# Deep Learning for Cell Segmentation

Grace Kim
yek1354@stanford.edu

Will Song
jsong5@stanford.edu

Lucia Zheng
zlucia@stanford.edu

June 3, 2022

## Abstract

*Recent advancements in imaging techniques and protocols have led to the production of high-throughput quantitative imaging of two-dimensional cell culture. Despite the vast expanse of opportunities in biological and medical studies enabled by the potential of deep learning-based for cellular image segmentation, various challenges have dampened growth in this area. One critical challenge lies in the limited amount of annotated data available. We further expand upon the LIVECell dataset, a newly released, annotated dataset of various cell lines, and investigate performance of U-Net and Mask R-CNN-based models to render semantic and instance segmentation.*

## 1 Introduction

Light microscopy is a cheap and accessible imaging technique to visualize and quantitatively capture cellular features at a large scale. The ability to accurately perform image segmentation on cell microscopy images could enable scientists to study a wide array of biological phenomena and move towards precision medicine treatments. Some patients respond better or worse to different treatments but how patients will respond can only be roughly estimated by a limited set of clinical phenotypes. The pathological characteristics of various tissue samples evaluated for patients may hold a rich repository of information that can further inform knowledge of which treatments are uniquely more suitable for different patients.

Despite the implications accurate cell segmentation holds for a vast array of use cases, performance quality of cell segmentation remains a limiting factor. The challenge of this task has persisted despite the vast improvement deep learning has shown in image segmentation in many other domains largely because of limitations in the size of available datasets. Due to the extensive labor, time, and expertise required to create annotated datasets, existing cellular datasets are magnitudes smaller than those used for image segmentation in other domains.

### 1.1 Related Work

In 2015, U-Net was trained and able to outperform all other models entered in the cell tracking and segmentation challenge of the 2015 IEEE International Symposium on Biomedical Imaging [9]. The U-Net is commonly used for biomedical cell segmentation and is designed for semantic segmentation tasks. It allows two pipelines to capture both context and locality of objects within an image. The motivation for choosing the U-Net is because under light microscopy, the cells on the image have almost no overlap so the projection of segments to a 2D binary mask is sufficient to capture all the needed information. The U-Net produces these binary masks for object segmentation with a high degree of accuracy with low training costs. Hence, the U-Net is perfectly suited for our task. In addition to the U-Net, we explored Mask R-CNN based model structures. Mask R-CNN allows for simultaneous object localization and instance segmentation which makes it prime material ripe for adaption for cell localization and single cell segmentation. Mask R-CNN allows for an opportunity to go beyond just identifying background pixel vs cell to identifying separate individual cells. Cellular instance segmentation, the detection of individual cells, is of major utility that opens the door to whole new world of opportunity. Using the tight bounding boxes generated through Faster R-CNN and clever generation of a binary mask indicating whether the pixel is part of the image, Mask R-CNN is able to both segment and classify objects. These advancements make model structures similar to Mask R-CNN invaluable to further explore in this line of cell segmentation work.

### 1.2 Problem Statement

In fall of 2021, a first-of-its-kind dataset called LIVECell [1] was released. It introduced the largest high-quality resource for label-free segmentation with to advance cell segmentation algorithms. It includes a variety of cell morphologies and high density of cells per image. Baseline models fine-tuned using this data boasted promising results surpassing many other state-of-the-art models for this task. This helped address the existing bottleneck in the advancement of deep learning for cell segmentation. However, even with these advancements, many challenges lie ahead. Cell morphologies with unique or asymmetrical shapes, such as those of the neuroblastoma cell line SH-SY5Y, have proven difficult to segment. Additionally, limited work has been done to examine how model performance changes as over-confluent cell types continue to split into new cells to form increasingly dense cell clusters. With the advent of LIVECell, we seek to examine some of these challenges through experimentation with different CNN-based models and data generation techniques.

