CAI 4104/6108: Machine Learning Engineering

Project Report: DeepNucleiNet: Advanced Deep Learning for Precision Nuclei Instance Segmentation

Divya Upadhyay
(Point of Contact)
divyaupadhyay@ufl.edu

Haoyang Li
lihaoyang@ufl.edu

Anamitra Das
a.das@ufl.edu

April 27, 2024

1 Introduction

Accurate segmentation of nuclei is essential in various fields such as medical image analysis, biological research, and drug development. In cancer research, alterations in cell nucleus morphology often indicate changes in pathological conditions. [2] Most existing medical image analysis tools rely on traditional instance segmentation algorithms, such as Fiji [9] and CellProfiler [6]. These tools require frequent adjustments to the configuration file to suit the specific experimental conditions, which can be inconvenient. Additionally, they depend on conventional algorithms like thresholding [7] and watershed [5]. This study utilizes the 2018 Data Science Bowl dataset to explore an automated nucleus detection technique that does not require manual configuration settings through the segmentation of human cell nuclei. Three advanced models SAM, Detectron2, and YOLOv8 - are employed in this study for comprehensive implementation and performance comparison. The goal is to offer researchers a more efficient approach to the detection of nuclei using the latest transfer learning approaches and contribute to the progress of medical image detection technology.

2 Approach: Dataset(s) & Pipeline(s)

2.1 Task

Our objective in this study is to develop automated models for detecting cell nuclei. We aim to accurately identify and segment individual nuclei within microscope images of biological samples, each with its own distinct shape and size. The input data for this task consists of labeled microscope images obtained from biological samples. The model is trained using a substantial amount of labeled data. The model's output includes the mask for each nucleus, which outlines the contours of each nucleus to enable precise segmentation.

2.2 Dataset

The dataset is from BBBC038v1[1], including 729 images depicting nuclei under diverse environmental conditions. It was annotated with 29,464 nuclei in the train set and 7,869 nuclei in the test set.

Table 1: Data Split

Train Set	Valid Set	Test Set
498	166	65

These images are characterized by variations in illumination, staining, cell type, and magnification. Each sample in the dataset is composed of two components: the real cell image and the corresponding mask representing each nucleus within the image. The masks can not overlap, meaning that each pixel belongs exclusively to a single mask.

[1]BBBC038v1 Dataset: https://bbbc.broadinstitute.org/BBBC038/.

2.3 Approach

2.3.1 SAM

We employed the Segment Anything Model (SAM) developed and open-sourced by Meta AI [4]. SAM leverages a powerful transformer-based architecture to generate high-quality segmentation masks for individual nuclei, which helps identify and demarcate nuclei boundaries. The key features of SAM offers Zero-shot generalization, thus it can be used to segment unseen objects. It can be prompted with a variety of input, including points, bounding boxes, and text descriptions. For, instance-segmentation of nuclei, we adopt two approaches, both of which use bounding boxes as prompts.

Approach I: We fine-tuned the pre-trained SAM[2] model. The dataset we had contained input images and corresponding masks in the form of images, with each nucleus represented as a separate mask. During training with this approach, each input image and its masks were resized to 256x256. Each input image was used repeatedly to train the model multiple times, with multiple forward and backward passes equal to the number of images in the masks. We used 576 training images. As this approach required many images, it was very slow, with each epoch taking approximately 40 minutes to train on an A100 GPU with a batch size of 1. We trained the model for 3 epochs using the Adam optimizer and the Dice loss from Monai[3].

Approach II: Instead of using the mask of each nucleus separately, we created multiple bounding boxes representing each nucleus and ran the training loop for each input image and multiple bounding boxes representing the masks of all nuclei. This training approach was faster, taking around 10-20 minutes on an A100 GPU. We trained the model for 98 epochs using the Adam optimizer and the Dice loss from Monai.

Both approaches were evaluated with 50 random samples drawn from the training and validation datasets (size: 134), and the Mean Intersection Over Union (IoU)[4] was calculated with threshold set to 0.5.

