Neuronal Cell Instance Segmentation

CS231n Spring 2022 Project Final Report **Project category: Computer Vision**

Keyu Nie
keyunie@stanford.edu

Stefan Zdrnja
szdrnja@stanford.edu

Swathi Gangaraju
swathig1@stanford.edu

Abstract

Goal of our project is to predict neuronal cell instances in microscopic images in Kaggle dataset created from LiveCell dataset. We applied a two-stage model structure to identify model(s) that performs well on cell instance segmentation tasks. Based on our experiments and comprehensive research, YOLOX as detector and Swin Transformer within UperNet architecture as segmentor, yielded best results for this problem validating the approach that won Kaggle competition for this task. We reported our findings in hyper parameter tuning, model comparisons, ablation study on the best model, WBF ensemble method improvement and visualization of prediction from the best model in our experiment results. Our implementation is based on MMCV [3], an open source Computer Vision Foundation toolbox based on PyTorch from OpenMMLab.

1. Introduction

Neurological disorders, such as Alzheimer's disease and brain tumors, are a primary cause of death and disability worldwide. Unfortunately, determining how effectively these deadly disorders respond to new medicine treatment is complex. Accurate segmentation of neuronal cells in microscopic images provides some leverage in this task. However, the limitation of low contrast and high object density can require specialized image segmentation mechanism. Current instance segmentation models demonstrate poor accuracy for neuronal cells. According to a recent Nature study [8], the neuroblastoma cell line SHSY5Y consistently has the lowest precision scores out of the eight cancer cell types researched in that study. The difficulty in recognizing these types of cells could be because neuronal cells have a distinct, irregular, and concave appearance, making them problematic to segment with commonly used mask heads. Models devised for this task need to be robust to visual variations caused by cell mitosis, cell distortion, cell adhesion, unclear cell contours, and background impurities. Furthermore, the tiny and slender structures such as filopodia and lamellipodia involved in cell movement render the problem even more difficult.

Goal of our project is to apply Computer Vision algorithms to detect single neuronal cells in microscopy images. The input to our algorithm is image and related metadata including image size. Output is image with masks for each instance on the image.

2. Related Work

Our task is a typical instance segmentation task which can be formulated as a classification problem of pixels with instance labels. It addresses the tasks of object detection and instance segmentation simultaneously. Usually, this involves dealing with multiple overlapping objects, complex backgrounds, and requires different outputs for the same class in distinct locations. The instance segmentation problem is the most challenging of the four basic tasks of computer vision [29].

One of the early algorithms that addressed the object detection task was Region-based convolutional network (R-CNN) [11]. This algorithm implemented selective search to extract region of interests (ROI) from the image in order to achieve accurate object detection and classification. Later, Fast R-CNN [10] and Faster R-CNN [24] architectures were introduced to speed up processing and improve efficiency by introducing ROI-pooling and Region Proposal Network (RPN) with anchors. However, problems of information loss for small objects in feature space and feature extraction remained. To address this, a top-down architecture Feature Pyramid Network (FPN) with lateral connections was developed to build high-level semantic feature maps at all scales. Besides these two-stage anchor-based object detection methods, many state-of-the-art one-stage detectors such as YOLO series [1, 15, 21–23], SSD [18], and RetinaNet [16] were developed. YOLOv1 [21] and SSD [18] utilize similar ideas but with different networks to directly predict object classification and bounding-box regression without using region proposals (like in RPN). RetinaNet [16] enhanced the classification loss with focal loss function to address class imbalance during training. Almost all of the above detectors relied on pre-defined anchor boxes. A fully convolutional one-stage object detector

(FCOS [30]) was developed to solve per-pixel without anchors, analogue to semantic segmentation. YOLOX [9], the latest version of YOLO series, utilizes the same backbone (Darknet53) as YOLOv3 [23] with an anchor-free detector and applies a decoupled head and SimOTA [7] to improve convergence speed.

For the object segmentation task, it was worth noting that the fully convolutional network (FCN) in 2015 [20] achieved end-to-end pixel-wise semantic segmentation. In the same year, an extension of FCN called UNet [25] was developed where successive deconvolutional layers with skip-connections were employed to produce more precise output. UNet outperformed other approaches in biomedical segmentation tasks.

Mask R-CNN [12] was presented in 2017 and it added a mask prediction branch to the Faster R-CNN network to support both object detection and segmentation tasks. To fix the misalignment of bounding boxes (BBOX) with different sizes, Mask R-CNN proposed a simple quantization-free layer called RoI Alignment (RoIAlign) that preserves exact spatial locations.

After Mask R-CNN, researchers realized that we can simultaneously reuse the same feature space (output from backbone with neck) for many downstream tasks, mimicking the multi-tiered perception observed in humans. UperNet [32] was one of these multi-task frameworks under encoder decoder fashion, which can be applied for feature extraction to capture visual hierarchical context in the image.

Due to recent success of Transformers [31] in language models, Vision Transformers (ViT) [6] was the attempt to apply the same principles to Computer Vision by dividing the input images into small patches. The Swin Transformer (SWIN-T) [19] applied a hierarchical Transformer whose representation is computed with shifted windows. The shifted windowing scheme brings greater efficiency by limiting self-attention computation to non-overlapping local windows while also allowing for cross-window connection. Swin Transformer [19] (unlike ViT [6]) applies self-attention at the layer level. This seems more promising than the ViT [6] approach in dealing with high resolution and varying scales of input images. It was quickly recognized that Transformers like Swin Transformer (SWIN-T) [19] can be used to replace the backbones in both object detection and segmentation models.