## 2 Dataset and Features

The dataset we will be using is LIVECell (Label-free In Vitro image Examples of Cells)[1] , a dataset of 5,239 manually annotated and expert-validated fluorescent microscopy images of 2D cell culture. There are a total of 1,686,352 individual cells that represent eight different cell types: A172, BT-474, BV-4, Huh7, MCF7, SH-SY5Y, SkBr3, and SK-OV-3. All images are grey-scale and each image only contains cells of one type. The images are captured for different cell cultures, seeded at varying densities and grown from early seeding to full confluence. Images of each cell line was taken every 4 hours over a period of 3-5 days as the cell lines grew. The annotations are available in the Microsoft MSCOCO [8] object detection format, where a polygon annotation is provided for each cell object.

## 3 Methods

### 3.1 Dataset Extending

The labor and time intensity of generating annotated data for cell segmentation greatly limits the amount of data available for the task. This in turn poses a limiting factor on task performance. Others, such as the authors of TissueNet [2], another cellular microscopy dataset, have approached this problem by using a semi-automated approach towards generating annotated data. In this approach, models generate initial noisy annotations which are then corrected by human experts. The expert annotated cells are erased and then the next cycle begins with the annotated cells that remain. This cycle continues until all the cells are annotated.

We propose an automated approach. We take an already labeled image with cells, "crop" out the cells using the annotation as a pixel mask, then randomly place cells on a new gray background. The core idea is all cells of the same type may look very similar, but can differ in location and orientation. We perform synthetic generation of cells using existing cells within images to maintain training distribution. We write the code for data generation from scratch.

In Figure 4, we show an example the data augmentation technique applied to vary cell density in an image. Figure 4(a) shows the original image. In Figure 4(b), we generate a sparse and relatively easy image. Figure 4(c) is more akin to the original image with $\sim 100$ fake cells. And Figure 4(d) is a more extreme example of overlapping and clustered cells we can use for hard cell examples.

## 3.2 Models

We choose to explore two primary model architectures in this project, U-Net [9] and CenterMask [6] because we are interested in evaluating model performance on two tasks, semantic segmentation and instance segmentation, on LIVECell. For the semantic segmentation task, we use U-Net to produce a binary mask labeling each pixel as cell or background. For the instance segmentation task, we use CenterMask to produce polygon mask representations to segment each individual cell objects.

Only the instance segmentation task and the CenterMask model is considered in the original LIVECell paper [1] and in general, the instance segmentation task is more challenging, but given hardware constraints, we treat the CenterMask model from [1] as the oracle and explore areas that were underexplored in the original paper: how to train low-latency cell segmentation models under resource constraints, the value-add of model size, architectural complexity, and pretraining for simpler segmentation tasks like semantic segmentation, and the dynamics of single-cell type training, particularly for cell types with unique or irregular shapes.

### 3.2.1 U-Net

We approached U-Net aiming for smaller filters and deeper layers. We implemented a U-Net with only size 3x3 filters and added two additional deeper layers. Additionally, we trained our deepest U-Net with each class forming an ensemble. The traditional U-Net has the same architecture as those described in [9]. We expanded the U-Net by adding an additional Up and Down layer with 1024 channels each at the bottom of the "U" shape. The expanded channels at the bottom of the U-Net should help with identifying images with more cells or difficult geometries.

The expanded ensemble U-Nets use a slightly larger architecture with both an additional Up/Down layer of 1024 channels, and another Up/Down layer of 2048 channels. These filters are extremely small and should work best with classes of small cells that have large counts. We expect to use a combination of single Up/Down expanded U-Nets and double expanded U-Nets, depending on cell count and geometries.

We write our own model classes for the Deep U-Net and the Deep U-Net Ensemble, and use the training harness from PyTorch-UNet GitHub repository[1].

