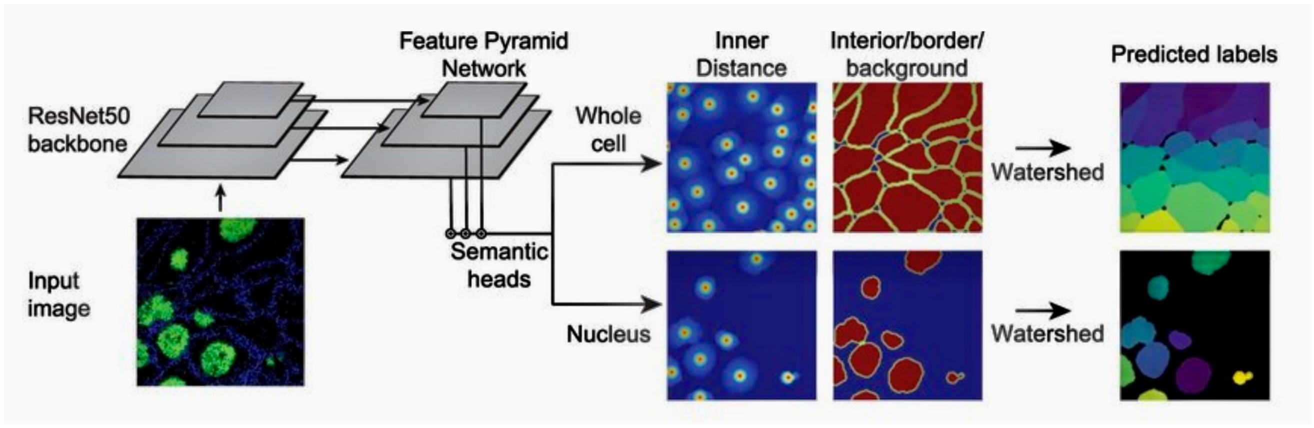


Figure 1. Mesmer network architecture. Adopted from Ref. [2]

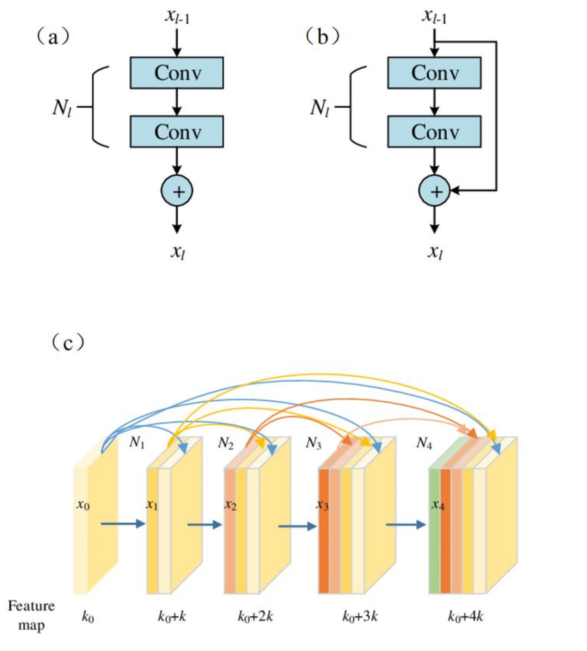


Figure 2. (a) a ConvNet block (b) a ResNet block (c) DenseNet blocks

the bottom-up path-way and the top-down pathway. Such merge operation enhances the top-down features because, although the bottom-up feature map is of lower-level semantics, its activations are more accurately localized as it was subsampled fewer times. Fig. 3 shows the building block that constructs the top-down feature maps. With a coarser-resolution feature map, the spatial resolution is upsampled by a factor of 2. The upsampled map is then merged with the corresponding bottom-up map (which undergoes a 1×1 convolutional layer to reduce channel dimensions) by