Our implementation is based on MMCV [3], an open source Computer Vision Foundation toolbox based on PyTorch from OpenMMLab. The toolbox provides a framework within which model architectures can be devised with ease. This, alongside the availability of pre-trained weights, considerably sped up design and implementation stages of model development. We explored different criteria to select the correct region proposals and combine bounding box predictions. Non-Maximum Suppression (NMS) [14] and Weighted Box Fusion(WBF) [27] are popular approaches in this regard.

3. Methods

The neuronal cells addressed in this report are small and irregular in size, and more often than not, their populations are dense. The segmentation output needs to recognize these cells and factor in their tiny structures like filopodia and lamellipodia. Due to these complications, the main approach described in this report implements two-staged segmentation. The following framework was used to search for the best models (Fig. 1).

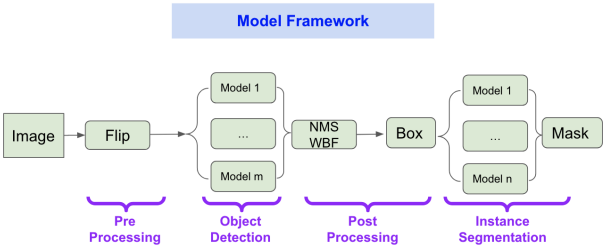
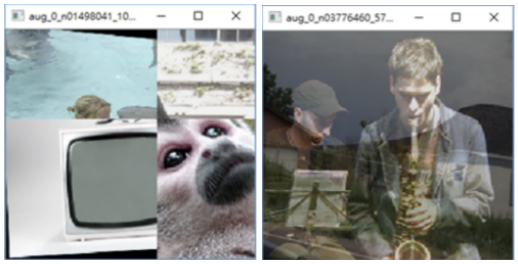


Figure 1. The framework of our models.

This model framework includes four parts: Pre-processing, Object Detection, Post-processing, and Instance Segmentation.

Pre-processing: During pre-processing a sequence of general data augmentation techniques was applied to the input image. These included color jitter, flipping the data and annotations in the vertical and horizontal direction, resizing the image with paddings, normalizing the image, photometric distortion [26] to change the brightness, contrast, and others. Mosaic (see in Fig. 2a) and MixUp (see in Fig. 2b) introduced in YOLOv4 [1] were applied to merge two or more images into one image for the detection task.



(a) Mosaic (b) MixUp

Figure 2. Data Argumentation examples.
Source: YOLOv4: Optimal Speed and Accuracy of Object Detection [1].

Object Detection: Various object detection models were researched for the bounding box (BBOX) prediction

task. Furthermore, model ensembling was used to enhance BBOX predictions. We believed that by improving the precision of BBOX, we would improve the instance segmentation in the second stage. After a careful consideration of many different frameworks, MMDetection [2] and MMSegmentation [4] packages with pretrained weights were utilized to solve the detection and segmentation tasks. According to the MMDetection framework, object detection model will have backbone, neck, RPN head, and ROI head parts. Model utilizes backbone and neck to extract features from images and relies on RPN and ROI (two-stage) for final BBOX predictions. The explored models are the following:

- **Faster/MASK R-CNN**: Faster/MASK R-CNN is a widely adopted object detection method. MASK R-CNN shares the same structure as Faster R-CNN for its object detector. The slight difference in MASK R-CNN, compared to our model framework, is that MASK R-CNN does not take the output of BBOX prediction from Faster R-CNN as its input for the mask prediction. Instead, MASK R-CNN extracts features from the output of the shared RPN network structure. The default Faster R-CNN model relies on ResNet-50 (or ResNet-101 for enhancement) [13] as its backbone connected with FPN as its neck for feature extraction. We studied the model performance with Swin Transformer as its backbone, and noted it as SWIN-T(MASK R-CNN).

- **RetinaNet**: RetinaNet is one of the best one-stage object detection models that reliably works well with dense objects. RetinaNet utilizes focal loss (FL) (equation 1) to address the class imbalance encountered during training of dense detectors.

$$FL(p_t) = -(1-p_t)^{\gamma}\log(p_t) \tag{1}$$

  Here p_t is the model probability prediction for the truth class t and γ is the focusing parameter tuned to balance between classes. The RetinaNet model shares similar feature learning structure as Fast R-CNN and its backbone is based on ResNet-50/ResNet-101 [13] as well. We also ran alternative Retinanet detector with Swin Transformer as backbone, and noted it as SWIN-T(RetinaNet).

- **YOLOX**: YOLOX is the state-of-the-art object detector on COCO dataset before transformers were introduced in Computer Vision. Input to YOLOX are images scaled to default size of 640×640. YOLOX leverages YOLOv3 as the backbone architecture and differs in having decoupled head. As shown in 3, for each level of FPN features a 1×1 conv layer is used to reduce the feature channel to 256. Afterwards, two parallel branches with two 3×3 conv layers are used for classification and regression. IOU branch is added on the regression branch. Output of the model are bounding box predictions and cell type classifications.