### 3.2.2 CenterMask

For the instance segmentation task, we choose to use a CenterMask architecture. The CenterMask backbone extracts features maps from input images to feed into a Feature Pyramid Network [7] used to help the model more effectively segment objects at these different scales. The scaled feature maps are fed to a fully convolutional one stage object detector (FCOS) [10]. Most instance segmentation models, such as the popular Mask R-CNN [3], rely on pre-defined anchor boxes. In contrast, the FCOS is anchor box free which reduces expensive computations and the number of hyperparameters. FCOS uses a multi-task loss that sums over the loss defined over each of the three tasks: bounding-box classification, classification, and center-ness. CenterMask adapts the FCOS architecture for anchor-free instance segmentation by adding a novel spatial attention-guided mask (SAG-Mask) branch to FCOS. The SAG-Mask predicts segmentation mask inside of the each detected box and uses a Spatial Attention Module (SAM) to attend to more informative pixels. CenterMask is trained with the following loss

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{center} + \mathcal{L}_{box} + \mathcal{L}_{mask} \tag{1}$$

where $\mathcal{L}_{cls}, \mathcal{L}_{center}, \mathcal{L}_{box}$ are as in FCOS [10] and $\mathcal{L}_{mask}$ is average binary cross-entropy loss, as in Mask R-CNN [3].

We choose to use the CenterMask architecture because it has been shown to outperform popular anchor-based architectures like Mask R-CNN [6]. Additionally, the original LIVECell paper [1] finds that an anchor-based architecture fails to detect a greater share of true cell objects than an anchor-free architecture. We use the VoVNet2 [5] backbone because it is more computationally efficient, outperforms traditional backbones like ResNet [4], and the smallest

[1] https://github.com/milesial/Pytorch-UNet

backbone is smaller than the smallest ResNet backbone (MobileNetV2). We leverage transfer learning by using a pretrained VoVNet2 backbone, initializing on weights pre-trained on the MS-COCO 2017 dataset. MS COCO (Microsoft Common Objects in Context) [8] is a dataset consisting of 328,000 images and 1.5 million segmented instances. Transfer learning on MS COCO allows the model to build upon general learnings in object recognition before it is finetuned on LIVECell [1] to learn unique cell morphologies and clustering patterns.

We finetune on LIVECell on a CenterMask architecture with the smallest VoVNet2 backbone with depthwise separable convolution and a thinner FPN with half the channel size, which we refer to as CenterMask V-19-SlimDW. We also evaluate on LIVECell on a CenterMask architecture with the largest VoVNet2 backbone finetuned on LIVECell, released by Edlund et al. [1] from the original LIVECell paper, which we refer to as CenterMask V-99 and treat as the oracle.

We use the detectron2 library [11] to write the finetuning script, adapted from the finetuning script in the CenterMask GitHub repository[2], and write custom model configuration files to adapt the finetuning to our hardware and training budget, which we describe in more detail in Section 4.1. We also write code to convert the predicted instance masks to binary segmentation masks to evaluate the CenterMask models for the semantic segmentation task.

# 4 Experiments and Results

## 4.1 Experimental Details

We use the LIVECell-wide train, validation, and test sets and single-cell type train,validation, and test sets from the original LIVECell dataset [1] for evaluation.

For the U-Net experiments, we train on 1 T4 GPU and use a mini-batch size of 10 and a learning rate between 1e-7 and 1e-8. The U-Nets require a small learning rate because of their depth.

For the CenterMask experiments, we finetuned the CenterMask-V-19-SlimDW model using hyperparameters adapted from those used to finetune the CenterMask-V-99 model in Edlund et al. [1] for our hardware and budget. Edlund et al. [1] finetuned on 8 V100s, with a mini-batch size of 16 and a learning rate of 0.01. Since we finetuned on 4 V100 GPUs, we reduced the mini-batch size by half to 8 and the learning rate by half to 0.005. Additionally, we constrain the finetuning time budget for each CenterMask model to 3 hours by reducing the maximum number of finetuning steps from 100,000 to 12,500 and adapt the learning rate schedule to 1,000 warmup steps, with reductions at 10,500 and 11,500 steps, instead of reductions at 80,000 and 90,000 steps.

## 4.2 Evaluation Metrics

We report F1 Score as our metric for semantic segmentation and average precision (AP) and average false-negative ratio (AFNR) as our metric for instance segmentation, as in Edlund et al. [1]. These metrics are calculated as follows.