2.3.2 Detectron2

To begin, the dataset was preprocessed and transformed into a COCO dataset. The model was trained within the Detectron2[10] framework[5], utilizing the Mask R-CNN framework with ResNet-50 plus FPN as the base network. To accelerate the initial learning process and enhance model performance, pre-training weights provided by Detectron2 were used for model initialization. Throughout the training process, various hyper-parameters were configured, including an initial learning rate of 0.0025, an SGD optimizer with a momentum of 0.9, and a learning rate tuning strategy with gradual decay. Periodic model evaluations were conducted using COCO Evaluator, assessing Average Precision (AP) and Intersection over Union Ratio (IoU). Furthermore, test-time augmentation (TTA) was implemented to assess model performance under diverse data augmentation conditions and ensure result stability and reliability.

2.3.3 YOLOv8

For YOLOv8[3], the dataset was first converted into the YOLO (You Only Look Once) format, from the COCO JSON format. Multiple models were trained using the Ultralytics framework[6]. In this framework, several pre-trained models are available for instance segmentation, from tiny to extra-large. On the first run, we fine-tuned the tiny model on our dataset, without any data augmentation, with a batch size of 4, for 50 epochs with a patience of 5. Then we tried out fine-tuning the larger models, added data augmentation to the training data, and tuned the hyperparameters such as batch size, dropout, epochs, and patience. Setting the optimizer to 'auto' chose the AdamW optimizer with an initial learning rate of 0.002 and a momentum of 0.9, with a gradual decay. The 'valid' mode during training kept track of mask precision, recall, and average precision with different thresholds. The best models from each run were then evaluated on the test set with the 'valid' function.

3 Evaluation Methodology

Our study uses Mean Average Precision (MAP) as the primary metric for assessment. MAP measures the model's capacity to segment instances by computing the AP across various IoU thresholds (ranging from 0.50 to 0.95). The dataset is prepared using COCO format annotations for segmentation, as split in the Dataset section. Subsequently, the model is trained on the training set, and then predictions are generated on the test set to generate the mask and bounding box (bbox) for each nucleus. After this, AP is conducted across IoU values from 0.5 to 0.95 and subsequently averaged to derive MAP. This procedure is illustrated in Figure 9.

[2]Segment Anything Model: https://huggingface.co/docs/transformers/main/en/model_doc/sam.
[3]MONAI: https://docs.monai.io/en/stable/losses.html.
[4]Mean IoU: https://docs.monai.io/en/stable/metrics.html#mean-iou.
[5]Detectron2: https://github.com/facebookresearch/detectron2.
[6]YOLOv8: https://github.com/ultralytics/ultralytics.

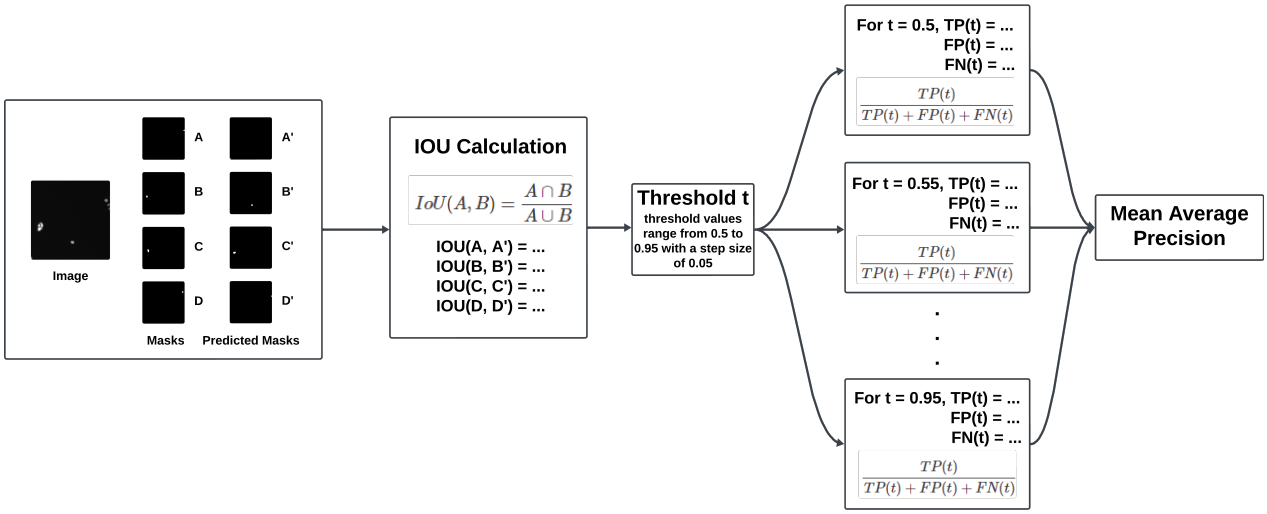


Figure 1: MAP Calculation

The dataset of this study is sourced from Kaggle's DataScienceBowl competition. The baseline performance metric selected for comparison is the top-ranking submission on the Kaggle leaderboard, which achieved a MAP score of 0.63164. The training data was split into training and validation sets with a split ratio of 0.25, as shown in Table 1.