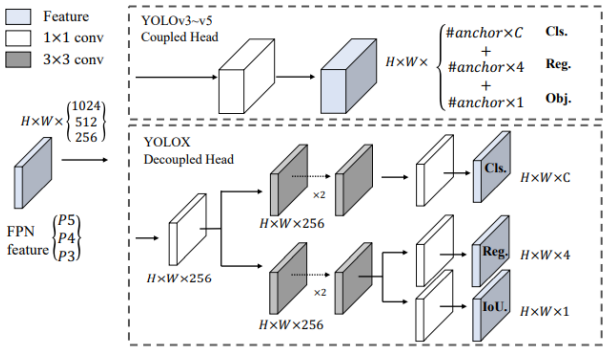


Figure 3. Comparison of YOLOX with YOLOv3
Source: YOLOX: Exceeding yolo series [9]

Post-processing: After the bounding box predictions, we filter out the overlapping bounding boxes of the same instance using post processing techniques like NMS and Weighted Boxes Fusion (WBF [28]). NMS technique includes selecting the bounding box with highest confidence score and removing rest of them if IOU is greater than the threshold. On the other hand, WBF uses all boxes to create a weighted box. The difference could be visualized in Fig. 4.

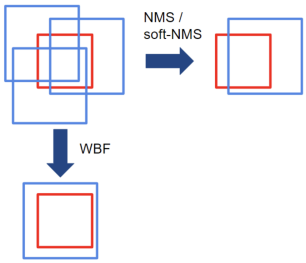


Figure 4. Schematic illustration of NMS/soft-NMS vs.WBF outcomes for an ensemble of inaccurate predictions.Blue – different models' predictions, red – ground truth.
Source: Weighted boxes fusion [28]

Similar data transformations as in pre-processing were applied before feeding our predicted bounding box into the instance segmentor. To make sure that the segmentor has the capability to deal with different sizes of BBOX and focus on one predicted BBOX at each time we added RoI Alignment with fixed size at the beginning of data argumentation.

Instance Segmentation: We are using a custom encoder-decoder architecture to perform task of instance segmentation. Following MMSegmentation [4] structure, the encoder-decoder model will have backbone, neck, decoder head and auxiliary head parts. Model encodes images with backbone and decodes into a semantic segmentation map of the same size as input. We may also apply simple average to ensemble multiple instance segmentor outputs at the end.

- **UperNet-SWIN-T**: UperNet is one of the best encoder-decoder models for multiple down streaming tasks including instance segmentation. As shown in Fig. 5, it applies a model based on ResNet as its encoder and UperHead as its decoder. Decoder processes dense locations of the features extracted from encoder to extract visual knowledge in the images. Upernet utilizs FPN structure as well. In UperNet-SWIN-T, we are using SWIN-Transformer (SWIN-T) as backbone encoder to replace the default ResNet-like backbone in UperNet, but keep the UperHeaD as Decoder Head to finetune and Fully Convolutional Network as Auxillary Head (UperNet-SWIN-T). SWIN-T (Fig. 6) is a powerful model to build hierarchical feature maps by merging image patches into deeper layers. It results in computational complexity proportional to the input image size due to self-attention processing occurring only within each local window.

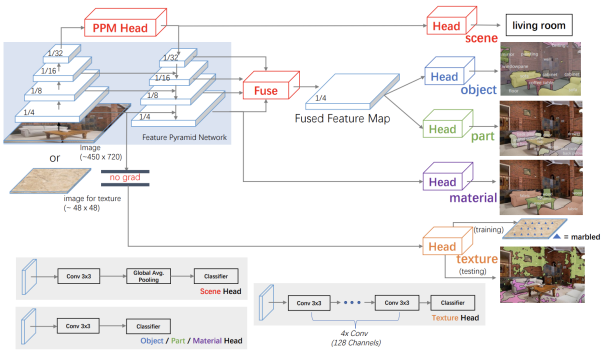


Figure 5. The Encoder-Decoder structure in UperNet.
Source:Unified Perceptual Parsing for Scene Understanding, [32]

We leveraged the suggested structure by Open MMLab-MMDetection and MMSegmentation toolkit to organize our data and code structure. This involved creating config files for each model and understanding the various features and config assumptions, available model architectures and pretrained models to leverage toolkit correctly to perform well on our task. We also developed necessary code for preprocessing, post-processing (WBF) and visualization of results. The YOLOX and UperNet-SWIN-T are also initialized based on the first place solution github [1] in this Kaggle Competition.

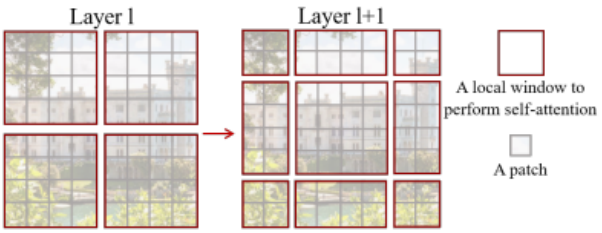


Figure 6. Illustration of the shifted window approach for computing self-attention in the proposed Swin Transformer architecture Transformer
Source:Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, [19]

4. Dataset

For this project we are using Kaggle competition data for Saratorius-Cell Instance Segmentation [2]. The Kaggle dataset contains 606 images ($\approx$ $140M$) for 3 cell types with 73585 annotations in training set and 240 images in testing set under COCO [17] format 704×520. There is a similar dataset with larger sizes ($\approx$ $1.3G$) for 8 cell types coming from LiveCell Dataset [8]. Typically, an image can contain multiple instances of a cell type. Each image is associated with a unique cell type. The dataset includes following metadata about the image: image ID, annotations, width, height, cell type, plate time, sample date and elapsed time delta. Fig. 7 provides five examples of metadata in the dataset. There are three cell types in the dataset: 10777 CORT (neurons), 52286 SHSY5Y (neuroblastoma), and 10522 ASTRO (astrocytes). We split up the dataset for training, validation and testing in a $70 - 15 - 15$ ratio, and report the model performance in test dataset.