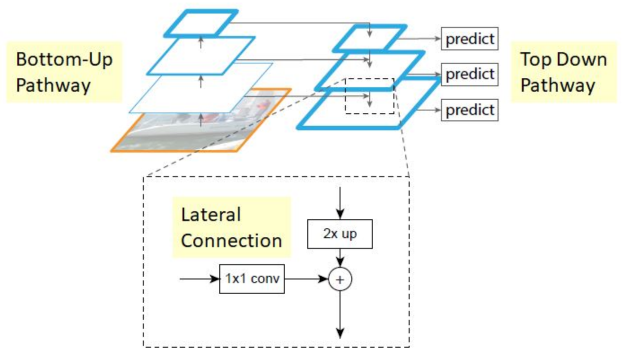


Figure 3. Feature pyramid network architecture [17]

element-wise addition. This process is iterated until the finest resolution map is generated. To start the iteration, a 1×1 convolutional layer is attached to C5 to produce the coarsest resolution map. Finally, a 3×3 convolution on each merged map is appended to generate the final feature map, which is to reduce the aliasing effect of upsampling. This final set of feature maps is called {P2, P3, P4, P5, P6, P7}, corresponding to {C2, C3, C4, C5, C6, C7} that are respectively of the same spatial sizes. Before being entered into the backbone model, images are concatenated with a coordinate map. In the Mesmer case, backbone layers C3–C5 and pyramid layers P3–P7 are used; for the pyramid layers depthwise convolutions are used for computational efficiency.

Semantic heads. Two semantic segmentation heads are attached to the top of the feature pyramid that perform upsampling to produce predictions the same size as the input image. One of the semantic heads is attached to a 'pixel-wise transform', while the other to an 'inner distance transform'. The pixel-wise transform is a prediction of whether each pixel belongs to the cell interior, cell boundary or background. The second transform captures the distance of each pixel inside a cell to that cell's centroid. If the distance of a cell's pixel to that cell's centroid is r, then we compute 'inner distance transform' through $\frac{1}{1+abr}$, where $a = 1/\sqrt{A}$, with A being the cell area, and b is a hyperparameter taken as 1 in both original Mesmer paper and our work. The softmax loss was used for the semantic head assigned to the pixel-wise transform and the mean squared error for the semantic head assigned to the inner distance transform. Similar to the orginal Mesmer paper, the softmax loss was scaled by a factor 0.01 so that training can be stabled. [2]

Watershed and postprocessing. Marker-based watershed [21] was used as a postprocessing step to convert continuous predictions from pixel-wise transform and inner distance transform into discrete label images, where the pixels belonging to each cells are assigned their unique integers. To perform this postprocessing step, a peak-finding algorithm [2] is first applied to the prediction image for the inner distance transform to locate the centroid of each cell in the image. These predictions are thresholded at a value of 0.1. The interior class of the prediction image for the pixel-wise transform is thresholded at a value of 0.3. The cell centroid locations and interior pixel prediction image are used as inputs to the marker-based watershed algorithm to produce the final label image. The transforms are smoothed with a Gaussian filter to eliminate minor variations and perform additional processing that removes holes and all objects with an area <15 pixels from the final prediction.

4. Dataset

We used LIVECell [1], a dataset of phase-contrast microscopy images collection of over 1.6 million cells for our experiments. There are 5,239 high-quality images that has been manually annotated and expert-validated for the segmentation.