4 Results

4.1 SAM

The performance of the SAM model was analyzed separately for both approaches. For Approach I, Figures 3 and 4 show the Probability Maps produced from the fine-tuned SAM Model for nuclei segmentation, both for the training data and validation data. We can observe a great overlap between the Mask and Probability Map, and promising results when observing the predictions on the unseen validation dataset. However, we can also observe a low Mean IoU of 0.223 for the training data and 0.289 for the validation data after training for 3 epochs. This can be observed due to poor mask quality, as seen in Figure 2, or due to low training samples and low training epochs resulting from the slower training process. To understand the role of longer training with more epochs, we trained the model for 98 epochs in Approach II. Surprisingly, the reduction of the Dice loss plateaued after 25 epochs, and no performance improvement was observed, as shown in Table 3. We can see that though the prediction result and probability maps show greater overlap, we get poor IoU(Intersection Over Union) due to low-resolution input images and masks introduced due to a variety of techniques used to source this dataset as well as some human error.

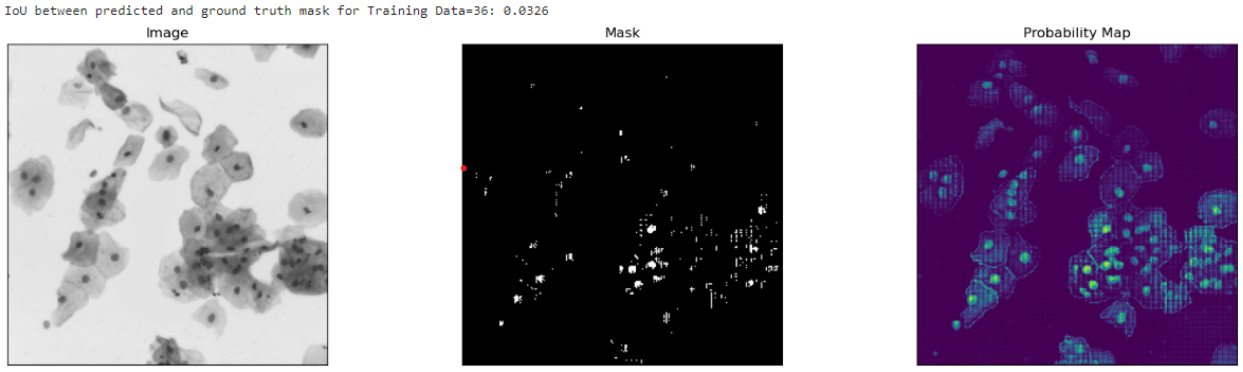


Figure 2: Nuclei Segmentation on Training Data with Poor Mask using SAM

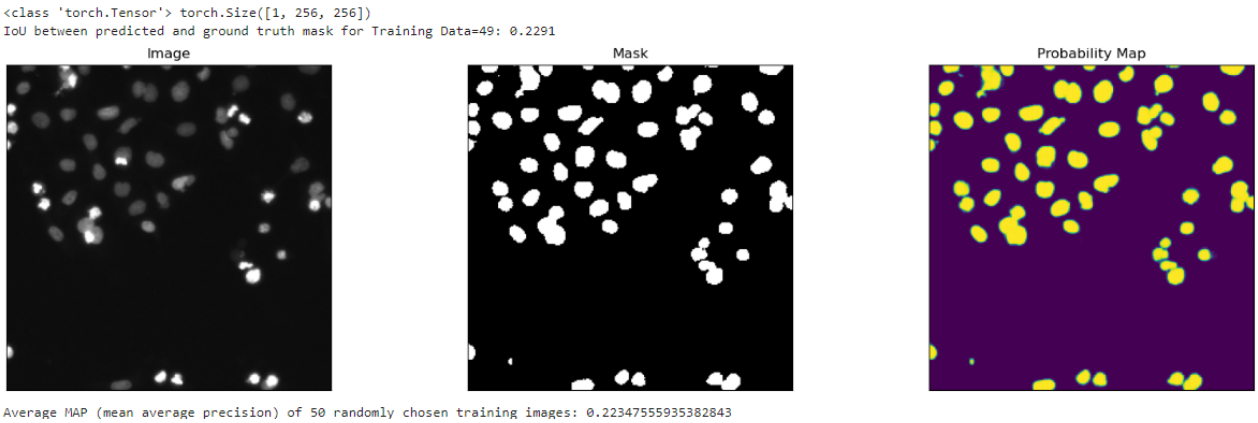


Figure 3: Nuclei Segmentation on Training Data with Approach I(SAM)

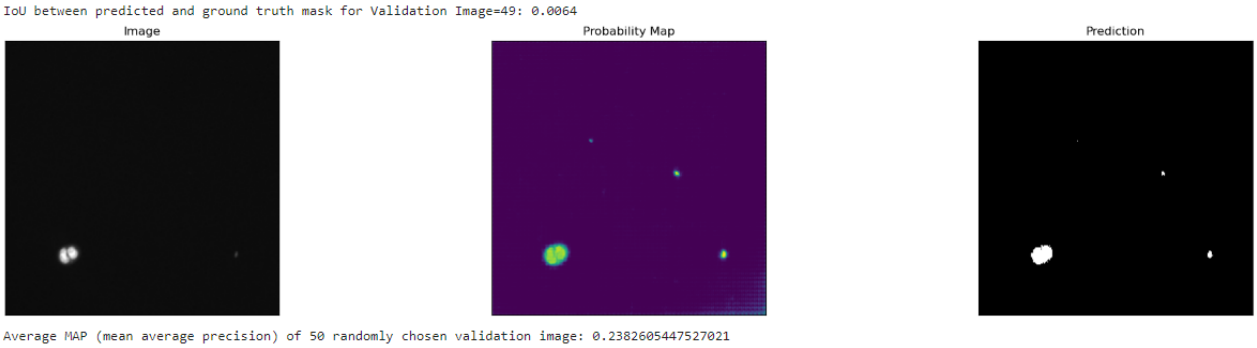


Figure 4: Nuclei Segmentation on Validation Data with Approach I(SAM)

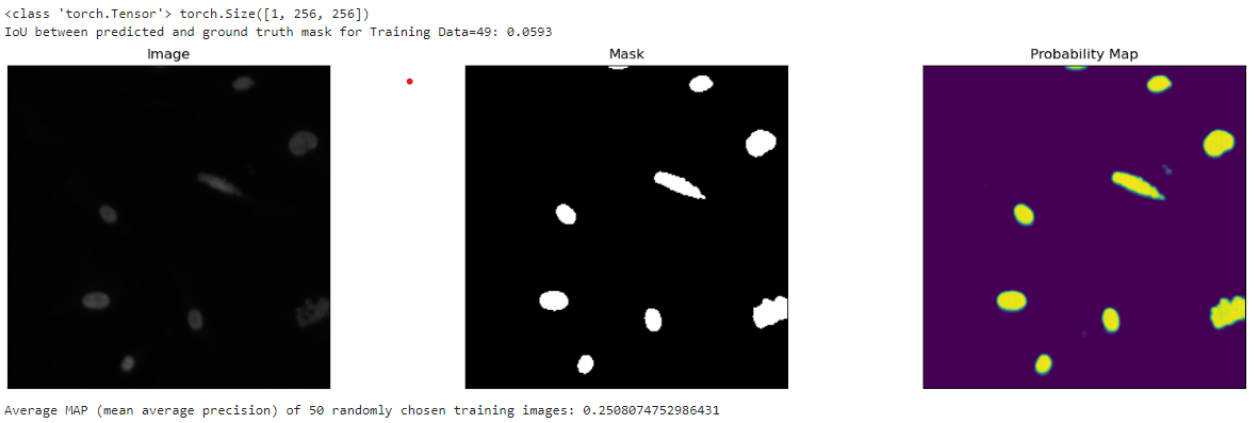


Figure 5: Nuclei Segmentation on Training Data with Approach II(SAM)