As shown in (Fig 8): CORT type is the smallest cell type in dataset, the average size of box is $(17, 17)$. SHSY5Y is bit bigger than CORT type, average size of SHSY5Y box is $(18, 18)$. ASTRO is the largest cell type in dataset. Shape of ASTRO type is quite unbalanced, this type has smallest and biggest shape over entire instances. Shape of ASTRO goes from 2 pixel to over 300 pixel, but box's size tend to centralize in range $[0, 150]$

[1] https://github.com/tascj/kaggle-sartorius-cell-instance-segmentation-solution
[2] https://www.kaggle.com/competitions/sartorius-cell-instance-segmentation

	id	annotation	width	height	cell_type	plate_time	sample_date	sample_id	elapsed_timedelta
0	0030fd0e6378	118145 6 118849 7 119553 8 120257 8 120961 9 1...	704	520	shsy5y	11h30m00s	2019-06-16	shsy5y[diff]_E10-4_Vessel-714_Ph_3	0 days 11:30:00
1	0030fd0e6378	189036 1 189739 3 190441 6 191144 7 191848 8 1...	704	520	shsy5y	11h30m00s	2019-06-16	shsy5y[diff]_E10-4_Vessel-714_Ph_3	0 days 11:30:00
2	0030fd0e6378	173567 3 174270 5 174974 5 175678 6 176382 7 1...	704	520	shsy5y	11h30m00s	2019-06-16	shsy5y[diff]_E10-4_Vessel-714_Ph_3	0 days 11:30:00
3	0030fd0e6378	196723 4 197427 6 198130 7 198834 8 199538 8 2...	704	520	shsy5y	11h30m00s	2019-06-16	shsy5y[diff]_E10-4_Vessel-714_Ph_3	0 days 11:30:00
4	0030fd0e6378	167818 3 168522 5 169225 7 169928 8 170632 9 1...	704	520	shsy5y	11h30m00s	2019-06-16	shsy5y[diff]_E10-4_Vessel-714_Ph_3	0 days 11:30:00

Figure 7. Metadata of 5 example instances

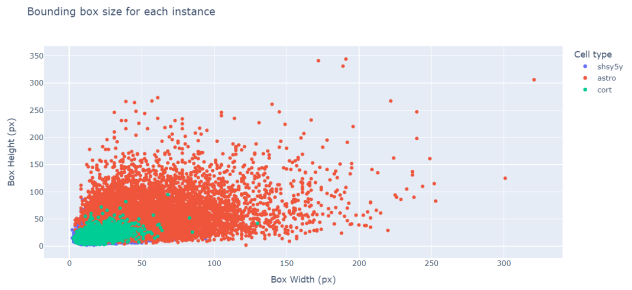


Figure 8. Bounding Box Size in three cell types.

Fig. 9 gives the distribution of the number of cell instances per image for each cell type. X-axis shows the number of instances and Y-axis represents number of images. Most images have less than 200 instances. Most CORT images have between 15 and 50 instances. Number of instances per SHSY5Y images is the largest where 80% SHSY5Y images have between 50 and 450 instances.

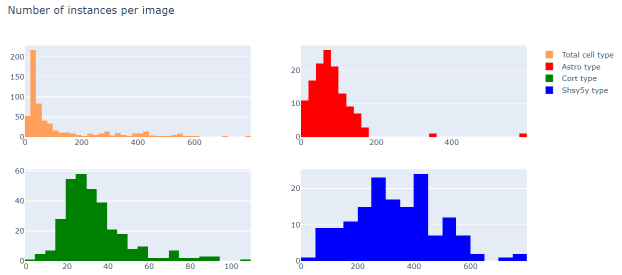


Figure 9. Number of Instances per Image

4.1. Evaluation Metric

We evaluate our results by the mean average precision (mAP) at different intersection over union (IoU) thresholds.The IOU between the prediction and the label per object is calculated as

$$IoU(A, B) = \frac{A \cap B}{A \cup B},$$

where A is the proposed set of object pixels and B is the set of true object pixels.

We define a threshold (t) for the IoU. If the IOU of an object between prediction and truth is higher than the threshold, the prediction is considered correct. At each threshold value (t), the precision is defined based on the number of true positives (TP), false negatives (FN), and false positives (FP)

$$precision = \frac{TP(t)}{TP(t) + FP(t) + FN(t)}.$$

The precision value sweeps over a range of IOU thresholds (t). The average precision (AP) of a single image is then calculated as the mean of the above precision values at the range of IOU thresholds from 0.5 to 0.95 with a step size 0.05 $(IOU0.5 : 0.05 : 0.95)$: $(0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95)$:

$$AP = \frac{1}{\#\ thresholds} \sum_t \frac{TP(t)}{TP(t) + FP(t) + FN(t)}$$

The overall score of the model is defined as the mean Average Precision (mAP) which is computed as the average AP over all images under 3 classes.

5. Experiments

5.1. Model Training and Experiment Settings

In our training, we use cross entropy loss for classification in most of object detectors, except in Retinanet, we use focal loss (Eq. 1). The loss function for BBOX is L1 norm.The loss function for segmentation is cross entropy.