F1 scores are widely used to measure classification models and are calculated by taking the harmonic mean of Precision $= \frac{\text{TP}}{\text{TP+FP}}$ and Recall $= \frac{\text{TP}}{\text{TP+FN}}$. Dice coefficient is equivalent to F1 Score. Dice coefficient is a metric commonly used for segmentation tasks to measure the similarity between predicted and ground truth segmentation masks.

$$\text{Dice Coefficient} = \frac{2 * |\text{Pred} \cap \text{Target}|}{|\text{Pred}| + |\text{Target}|} \tag{2}$$

For AP, the degree of overlap between each prediction and its closest ground truth object is quantified using Intersection over Union (IoU), IoU $= \frac{|\text{Pred} \cap \text{Target}|}{|\text{Pred} \cup \text{Target}|}$. If the IoU between the prediction and the closest ground truth object is larger than a certain threshold, the object is considered correctly detected. This gives Precision and Recall at varying IoU thresholds.

The AP at the IoU threshold, $\text{AP}_{\text{IoU}}$, is then given by the area under the curve when plotting the precision against recall for the instance predictions given at the IoU threshold. We report COCO-standard overall AP, which is the average AP over IoU thresholds from 0.5 to 0.95 with a step size of 0.05 is used

[2] https://github.com/youngwanLEE/CenterMask

instead of a single IoU threshold. Note, the COCO-standard overall AP is averaged over all classes, which is traditionally called mean average precision (mAP).

$$\mathrm{AP} = (\mathrm{AP}_{0.5} + \cdots + \mathrm{AP}_{0.95})/10 \tag{3}$$

For FNR, $\mathrm{FNR}_{\mathrm{IoU}} = 1 - \mathrm{Recall}_{\mathrm{IoU}}$ and analogous to average precision, we calculate the average false-negative ratio (AFNR) as:

$$\mathrm{AFNR} = (\mathrm{FNR}_{0.50} + \cdots + \mathrm{FNR}_{0.95})/10 \tag{4}$$

## 4.3 Results

We report results on U-Net with LIVECell-wide training in Table 1, U-Net with single-cell type training in Table 2, CenterMask with LIVECell-wide finetuning in Table 3, and CenterMask with single-cell type finetuning in Table 4.

For the single-cell type training experiments, we select only two cell types to experiment with: (1) SkBr3, the breast cancer cell line, which has regular, square shaped cells, (2) SH-SY5Y, the neuroblastoma cell line, which has long spindly shaped cells. These two cell types were the cell types that the oracle performed the best and worst on respectively, likely due to their morphological attributes.

## 4.4 Discussion

### 4.4.1 Qualitative Analysis

**How can the generated synthetic training data be characterized?** Our data extension methods roughly mimicked increasing confluence levels by adjusting cell density and location. Knowledge of this effect provides critical context for assessing the performance of the different models and cell lines on the original vs extended dataset. This is of especial relevance considering the unique morphologies of each of the different cell lines. In particular, out of the eight cell lines, BV-2, Huh7, and SH-SY5Y are considered to be immortalized cell lines. Unlike typical cell lines which can only proliferate a limited number of times, immortalized cell lines can continue to proliferate indefinitely. And so, such cell lines may experience much higher cell densities than other typical cells. Since all images were obtained from snapshots of early seeding to full confluence, we find that utilizing the data extensions methods to widen the range of densities from very low to very high appropriately captures the full range of growth for such proliferative cell lines. However, for cell lines that reach full confluence at lower density thresholds, there may be

| | U-Net (L) | U-Net (EL) | Deep U-Net (L) | Deep U-Net Ensemble (EL) |
|---|---|---|---|---|
| Split | F1 | F1 | F1 | F1 |
| LIVECell | 0.5873 | 0.2591 | 0.5832 | 0.6052 |
| A172 | 0.4391 | 0.3229 | 0.4326 | 0.4496* |
| BT-474 | 0.7100 | 0.0123 | 0.7127 | 0.6184 |
| BV-2 | 0.7146 | 0.7953 | 0.7266 | 0.7999 |
| Huh7 | 0.6401 | 0.0324 | 0.6358 | 0.6798 |
| MCF7 | 0.6374 | 0.1543 | 0.6293 | 0.6304 |
| SH-SY5Y | 0.5960 | 0.0959 | 0.5825 | 0.6104* |
| SkBr3 | 0.6751 | 0.4360 | 0.6759 | 0.7128 |
| SK-OV-3 | 0.2857 | 0.2245 | 0.2700 | 0.3405* |