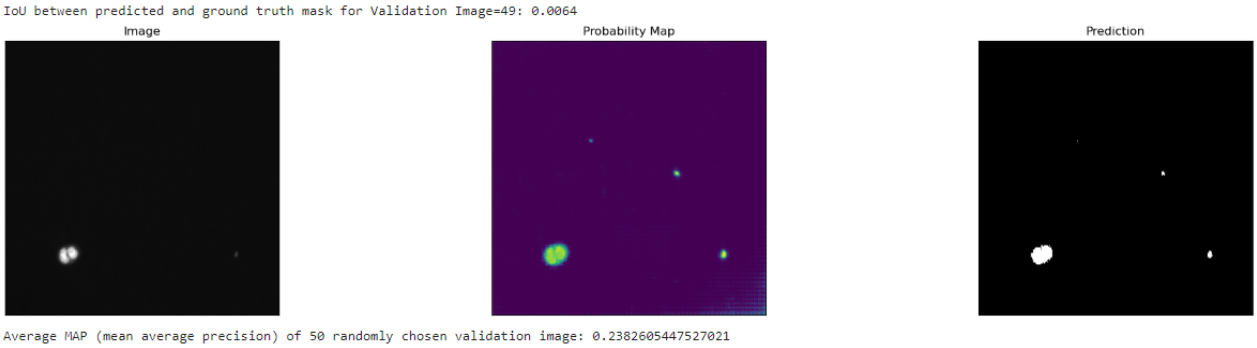


Figure 6: Nuclei Segmentation on Validation Data with Approach II(SAM)

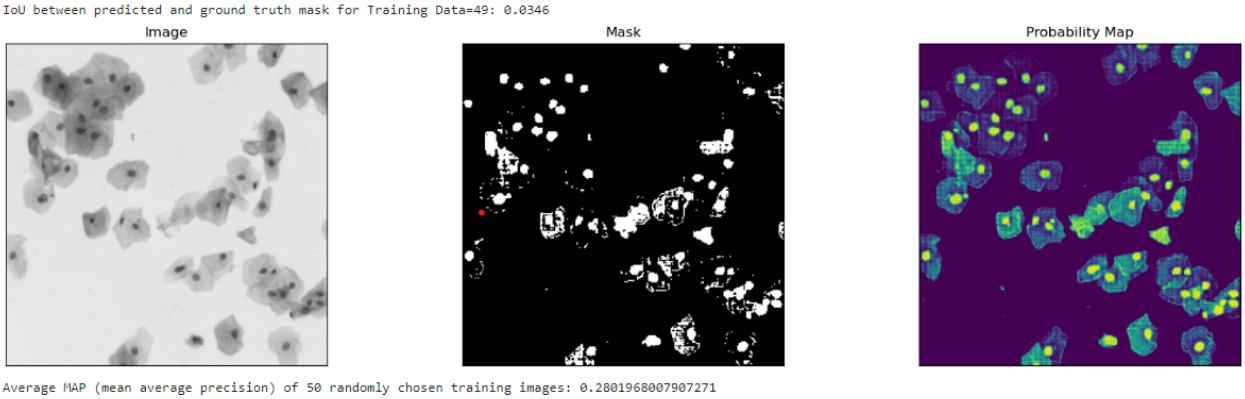


Figure 7: Nuclei Segmentation on Training Data with Approach II(SAM)