4.1. Cell Images

The images consists of 1,686,352 cells from a diverse morphology status and culture densities, spanning from the small and round, to large and flat and neuronal-like, etc [1]. This diversity ensures the segmentation is applicable to the complete time period for general cell biology experiments. There were eight cell types chosen (A172, BT-474, BV-2, Huh7, MCF7, SH-SY5Y, SkBr3 and SK-OV-3). Most of them are from various human cancerous cell lines except BV-2, which is from mouse. Specifically, A172 was developed from human glioblastoma that grows over to form high densities; BT-474 was developed from human breast cancer and grows in rafts that is challenging to determine the boundaries; BV-2 cells are very homogeneously small spherical shapes; Huh7 and MCF7 cells are very low contrast and have vague boundaries; Because of being human neuroblastoma cell line, SH-SY5Y cells show neuronal morphology characteristics of long protrusions and overlapping; SkBr3 and SK-OV-3 cells all have heterogeneous morphology and some low contrast images. All the cell lines were cultured in 96-well plates and several wells for each cell type were seeded. To cover the various genetics, epigenetics and micro-environmental factors, over the course of 3–5 days, at every 4 hours the images have been taken by Incucyte S3 Live-Cell Analysis system (Sartorius) equipped with its standard CMOS camera. The phase-contrast images were acquired using a 10-time objective from two positions in each well. Each images were further cropped into four equally sized images of 704×520 pixels that were then annotated.

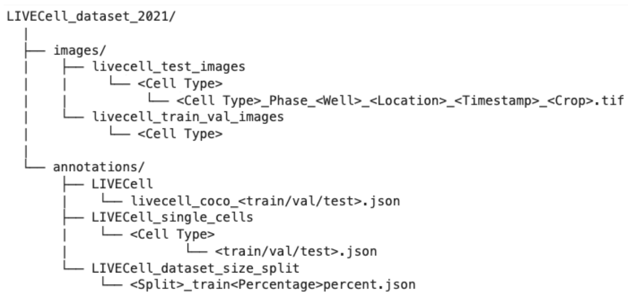
High quality annotations were provided with many precautions according to the dataset provider [1]. First, the images were annotated by a dedicated team of professional annotators. They are trained to do cell segmentation by an experienced cell biologist. Second, the carefully balanced batches of images by the cell types and growth conditions were annotated as batch by batch to minimize and randomize the risk of introducing bias into any single part of the dataset. Last, an annotation manager and an experienced cell biologist will carry out quality assurance process to finalize the annotation.

4.2. Dataset File Structure

To further explore the impact of dataset size on model performance the dataset was divided into subsets corresponding to 2, 4, 5, 25, 50 and 100% of the total dataset size. The subsets were created by taking every nth image from the training dataset $n \in \{50, 25, 20, 4, 2, 1\}$. In addition, the datasets corresponding to individual cell types are also provided. All the LIVECell datasets are stored in an Amazon Web Services (AWS) S3-bucket. After downloading the dataset from `https://sartorius-research.github.io/LIVECell/`, the files are organized in 2 subdirectories named "images" and "annotations". There are 2 subdirectories inside of "images" for "train_val" and "test", whereas 3 subdirectories exist inside of "annotations" for the whole, single cell types and various size-split experiments. The 3 "annotations" subdirectories store the metadata, annotations and mappings to the images as json files. All annotations are in Microsoft COCO Object Detection-format [1]. The detailed file structure is demonstrated in Figure 4.

4.3. Data Processing and Augmentation

In order to be fed into the DeepCell [2] Mesmer model that only takes inputs with shape of $2^n \times 2^n$, the images and annotations are all loaded, parsed, resized and compressed into npz files by a custom utility using skimage, pycocotools and numpy libraries. The semantic transformation and im-



```
LIVECell_dataset_2021/
  |
  ├── images/
  |     ├── livecell_test_images
  |     |     └── <Cell Type>
  |     |           └── <Cell Type>_Phase_<Well>_<Location>_<Timestamp>_<Crop>.tif
  |     └── livecell_train_val_images
  |           └── <Cell Type>
  |
  └── annotations/
        ├── LIVECell
        |    └── livecell_coco_<train/val/test>.json
        ├── LIVECell_single_cells
        |    └── <Cell Type>
        |              └── <train/val/test>.json
        └── LIVECell_dataset_size_split
             └── <Split>_train<Percentage>percent.json
```

Figure 4. The file structure of the downloaded dataset.

age augmentation are fully integrated in the DeepCell API as a data generator (SemanticDataGenerator). It provides the transformations for “inner-distance”, “outer-distance”, “foreground/background” and “pixel-wise”. The training dataset is also augmented by rotation, zooming and flipping only, suggested by the DeepCell publication [2]. All the images are normalized with the batch-wise mean and standard deviation to improve robustness.