Table 1: U-Net (LIVECell-wide training) performance comparison on semantic segmentation task. (L) or (EL) specifies whether the model was trained on the original LIVECell dataset or the extended LIVECell dataset. Split refers to the evaluation test set split. * denotes layer 1 expansion and no * denotes layer 2 expansion.

| | U-Net (L) | U-Net (EL) |
|---|---|---|
| Split | F1 | F1 |
| SH-SY5Y | 0.4676 | 0.6081 |
| SkBr3 | 0.6561 | 0.2070 |

Table 2: UNet (Single-cell type training) performance comparison on semantic segmentation task. (L) or (EL) specifies whether the model was trained on original single-cell type LIVECell dataset or the extended single-cell type LIVECell dataset. Split refers to the evaluation test set split.

| | CenterMask-V-19-SlimDW (L) | | | CenterMask-V-99 (L) | | |
|---|---|---|---|---|---|---|
| Split | F1 | AP | AFNR | F1 | AP | AFNR |
| LIVECell | 0.823 | 34.7 | 56.2 | 0.95 | 48.5 | 44.8 |
| A172 | 0.92 | 20.5 | 61.2 | 0.95 | 39.4 | 49.7 |
| BT-474 | 0.82 | 27.9 | 55.5 | 0.86 | 45.6 | 43.3 |
| BV-2 | 0.813 | 46.7 | 47.1 | 0.86 | 53.3 | 40.2 |
| Huh7 | 0.81 | 31.3 | 45.9 | 0.90 | 54.7 | 33.1 |
| MCF7 | 0.88 | 25.2 | 65.7 | 0.91 | 40.1 | 52.4 |
| SH-SY5Y | 0.80 | 15.9 | 75.2 | 0.84 | 27.7 | 63.2 |
| SkBr3 | 0.9025 | 61.5 | 33.2 | 0.93 | 66.5 | 28.6 |
| SK-OV-3 | 0.91 | 27.7 | 55.8 | 0.94 | 54.4 | 37.5 |

Table 3: CenterMask (LIVECell-wide finetuning) performance comparison on semantic segmentation and instance segmentation tasks. (L) or (EL) specifies whether the model was finetuned on the original LIVECell dataset or the extended LIVECell dataset. Split refers to the evaluation test set split.

| | CenterMask-V-19-SlimDW (L) | | CenterMask-V-19-SlimDW (EL) | | CenterMask-V-99 (L) | |
|---|---|---|---|---|---|---|
| Split | AP | AFNR | AP | AFNR | AP | AFNR |
| SH-SY5Y | 12.0 | 77.4 | 0.02899.4 | | 23.9 | 65.6 |
| SkBr3 | 62.7 | 32.6 | 0.01298.9 | | 65.8 | 29.6 |

Table 4: CenterMask (Single-cell finetuning) performance comparison on semantic segmentation and instance segmentation task. (L) or (EL) specifies whether the model was trained on original single-cell type LIVECell dataset or the extended single-cell type LIVECell dataset. Split refers to the evaluation test set split.

density levels introduced by the data extension technique that are far beyond their natural limits. And thus while our original hypothesis was that extending the dataset would increase performance across the board, especially since data limitations is a major bottleneck for deep-learning in this space, we instead found that performance on the original vs. extended datasets was mixed depending on the particular cell line, as further discussed in latter sections.

To further investigate the critical differences between the different cell lines that affect model performance, we perform principle component analysis (PCA) on the original and extended datasets.