In our experiments, each detector and segmentor was trained for 30 epochs. GPU was set with half floating point precision (fp16) and loss scale factor 512. If not specially mentioned, ResNet50 was the default backbone and load with pretrained weights from ImageNet [5]. All models loaded the weights pretrained on COCO dataset ([17]) and with options to continue pretraining on LiveCell dataset. AdamW optimizer was used with learning rate 0.000078125 (0.0000125 for segmentors) with $\beta = (0.9, 0.999)$ plus the weight decay 0.0005. We also added a linear learning rate scheduler with warm up 500 steps, linear/Exponential decay and minimal learning rate ratio 0.1. Regarding the data augmentation in both pre-processing and post-processing period, we use one of the followings:

- Data augmentation process for Faster/Mask-RCNN and RetinaNet: RandomFlip with ratio=0.5, resize to multi-scale but keep the ratio, normalization, padding.
- Data augmentation process for YOLOX: Mosaic, MixUp, RandomFlip with ratio=0.5, Photometric Distortion, resize to multi-scale but keep the ratio, normalization, padding.
- Data augmentation process for UperNet-SwinT: Box Jitter, ROI Alignment with crop size 128×128, Random flip with ratio=0.5, resize to multi-scale but keep the ratio, normalization.

Model	Resize	BackBone	Optimizer	batch	Pretrain on LiveCell	mAP	AP		
							SHSY5Y	ASTRO	CORT
Faster R-CNN	(1333, 1333) (1280, 1280) (1024, 1024)	ResNet50	AdamW	2	YES	0.237	0.084	0.229	0.397
Faster R-CNN	(1333, 1333) (1280, 1280) (1024, 1024)	ResNet50	AdamW	2	NO	0.244	0.085	0.253	0.393
Faster R-CNN	(640, 640)	ResNet50	AdamW	4	NO	0.184	0.046	0.228	0.28
Faster R-CNN	(640, 640)	ResNet50	$Momentum^*$	2	NO	0.185	0.051	0.225	0.283
RetinaNet	(1333,640) (1333,800)	ResNet	$Momentum^*$	2	NO	0.216	0.076	0.202	0.372
SWIN-T(RetinaNet)	(1333, 1333) (1280, 1280) (1024, 1024)	SWIN-T	$Momentum^*$	2	NO	0.068	0.011	0.011	0.183
YOLOX	(640,640)	CSPDarknet	AdamW	4	NO	0.286	0.064	0.003	0.138
YOLOX	(1536, 1536)	CSPDarknet	AdamW	4	NO	0.334	0.294	0.296	0.41

Table 1. Object detection model performance under different hyperparameter settings. The Momentum optimizer utilize lr=0.00015625, Momentum=0.9, and weight decay=0.005. If the resize shape contains multiple choices, the actual output will random pick one with equal probability.

We ran into memory allocation issues whenever we ran with more than 2 samples per GPU. Sometimes even had issue with 1 sample as well. So the batch size is set to 2 for most models and that is changed to 1 when pre-training models on LiveCell data. We also ran into issues when scaling up image size which we decided to do for model to be able to pick up on small differences. These challenges limited our model exploring space.

We reported the mAP (IOU $0.5 : 0.05 : 0.95$) and AP (IOU $0 : 0.1 : 1$) in three cell types for each model training. In the reports of model performance using WBF, AP (IOU $0 : 0.1 : 1$) in different instance sizes (small objects: area $< 32^2$, medium objects: $32^2 <$ area $< 96^2$ and large objects: area $> 96^2$) are also recorded.

5.2. Hyper-parameter Tuning

To start with, we first reported how we ended up with certain hyper-parameters in our model training.We believe that higher accuracy of BBOX prediction in object detection will also help the second stage instance segmentation tasks. So we tried to resize input image into different scale, different batch size, AdamW vs Momentum with weight decay and with/o pretraining on LiveCell dataset. The performance of several trials could be found in Table 1.

We had the following findings:

1. Generally AdamW optimizer served better results than Momentum optimizer with weights decay.

2. A bigger resize shape would improve model performance. We credited this to the extremely small shape of the neuronal cells, which might be ignored in current image datasets with labels.

3. Pretrain our model on LiveCell dataset would increase our model performance as well. Faster R-CNN with pretrain (on LiveCell dataset) increased the AP in ASTRO 3% (0.229 vs 0.253). We also speculated the low performance of Swin-T(Retinanet) was due to the limited image sizes in our Kaggle data, as Transformers-alike usually required large models with big image sizes. Here Swin-T(RetinaNet) stands for a RetinaNet strucutre but replace the backbone with Swin-T. We were not able to pretrain Swin-T(RetinaNet) on LiveCell dataset even in V100-16G due to out of memory error.

We also noticed that the SHSY5Y cell type is very sensitive to IOU thresholds. This explained the low performance of models like in Faster R-CNN had very low AP (0.046) in SHSY5Y but with much higher mAP (0.184) given that SHSY5Y was the dominant cell types ($> 70\%$). Faster R-CNN generally did OK, even on the Kaggle dataset only. RetinaNet did not outperform Faster R-CNN, it might be related to the fact that although there is only one cell type in each image but each cell type was well presented in the dataset. With no surprise, the YOLOX achieved the best performance (mAP=0.334) in all models we have searched. It was a robust model, since it had already performed well enough (comparing with all other detector models we had searched so far) on Kaggle dataset.

5.3. Model Comparison

After several tries in our hyper-parameter tuning, we decided to utilize Mask R-CNN as our baseline and explore the performance increase under different backbones: ResNet50, ResNet101 and SWIN-T. We also recorded the performance of RetinaNet-UperNet-SWIN-T which employed RetinaNet as object detector and UperNet-SWIN-T as the instance segmentor. We compared all of our models with YOLOX-UperNet-SWIN-T, the kaggle first place solution, [3] which employed YOLOX as object detector and UperNet-SWIN-T as the instance segmentor. Both the objector and segmentor are pretrained on LiveCell dataset.