5. Results and Discussion

5.1. End-to-End Pipeline

We carefully designed series of experiments to implement, optimize and finalize the training of Mesmer using either small batches or full sized LIVECell datasets. After the processing and transformation LIVECell data are fully compatible with the Deepcell API [22]. An end-to-end pipeline was constructed to train, validate and test the Mesmer models from the perspective of semantic predictive heads, cell types and choices of backbone models. Custom Image Data Generators from Deepcell was used to generate augmented batches of training data. These custom generators extend the Keras ImageDataGenerator class, and allow for training with label masks, bounding boxes, and more customized annotations. They also provide semantic transformation functionality. The Mesmer model, namely “PanopticNet” in Deepcell API, features high-level pyramid network utility functions. It defines the full model structure on top of Keras API [23] and provides flexibility related to backbone model selection, backbone custom structure and semantic heads integration. There are a number of hyperparameters for the “PanopticNet” model training, including the number of epoch, learning rate, optimizer, learning rate scheduler and batch size. After the model training and saving, the instance annotations for the test datasets are generated by the predict method.

5.2. Experiment Evaluation

The evaluation metric used for the experiments is built on the mean average precision. The different intersection-over-union (IoU) thresholds are proposed and a successful segmentation detection was determined by an IoU test (also known as the Jaccard index):

$$IoU(x,y) = \frac{x \cup y}{x \cap y} = \frac{x \cap y}{|x| + |y| - |x \cap y|}$$

which measures the overlap between predicted mask x and the annotated mask y over the union. Using a threshold ranging from 0.5 to 0.95 with steps of 0.05, true positive (TP) instance, false positive (FP) instance and false negative (FN) instance were determined. For a threshold of 0.5, a predicted object is considered a “hit” if the IoU is greater than 0.5. We evaluated our results with an additional metric based on the IoU detection test: F1-score:

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

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

$$F1\ score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

The utility functions for both of the IoUs and F1 scores were provided in Deepcell API [22].

5.3. Semantic Transformation

Previously the cell detection has been modeled as a task of finding most probable pixel as cell centers in an image, which redefined the task to learn an inner distance metric [24]. Same techniques has also been shown to improve the single-shot detection in the state-of-the-art RetinaMask [25]. Inspired by these works, Greenwald et. al. had adjusted these additional sub models to improve the prediction of two distinct transforms: “pixel-wise” transform and “inner-distance” transform [2] (details in Methods). Similarly, “outer-distance” transform and “foreground/background” transform are implemented by Deepcell API [22]. Even though only two semantic transformation heads will be used in the marker-based watershed [26] as a postprocessing step to convert these continuous predictions into discrete label images, all the transformation may be trainable and contribute to the model performance. Therefore, limited combinations of the semantic transformations were tested to optimize the Mesmer training by LIVECell dataset. In order to speed up the search, randomly selected 5% LIVECell datasets were used to train, validate and test the models. The training’s of the models were carried out for only 50 epochs since the loss has reached low enough for searching the optimal semantic transformations. A widely used backbone “ResNet50” was adopted in Mesmer throughout this experiment. As shown in Figure 5, the performances of various semantic transformation combinations, with or without small objects, were plotted as grouped bar chart. The error bars represent the 95% confidence interval for the mean F1 scores of five experiments that were differently seeded in the train, validation and test data splits.