In Figure 1 we see an interesting trend. Despite plotting the data points on the two most principal components, we see that the points are clustered around the origin with limited variance across these components. This effect is particularly apparent for cell lines like Huh7 but not as apparent for cell lines such as BV-2. In Figure 2, we see that we extend the dataset the data points demonstrate a correlated, linear spread across the two principal components. Because the dataset extension introduces locational invariance and increases the density variance, we suspect that the two principal components shown are related to cell location and density.

**For the simpler semantic segmentation task, are there gains from increased model size, greater model complexity, transfer learning? How significant are these gains, when we consider the trade-off in computational efficiency and cost?** We see from the predicted masks in Figure  that the CenterMask model, though trained on a

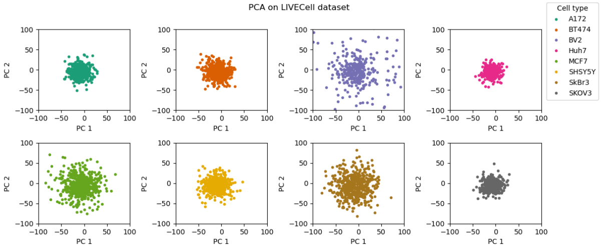


Figure 1: PCA on original dataset

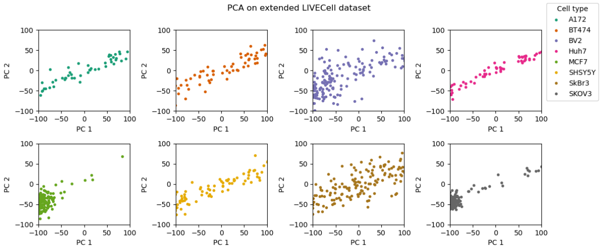
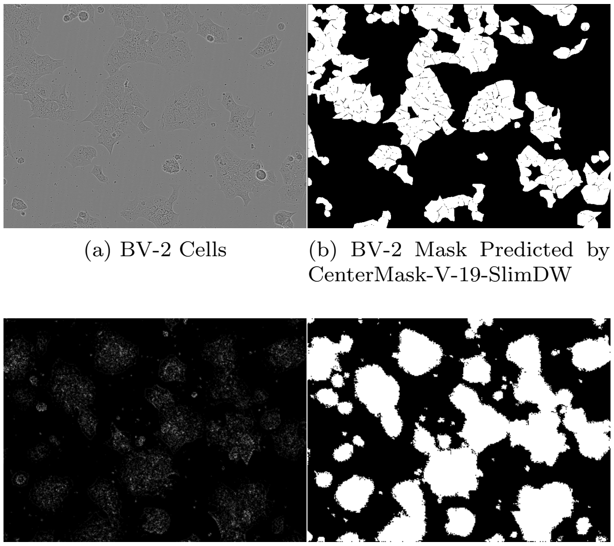


Figure 2: PCA on extended dataset



(a) BV-2 Cells

(b) BV-2 Mask Predicted by CenterMask-V-19-SlimDW

(c) Huh7 Cells

(d) Huh7 Mask Predicted by Deep U-Net

Figure 3: Comparison of Mask Generated by CenterMask-V-19-SlimDW vs. Deep U-Net

more challenging instance segmentation training objective and never trained explicitly for semantic segmentation, produces much higher-quality, crisper binary segmentation masks. This suggests that large pretrained models trained for instance segmentation outperform smaller models trained for semantic segmentation that don't leverage transfer learning, when compared on mask quality for semantic segmentation. This is also supported by the quantitative results, where the F1 scores for even the smaller CenterMask model in column 1 of Table 3 are significantly higher than the F1 scores in column 4 of Table 1, the largest and best performing Deep U-Net Ensemble. Though the CenterMask model takes 3x more GPU-hours to train, the difference in F1 score on LIVECell, 0.23, is considerable (F1 score is between 0 - 1).