The model performances can be found in Fig. 10. Based on the models in our radar, YOLOX-UperNet-SWIN-T is still the best models with the highest mAP in both BBOX prediction (0.373) and instance segmentation (0.337) evaluations. It also achieved the highest AP each cell class.The SWIN-T under MASK R-CNN structure was reported to outperform both YOLOX (0.587 vs 0.509 in object detection) and UperNet-SWIN-T (0.511 vs 0.502 in segmentation) on COCO dataset. But we failed to reproduce it in our training. Generally MASK R-CNN structure could provide good results in our first run in all backbone choices without too much parameter tuning even on Kaggle dataset itself. We attributed this to the architecture network design in MASK R-CNN. We also noticed that training the loss of BBOX detection and instance segmentation together in

[3] https://www.kaggle.com/competitions/sartorius-cell-instance-segmentation/discussion/298869

MASK R-CNN increased the mAP (0.266) in BBOX compared to Faster R-CNN (0.237) training loss of BBOX detection only. Enhancing the backbone (replacing ResNet50 with ResNet101) and pretraining on LiveCell dataset provided improvement in terms of mAP and AP in all cell types.

5.4. Ablation Study

In order to get a better understanding of how the best model YOLOX-UperNet-SWIN-T works, we plotted an ablation study on it by switching on and off the pretraining on LiveCell dataset for both detector (YOLOX) and segmentor (UperNet-SWIN-T). The results could be viewed in Fig. 11.

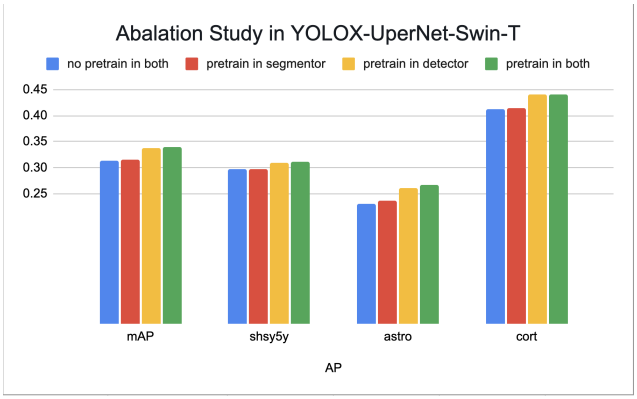


Figure 11. Performance analysis on YOLOX-UperNet-Swin-T model with/o pretrained on LiveCell data for both object detector and instance segmentation.

From the results we observed that the benefit of further pretraining on LiveCell data was mainly coming from the detector. This also lined up with our previous hypothesis that detector failing to capture the BBOX of target cell instances was the major bottleneck for this challenge. YOLOX provided a good BBOX predictions as input to the segmentor UperNet-Swin-T.

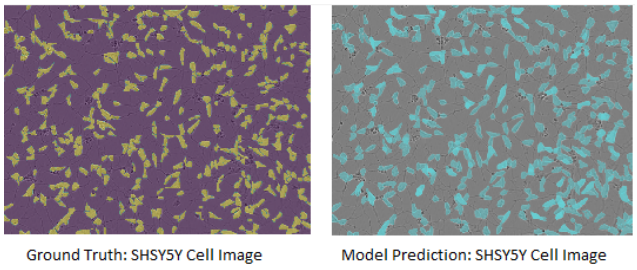
5.5. The analysis on WBF ensemble

We ran WBF on 25 different combinations using four detector outputs from YOLOX (with pretrained on LiveCell), YOLOX (trained on Kaggle only), Faster RCNN, and RetinaNet models. The WBF was used for combining the bounding box predictions. Fig. 12 showed mAP scores for each of these combinations. Assumption is merging predictions from a good model and a bad model will result in predictions with overall mAP worse than the original predictions from the good model. But, improvements can be seen in mAPs specific to different object sizes. For example, RetinaNet performs well on medium and large objects and thus after merging it with YOLOX (pretrained) we actually increase the overall performance on medium and large objects. Merging detector predictions between equally performing models actually increased precision by a significant amount. This can be seen in the data from Faster R-CNN and RetinaNet individually and comparing it to the WBF fused predictions in (Faster R-CNN RetinaNet).

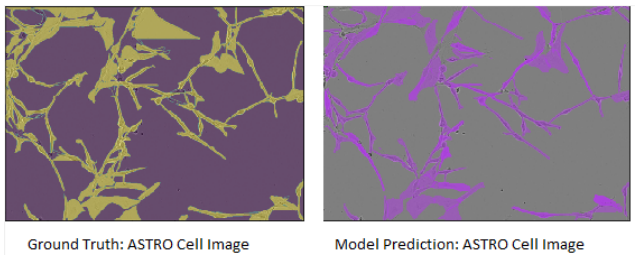
Due to the limit of time and computing resources, we couldn't find an ensemble model to outperform the best model so far (YOLOX-UperNet-SWIN-T). But the analysis showed the potential in applying WBF on ensemble models to achieve better results.

5.6. Visualization

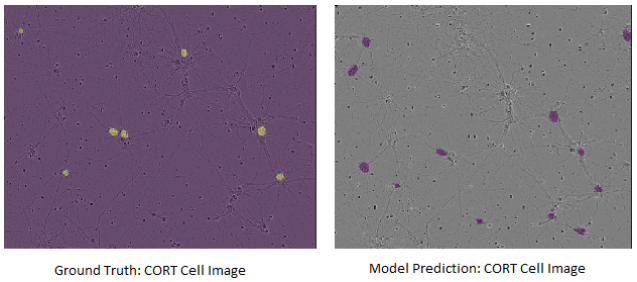
Fig. 13a, 13b, and 13c show prediction output with instance segmentation for each detected cell under YOLOX-UperNet-SWIN-T, the best model in our research.