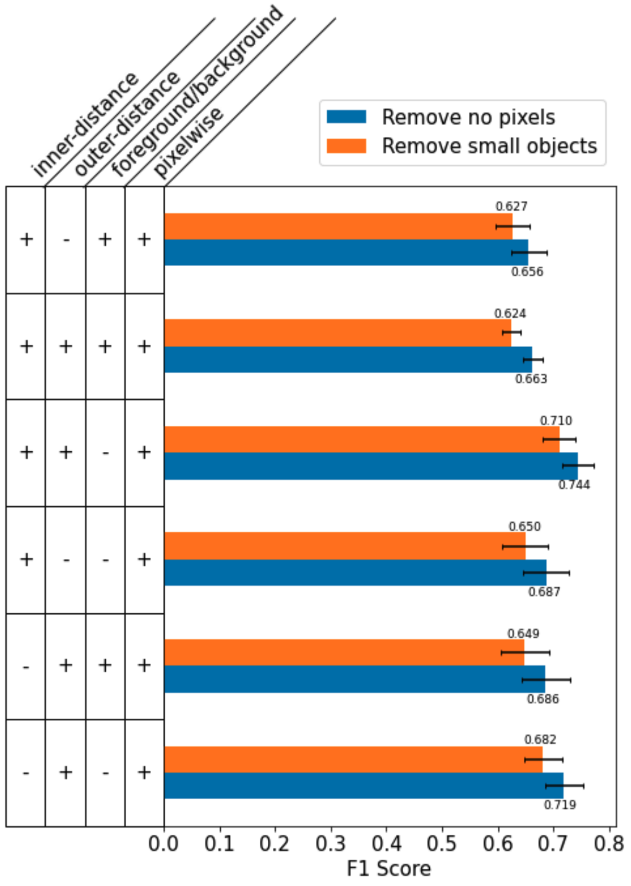


Figure 5. Performances of Various Semantic Transformation Combinations

The result showed that the combination of "inner-distance" with "pixel-wise" alone is significantly better than ones without "inner-distance". This finding was in agreed with the practises in Mesmer trained by TissueNet [2] that confirms the importance of the features captured by distance of each pixel inside a cell to that cell's centroid, and the features that help classify cell interior, cell boundary or background. The result also illustrated that combination of "outer-distance" and / or "foreground/background" produced worse model performances, though not at a level of being statistically significant. The images in LIVECell datasets have only one channel and the cells were not stained, whereas the images in TissueNet dataset have two channels and partially stained. It suggested that using "outer-distance" and / or "foreground/background" would not help Mesmer model training by the transparent cell images like those from LIVECell. It may indeed hurt their performances.

5.4. Cell Types

Latest works on cytoplasmic segmentation continues to experience growing problems that need to be resolved for creatively generic and robust techniques. The diversity of cell morphology is growing rapidly and the demand for segmenting different types of cell images continue to increase in recent years. It is still very difficult to segment various types of cells automatically and robustly with one model [27]. In this experiment, we aim to evaluate the sensitivity of Mesmer model to the training of using the images of each cell types from LIVECell datasets. To keep the data size consistent with Section 5.3, we used 50% of the datasets corresponding to each cell type. The experimental parameters and configurations were all the same as in Section 5.3, except that the semantic transformations were fixed to be "inner-distance" and "pixel-wise".