#### 4.4.2 Quantitative Analysis

**How does synthetic data generation, increasing model capacity, and ensembling affect performance on semantic segmentation and instance segmentation tasks on LIVECell?** We see from Table 1 that the U-Net trained on the extended LIVECell dataset has much lower LIVECell F1 of 0.2591, compare to the U-Net trained on the original LIVECell dataset, which has a LIVECell F1 of 0.5873. For both U-Net and CenterMask, it appears that training on the extended LIVECell dataset only improves performance for the U-Net model trained and evaluated on the single-cell type SH-SY5Y, in Table 2, where F1 increases from 0.4676 to 0.6081. There are two likely explanations for why the synthetic training data doesn't help much. First, we think the method to generate data may have produced data that was too dissimilar or out of distribution compared to the original data, since we varied cell density in the generated images to the same upper threshold for all cell types, but this is perhaps unrealistic, since some cell types have greater density at full confluence than others. This explanation also explains why we saw benefit from the extended dataset for SH-SY5Y, given its proliferative nature. The PCA of the original and synthetic data suggests that this could be the case, since the original data appears to have less variance along the first principal component compared to the synthetic data, which could be characterizing cell density. Second, the U-Net model we train with the additional synthetic data on is small compared to all of the other models, so perhaps it lacked the capacity to learn more complex patterns from more training data, which was likely also compounded by the high cell density of the synthetic training images.

We see from column 3 and 4 in Table 1 that increasing the depth of the U-Net and ensembling U-Nets improves performance on the full LIVECell dataset and all single-cell type splits, with at least one of the Deep U-Net or the Deep U-Net Ensemble achieving higher performance than the shallow U-Net. We see a similar relationship between model capacity and performance for CenterMask, with higher AP in column 5 of Table 3 than in column 2 of Table 1 for evaluation on LIVECell and all single-cell type splits.

**Are there cases where a lighter weight pretrained backbone competes with a a larger pretrained backbone for CenterMask?** For CenterMask, we see that the performance regresses more with the smaller model with the lighter weight pretrained backbone for cell types with protruding or irregular shapes that are difficult to segment as a result, such as SH-SY5Y (AP difference is -11.8) and A172 (AP difference is -19.4), compared to cell types with more regular, round shapes, such as SkBr3 (AP difference is -5) and BV-2 (AP difference is -6.6), but across the board, the larger model with the larger pretrained backbone outperforms the smaller model wit h the lighter weight pretrained backbone for all cell types.

**Are there cell types for which models trained on single-cell types outperform models trained on all cell types?** We observe that the small CenterMask finetuned on the single-cell type SkBr3 outperforms the small CenterMask finetuned on the full LIVECell dataset when evaluated on the single-cell type SkBr3, with AP of 62.7 compared to AP of 61.5. This relationship is inverted for the large CenterMask model. This suggests that a small CenterMask model may lack the capacity to learn features that generalize well across all cell types and in particular, for regular cell types that may be similar in shape to other common objects that may have been seen during pretraining, it may be beneficial to finetune a cell-specific model.

**Is the performance of models trained on single-cell types bottle-necked by shared representations that generalize across cell types or the amount / diversity of the training data?** From the U-Net results, we see that for the SH-SY5Y cell type, the difference between the F1 score for the shallow U-Net trained on the full LIVECell dataset versus only SH-SY5Y split is -0.1284, while the difference between the F1 score for the shallow U-Net trained on the full LIVECell dataset versus the extended SH-SY5Y split with synthetic data is 0.0121. This suggests that for cell types with irregular shape, we can do better by increasing the amount / diversity of training data, whereas the LIVECell paper showed that the single-cell type training always does worse due to the model losing out on learning shared representations that generalize across cell types.