(a) Prediction vs Ground Truth: SHSY5Y cell.



(b) Prediction vs Ground Truth: ASTRO cell.



(c) Prediction vs Ground Truth: CORT cell.

We observed that the instance cells in CORT cell-type image (Fig. 13c) were usually sparse. Our model could easily detect the BBOX and hence label the correct instance masks. YOLOX also performed well enough in dense labels such as in Fig. 13a, the SHSY5Y cell-type image. The low performance (in AP) at ASTRO cell-type images (Fig. 13a) might be caused by the irregular shapes in ASTRO cell.

Model Name	Detector Backbone	Pretrained on Livecell Dataset	BBOX				
			mAP	AP by Category			COCO Bechmark
				shsy5y	astro	cort	
Mask R-CNN	ResNet50	N	0.266	0.135	0.262	0.401	0.382
Mask R-CNN	ResNet50	Y	0.278	0.133	0.292	0.408	0.382
Mask R-CNN-2	ResNet101	N	0.274	0.142	0.282	0.399	0.428
SWIN-T(Mask R-CNN)	SWIN-T	N	0.226	0.134	0.148	0.398	**0.587**
RetinaNet-UperNet-SWIN-T	ResNet50	N	0.216	0.076	0.202	0.372	N/A
YOLOX-UperNet-SWIN-T	YOLOX	Y	**0.373 (0.396*)**	**0.317 (0.334*)**	**0.363 (0.399*)**	**0.439 (0.456*)**	0.509

(a) Model Detectors Performance Comparison.

Model Name	Segmentor Backbone	Pretrained on Livecell Dataset	Segmentation				
			mAP	AP by Category			COCO Benchmark
				shsy5y	astro	cort	
Mask R-CNN	ResNet50	N	0.247	0.149	0.181	0.413	0.347
Mask R-CNN	ResNet50	Y	0.258	0.150	0.207	0.419	0.347
Mask R-CNN-2	ResNet101	N	0.251	0.155	0.187	0.411	0.384
SWIN-T(Mask R-CNN)	SWIN-T	N	0.205	0.141	0.079	0.396	**0.511**
RetinaNet-UperNet-SWIN-T	UperNet-SWIN-T	Y	0.204	0.076	0.146	0.387	N/A
YOLOX-UperNet-SWIN-T	UperNet-SWIN-T	Y	**0.337 (0.362*)**	**0.310 (0.328*)**	**0.261 (0.302*)**	**0.440 (0.456*)**	0.502

(b) Model Segmentors Performance Comparison.

Figure 10. Model Comparison. The numbers with ⋆ marker are reported in Kaggle first place authors threads.

	AP 0.5-0.95 all	AP 0.5 all	AP 0.75 all	AP 0.5-0.95 small	AP 0.5-0.95 medium	AP 0.5-0.95 large
Faster R-CNN	0.205	0.455	0.157	0.188	0.109	0.221
Faster R-CNN / RetinaNet	0.24	0.524	0.189	0.211	0.173	0.343
Faster R-CNN (0.205) / RetinaNet (0.216)	0.24	0.524	0.19	0.211	0.175	0.346
Faster R-CNN / YOLOX	0.309	0.683	0.233	0.288	0.175	0.319
Faster R-CNN (0.205) / YOLOX (0.331)	0.315	0.692	0.239	0.293	0.19	0.326
Faster R-CNN / YOLOX pretrain	0.341	0.715	0.28	0.312	0.213	0.43
Faster R-CNN (0.205) / YOLOX pretrain (0.371)	0.353	0.731	0.295	0.321	0.238	0.456
Faster R-CNN / YOLOX pretrain / RetinaNet	0.339	0.716	0.275	0.307	0.248	0.455
Faster R-CNN (0.205) / YOLOX pretrain (0.371) / RetinaNet (0.216)	0.35	0.729	0.29	0.317	0.26	0.471
Faster R-CNN / YOLOX pretrain / YOLOX	0.345	0.724	0.281	0.316	0.231	0.429
Faster R-CNN (0.205) / YOLOX pretrain (0.371) / YOLOX (0.331)	0.35	0.732	0.289	0.32	0.243	0.44
Faster R-CNN / YOLOX pretrain / YOLOX / RetinaNet	0.346	0.727	0.28	0.315	0.261	0.446
Faster R-CNN (0.205) / YOLOX pretrain (0.371) / YOLOX (0.331) / RetinaNet (0.216)	0.351	0.735	0.289	0.321	0.266	0.463
Faster R-CNN / YOLOX / RetinaNet	0.316	0.693	0.241	0.291	0.222	0.39
Faster R-CNN (0.205) / YOLOX (0.331) / RetinaNet (0.216)	0.322	0.702	0.245	0.296	0.234	0.379
RetinaNet	0.216	0.474	0.161	0.183	0.186	0.343
YOLOX	0.331	0.706	0.267	0.306	0.231	0.351
YOLOX pretrain	0.371	0.738	0.325	0.335	0.278	0.481
YOLOX pretrain - RetinaNet	0.347	0.725	0.287	0.314	0.28	0.477
YOLOX pretrain (0.371)- RetinaNet (0.216)	0.355	0.734	0.296	0.321	0.283	0.491
YOLOX pretrain / YOLOX	0.347	0.724	0.285	0.317	0.248	0.431
YOLOX pretrain (0.371) / YOLOX (0.331)	0.348	0.724	0.287	0.318	0.25	0.435
YOLOX pretrain / YOLOX / RetinaNet	0.348	0.73	0.281	0.319	0.278	0.44
YOLOX pretrain (0.371) / YOLOX (0.331) / RetinaNet (0.216)	0.353	0.735	0.289	0.322	0.273	0.463
YOLOX / RetinaNet	0.314	0.69	0.236	0.289	0.24	0.365
YOLOX (0.331) / RetinaNet (0.216)	0.318	0.697	0.239	0.293	0.241	0.339

Figure 12. Performance analysis using WBF with various Detector combinations. The predictions weight for a model is supplied in the parenthesis next to the model name. If there is no weight provided then all the model predictions were weighted the same.