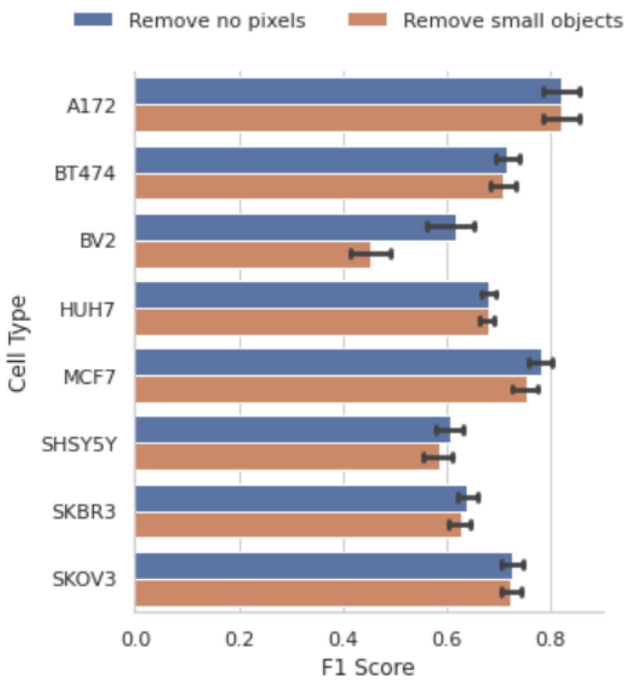


Figure 6. Performances of Mesmer Trained By Individual Cell Type

As show in Figure 6, our results indicated that Mesmer trained by the images of A172 performed the best amongst all the cell types, and the one for BV2 had the lowest F1 score. Given that cell morphology and image characteristics can vary depending on species, organ site and disease origins, a cell type specific model could lead to superior performance when compared to others. These results suggested that it might be helpful to generate more aggressive image augmentations from LIVECell so that Mesmer's generalizability will be increased to cover more diversified cell morphology.

5.5. Backbone Alternatives

According to final conclusion of section 5.3, we only apply inner distance and pixiwise semantic transformation to train the final model and plug pretrained ResNet [18], Densenet [19] and EfficientNet [20] as the backbone models to classify boxing images. The models were trained on Tesla P100 with 50 epochs for each. The optimizer we selected is Adam with learning rate as 5e-4 clipping at 1e-3 and we use 2 images as the one batch. The results of the models' performances are as shown at Table 1. The learning curve of model EfficientNet is shown in Figure 7

Backbone	Average Pixel IoU	F-1	Time for step (ms)
Resnet50	0.4979	0.6647	410
DenseNet201	0.5518	0.7111	420
EfficientNetb7	0.5087	0.6743	470

Table 1. Model results

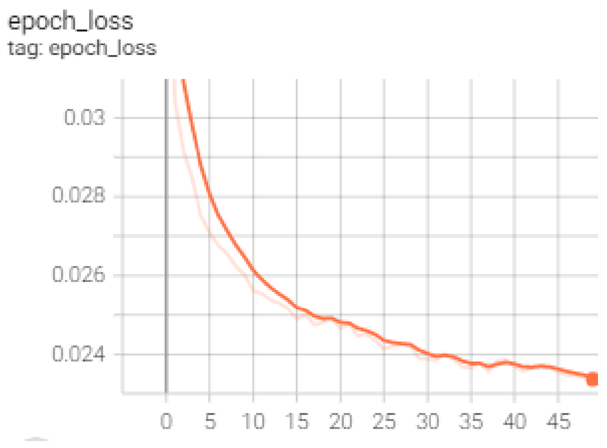


Figure 7. Learning curve for EfficientNetb7

We can see that DenseNet works best as backbone models. It might be due to the fact that the cell images are quite similar to each other and therefore a lot of low-level features can be reused. The DenseNet is designed to strengthen feature propagation and encourage feature reuse which should be suitable in this case. Also, due to time limitation, we only trained with limited epochs (50) and few sets of hyperparameters. But according to the trend of learning curve, we believe the current model is still underfit and able to be improved through more extensive trainings.

6. Conclusion and Future Work

This study used small sample dataset of LIVECell to train and optimize Mesmer, a deep-learning-enabled segmentation algorithm. We found that using the combination of "inner-distance" and "pixel-wise" semantic heads, Mesmer produces optimal cytoplasmic image segmentation. The models trained by each single cell type images show various performances, with the one by A172 being the best amongst them. The extensive search of backbone models for Mesmer confirms that DenseNet201 provides the best result, outperforming the ResNet50 as used in the original Mesmer paper.

The Mesmer model implemented in Deepcell API has the backbone levels specified at $[C3, C4, C5]$ and the pyramid levels specified at $[P3, P4, P5, P6, P7]$, both of them may be adjusted for optimization [22]. In addition, Deepcell API provides a number of tunable hyperparameters, including the number of epoch, learning rate, optimizer, learning rate scheduler and batch size. Future work may involve an extensive search to identify the best combination of the hyperparameters.