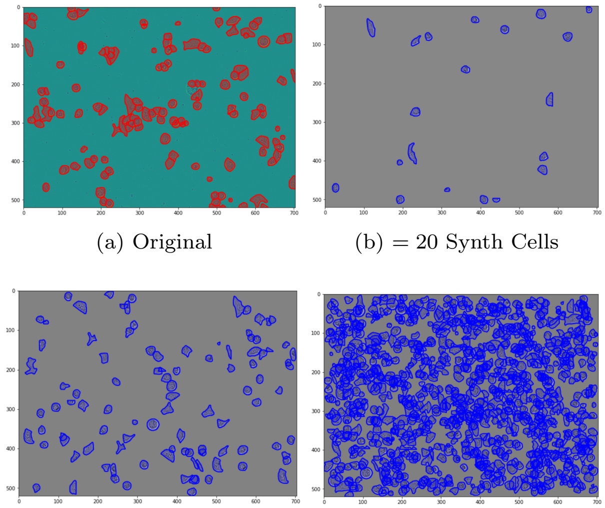
## 5 Conclusion/Future Work

We see applications of CenterMask and U-Net work very well to segment individual cells in cell microscopy, especially transfer learning from other tasks. Our Naive U-Net performed reasonably well, but CenterMask performs very well on cell segmentation. This allows researchers to automate the extremely labor intensive human segmentation task.

A potential future application of models to the LIVECell dataset involves characterizing time as a latent parameter to the evolving cell geometries across phases. Namely, the cells grow with a time dynamic and have geometric locality in phases. Hence, two future directions are discretizing along the time parameter and using Ensemble U-Net for each phase. Additionally, another direction is potentially using a time-series model like a Transformer or VAE to directly inject time as a latent variable.

Another potential future direction is coupling data augmentation with curriculum learning. Since we can vary the density and geometry of cells within synthetic datasets, we can vary the difficulty at every stage of learning within the dataset.

# 6 Appendix



(a) Original

(b) = 20 Synth Cells

(c) = 100 Synth Cells

(d) = 1000 Synth Cells

Figure 4: Synthetic images for number of cells 20-1000 from a single image

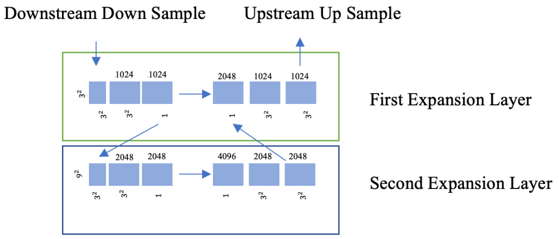


Figure 5: Additional two layers added to the bottom of the U-Net.

# 7 Contributions

Links to GitHub repositories used or referenced

- https://github.com/milesial/Pytorch-UNet
- https://github.com/sartorius-research/LIVECell
- https://github.com/youngwanLEE/centermask2
- https://github.com/facebookresearch/detectron2

WS generated the extended dataset, ran Deep U-Net and Deep U-Net Ensemble experiments, and transformed predicted instance masks to evaluate CenterMask for semantic segmentation. LZ ran CenterMask-V-19-SlimDW and CenterMask-V-99 experiments and generated the PCA visualizations. GK ran U-Net on the extended dataset, U-Net single-cell type experiments, and ran evaluation for CenterMask-V-99.

# References

[1] Christoffer Edlund, Timothy R Jackson, Nabeel Khalid, Nicola Bevan, Timothy Dale, Andreas Dengel, Sheraz Ahmed, Johan Trygg, and Rickard Sjögren. Livecell—a large-scale dataset for label-free live cell segmentation. *Nature methods*, 18(9):1038–1045, 2021. 2, 3, 4

[2] Noah F Greenwald, Geneva Miller, Erick Moen, Alex Kong, Adam Kagel, Thomas Dougherty, Christine Camacho Fullaway, Brianna J McIntosh, Ke Xuan Leow, Morgan Sarah Schwartz, et al. Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning. *Nature biotechnology*, pages 1–11, 2021. 2

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 3

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[5] Youngwan Lee, Joong-won Hwang, Sangrok Lee, Yuseok Bae, and Jongyoul Park. An energy and gpu-computation efficient backbone network for real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019. 3

[6] Youngwan Lee and Jongyoul Park. Centermask: Real-time anchor-free instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13906–13915, 2020. 3

[7] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 3

[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2, 4

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1, 3

[10] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019. 3

[11] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. 4