6. Conclusion and Future Work

In this project, we were able to use MMCV toolkit to predict neuronal cell instances in microscopy images from a Kaggle competition. We applied a two-stage model structure to identify best cell instance segmentation model for the task. Overall based on our experiments it is clear that YOLOX as detector and Swin Transformer as segmentor (YOLOX-UperNet-SWIN-T) yielded best results for this problem. Our ablation study showed that YOLOX is the key reason for YOLOX-UperNet-SWIN-T being the rock-star combination for this task. Enhanced YOLOX with pre-trained on LiveCell dataset also increased the final instance segmentation prediction.

While we were not able to out-performed the Kaggle competition winning model we were able to achieve comparable results with one GPU and 24GB memory by using YOLOX and Swin Transformer models. We were able to complete comparative analysis of various detection algorithms and observe bounding box and confidence score outputs for each image in the dataset and compare the model performance using mAP metric. In the future, we could attempt to increase the sample size given better memory and computing power. We would also explore more SOTA models in instance segmentation field for better results. We would also research more on other techniques to improve feature extraction to do better on irregular instance shapes.

Contributions Acknowledgements

Keyu Nie, Swathi Gangaraju and Stefan Zdrnja worked together to research various project ideas. Post identification of the project, all three focused on research on literature available on Neuronal Cell detection and segmentation, papers on approaches and algorithms used for object detection and instance segmentation. We worked together to deliberate on the literature. Swathi found on initial EDA, pre-processing code, and identified our initial analysis approach using UNet, and Keyu and Stefan ran the Mask RCNN baseline models. Keyu researched and proposed the main approach, and initialized most of the models in our research. Swathi and Stefan contributed with ideas to scope the project and finalize design details.

We all divided and conquered creating necessary configuration files and associate code to run 2 detectors each and run segmentation model. Swathi contributed on final results in Retinanet and YOLOX training. Stefan contributed on final results in Faster R-CNN training. Keyu contributed on final results of models including MASK R-CNN, MASK R-CNN-2, MASK R-CNN with pretrain, Faster R-CNN, Faster R-CNN with pretrain, SWIN-T(MASK R-CNN), SWIN-T(RetinaNet), YOLOX, YOLOX with pretrained, UperNet-Swin-T, UperNet-Swin-T with pretrain. Stefan and Swathi also worked on WBF and visualization code. We all contributed to content creation for the project proposal, the project milestone, final report, and the poster. We are thankful to Stefan Su for his guidance and feedback on this project. We also acknowledge that ideas are significantly influenced by contributions of many Kaggle participants who worked on this problem that made us better informed and helped us weigh our options on where we wanted to invest our energies in analysis and execution. Credit goes to the Kaggle competition winner for giving us the idea to explore the MMDet and MMSeg toolkit and to explore YOLOX and Swin Transformers as architecture.

Github Code: https://github.com/Stanford-AI-program-projects/CS-231N-Neuron-Cell-Instance-Segmentation

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[2] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Mmdetection: Open mmlab detection toolbox and benchmark. 2019.

[3] MMCV Contributors. MMCV: OpenMMLab computer vision foundation. https://github.com/open-mmlab/mmcv, 2018.

[4] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[7] Yuming Du, Wen Guo, Yang Xiao, and Vincent Lepetit. 1st place solution for the uvo challenge on image-based open-world segmentation 2021. 2021.

[8] 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.

[9] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[10] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[12] 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.

[13] 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.

[14] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. July 2017.

[15] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Jiacong Fang, imyhxy, Kalen Michael, Lorna, Abhiram V, Diego Montes, Jebastin Nadar, Laughing, tkianai, yxNONG, Piotr Skalski, Zhiqiang Wang, Adam Hogan, Cristi Fati, Lorenzo Mammana, AlexWang1900, Deep Patel, Ding Yiwei, Felix You, Jan Hajek, Laurentiu Diaconu, and Mai Thanh Minh. ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference, Feb. 2022.

[16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[17] 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.

[18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[19] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. 2021.

[20] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[21] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[22] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[23] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[25] 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.

[26] Christian Simon, In Kyu Park, et al. Correcting geometric and photometric distortion of document images on a smartphone. *Journal of Electronic Imaging*, 24(1):013038, 2015.

[27] Roman Solovyev, Weimin Wang, and Tatiana Gabruseva. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, 107:104117, mar 2021.

[28] Roman Solovyev, Weimin Wang, and Tatiana Gabruseva. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, 107:104117, 2021.

[29] Di Tian, Yi Han, Biyao Wang, Tian Guan, Hengzhi Gu, and Wei Wei. Review of object instance segmentation based on deep learning. *Journal of Electronic Imaging*, 31(4):041205, 2021.

[30] 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 (ICCV)*, October 2019.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434, 2018.