Several reviewed segmentation models or systems including nucleAIzer [16], CellPose [28], Stardist [29], EVICAN-MRCNN [30] and FeatureNet [31] showed the promising or State-Of-Art performances. In the future we propose to test how they compare with Mesmer as a benchmark, after being trained and optimized using LIVECell datasets.

7. Contributions and Acknowledgements

All authors contributed to conceiving the project, defining its scope, preprocessing the input dataset, implementing and deploying the models, carrying out the experiments, analyzing the results and writing up the reports.

We acknowledge our project mentor, Professor Andrew Gentles from Stanford Medical School, for his helpful guidance on this work. We also acknowledge our project TA, Ms. Zhuoyi Huang, for her valuable technical assistance and project discussion.

References

[1] C. Edlund, T. R. Jackson, N. Khalid, N. Bevan, T. Dale, A. Dengel, S. Ahmed, J. Trygg, and R. Sjögren, "Livecell—a large-scale dataset for label-free live cell segmentation," *Nature Methods*, vol. 18, pp. 1038–1045, Sep 2021. 1, 4

[2] N. F. Greenwald, G. Miller, E. Moen, A. Kong, A. Kagel, T. Dougherty, C. C. Fullaway, B. J. McIntosh, K. X. Leow, M. S. Schwartz, C. Pavelchek, S. Cui, I. Camplisson, O. Bar-Tal, J. Singh, M. Fong, G. Chaudhry, Z. Abraham, J. Moseley, S. Warshawsky, E. Soon, S. Greenbaum, T. Risom, T. Hollmann, S. C. Bendall, L. Keren, W. Graf, M. Angelo, and D. Van Valen, "Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning," *Nature Biotechnology*, vol. 40, pp. 555–565, Apr 2022. 1, 3, 4, 5, 6

[3] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *Proceedings of the IEEE conference*

on computer vision and pattern recognition, pp. 5221–5229, 2017. 1

[4] K. R. Mosaliganti, R. R. Noche, F. Xiong, I. A. Swinburne, and S. G. Megason, "Acme: automated cell morphology extractor for comprehensive reconstruction of cell membranes," *PLoS computational biology*, vol. 8, no. 12, p. e1002780, 2012. 1

[5] P. Mayorga, J. Valdez, C. Druzgalski, V. Zeljković, and L. Q. López, "Cnn networks to classify cardiopulmonary signals redes cnn en clasificación de señales cardiopulmonares," in *2022 Global Medical Engineering Physics Exchanges/Pan American Health Care Exchanges (GMEPE/PAHCE)*, pp. 1–4, IEEE, 2022. 1

[6] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017. 2

[7] X. Chen, R. Girshick, K. He, and P. Dollár, "Tensormask: A foundation for dense object segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2061–2069, 2019. 2

[8] K. Sofiiuk, O. Barinova, and A. Konushin, "Adaptis: Adaptive instance selection network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7355–7363, 2019. 2

[9] Z. Tian, C. Shen, and H. Chen, "Conditional convolutions for instance segmentation," in *European Conference on Computer Vision*, pp. 282–298, Springer, 2020. 2

[10] J. Yi, P. Wu, Q. Huang, H. Qu, B. Liu, D. J. Hoeppner, and D. N. Metaxas, "Multi-scale cell instance segmentation with keypoint graph based bounding boxes," in *International conference on medical image computing and computer-assisted intervention*, pp. 369–377, Springer, 2019. 2

[11] Z. Zhao, L. Yang, H. Zheng, I. H. Guldner, S. Zhang, and D. Z. Chen, "Deep learning based instance segmentation in 3d biomedical images using weak annotation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 352–360, Springer, 2018. 2

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017. 2

[13] H. Chen, X. Qi, L. Yu, and P.-A. Heng, "Dcan: deep contour-aware networks for accurate gland segmentation," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2487–2496, 2016. 2

[14] T. Falk, D. Mai, R. Bensch, Ö. Çiçek, A. Abdulkadir, Y. Marrakchi, A. Böhm, J. Deubner, Z. Jäckel, K. Seiwald, *et al.*, "U-net: deep learning for cell counting, detection, and morphometry," *Nature methods*, vol. 16, no. 1, pp. 67–70, 2019. 2

[15] D. A. Van Valen, T. Kudo, K. M. Lane, D. N. Macklin, N. T. Quach, M. M. DeFelice, I. Maayan, Y. Tanouchi, E. A. Ashley, and M. W. Covert, "Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments," *PLoS computational biology*, vol. 12, no. 11, p. e1005177, 2016. 2

[16] R. Hollandi, A. Szkalisity, T. Toth, E. Tasnadi, C. Molnar, B. Mathe, I. Grexa, J. Molnar, A. Balind, M. Gorbe, M. Kovacs, E. Migh, A. Goodman, T. Balassa, K. Koos, W. Wang, J. C. Caicedo, N. Bara, F. Kovacs, L. Paavolainen, T. Danka, A. Kriston, A. E. Carpenter, K. Smith, and P. Horvath, "nucleaizer: A parameter-free deep learning framework for nucleus segmentation using image style transfer," *Cell Systems*, vol. 10, no. 5, pp. 453–458.e6, 2020. 2, 7

[17] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2016. 2, 3

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015. 2, 7

[19] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, and K. Weinberger, "Convolutional networks with dense connectivity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 2, 7

[20] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *International Conference on Machine Learning*, 2019. 2, 7

[21] F. Meyer and S. Beucher, "Morphological segmentation," *Journal of Visual Communication and Image Representation*, vol. 1, no. 1, pp. 21–46, 1990. 4

[22] "Deepcell-tf is a deep learning library for single-cell analysis.." `https://deepcell.readthedocs.io/en/master/index.html#`. Accessed: 2022-06-01. 5, 7

[23] "Keras is an open-source software library that provides a python interface for artificial neural networks. keras acts as an interface for the tensorflow library.." `https://keras.io/`. Accessed: 2022-06-01. 5

[24] C. F. Koyuncu, G. N. Gunesli, R. Cetin-Atalay, and C. Gunduz-Demir, "DeepDistance: A multi-task deep regression model for cell detection in inverted microscopy images," *Med Image Anal*, vol. 63, p. 101720, May 2020. 5

[25] C. Fu, M. Shvets, and A. C. Berg, "Retinamask: Learning to predict masks improves state-of-the-art single-shot detection for free," *CoRR*, vol. abs/1901.03353, 2019. 5

[26] M. Weigert, U. Schmidt, R. Haase, K. Sugawara, and G. Myers, "Star-convex polyhedra for 3d object detection and segmentation in microscopy," *CoRR*, vol. abs/1908.03636, 2019. 5

[27] Z. Wang and Z. Wang, "Robust cell segmentation based on gradient detection, gabor filtering and morphological erosion," *Biomedical Signal Processing and Control*, vol. 65, p. 102390, 2021. 6

[28] C. Stringer, T. Wang, M. Michaelos, and M. Pachitariu, "Cellpose: a generalist algorithm for cellular segmentation," *Nature Methods*, vol. 18, pp. 100–106, Jan 2021. 7

[29] M. Weigert, U. Schmidt, R. Haase, K. Sugawara, and G. Myers, "Star-convex polyhedra for 3d object detection and segmentation in microscopy," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3666–3673, 2020. 7

[30] M. Schwendy, R. E. Unger, and S. H. Parekh, "EVICAN-a balanced dataset for algorithm development in cell and nucleus segmentation," *Bioinformatics*, vol. 36, pp. 3863–3870, June 2020. 7

[31] D. A. Van Valen, T. Kudo, K. M. Lane, D. N. Macklin, N. T. Quach, M. M. DeFelice, I. Maayan, Y. Tanouchi, E. A. Ashley, and M. W. Covert, "Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments," *PLOS Computational Biology*, vol. 12, pp. 1–24, 11 2016. 7