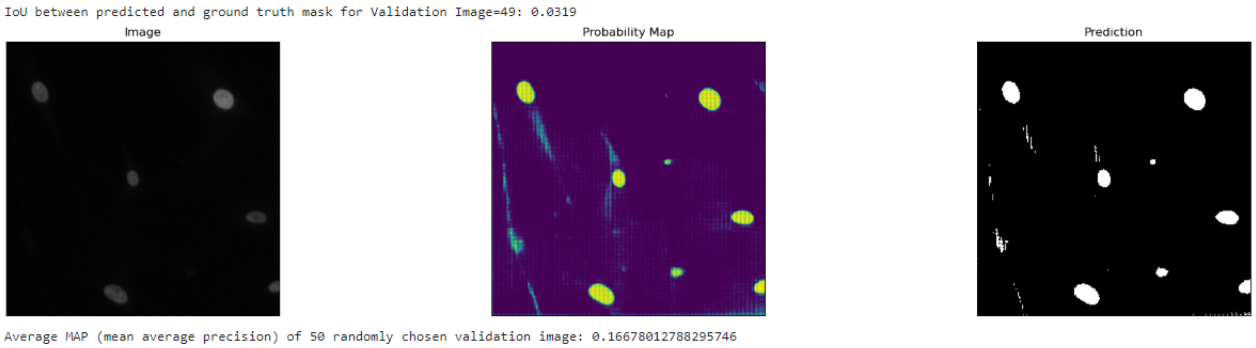


Figure 8: Nuclei Segmentation on Validation Data with Approach II(SAM)

Table 2: Dice Loss and IoU in Approach I

Epoch	Mean Dice Loss
1	48368.90325064422
2	48133.903622367834
3	48087.70372782318

Dataset	IoU
Training	0.223475
Validation	0.2889460

Table 3: Dice Loss and IoU in Approach II

Epoch	Mean Dice Loss
1	40178.0204
25	37745.9433
50	37610.5766
98	37495.4389

Epoch	Dataset	IoU
25	Training	0.250807
	Validation	0.238260
50	Training	0.280196
	Validation	0.166780
98	Training	0.233741
	Validation	0.172054

4.2 Detectron2

The performance of the model is shown in Table 4. The model is strong in segmenting nuclei, especially when it comes to an IoU of 0.50, where the AP reaches 73.776%, indicating a high level of segmentation capability. However, as the conditions become more strict, such as under IoU=0.75, the AP drops to 55.046%, suggesting that the model is effective in localizing and segmenting nuclei. However, its precision in capturing edge details needs improvement. This may be because of the model's insufficient refinement for edge segmentation and limited training epochs due to computational constraints; increasing the number of training epochs might help address this issue.

Table 4: Bbox and Segm Metrics

Metric	IoU	Area	MaxDets	Value
AP	0.50:0.95	all	100	0.485
AP	0.50	all	100	0.738
AP	0.75	all	100	0.551
AP	0.50:0.95	small	100	0.437
AP	0.50:0.95	medium	100	0.775
AP	0.50:0.95	large	100	-1.000
AR	0.50:0.95	all	1	0.018
AR	0.50:0.95	all	10	0.170
AR	0.50:0.95	all	100	0.530
AR	0.50:0.95	small	100	0.484
AR	0.50:0.95	medium	100	0.813
AR	0.50:0.95	large	100	-1.000

Metric	IoU	Area	MaxDets	Value
AP	0.50:0.95	all	100	0.492
AP	0.50	all	100	0.738
AP	0.75	all	100	0.550
AP	0.50:0.95	small	100	0.444
AP	0.50:0.95	medium	100	0.789
AP	0.50:0.95	large	100	-1.000
AR	0.50:0.95	all	1	0.018
AR	0.50:0.95	all	10	0.171
AR	0.50:0.95	all	100	0.537
AR	0.50:0.95	small	100	0.492
AR	0.50:0.95	medium	100	0.816
AR	0.50:0.95	large	100	-1.000

Furthermore, APl is NaN, indicating a lack of large-sized target samples in the dataset. Comparing APs and APm reveals that the model performs most effectively on medium-sized nuclei but is less accurate on smaller objects. This discrepancy may be due to a scarcity of small-sized targets in the training samples or inadequate capture of details by the model's feature extraction layer.

Although our model's MAP score doesn't reach 0.63164, its performance (a MAP score of 0.492) can be ranked within the top 20% of submissions on the leaderboard.

Additionally, Meta provides the baseline[7] for the Mask R-CNN model, shown in Table 5. The R50-FPN model achieves a boxAP of 41.0 and a maskAP of 37.2. Upon comparison with Table 6, we can see that the model achieves a boxAP of 48.462 and a maskAP of 49.211, surpassing the baseline metrics.

[7]Detectron 2 Baseline: https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md.

Table 5: COCO Instance Segmentation Baselines with Mask R-CNN

Name	lrsched	traintime (s/iter)	inferencetime (s/im)	trainmem (GB)	boxAP	maskAP
R50-C4	3x	0.575	0.111	5.2	39.8	34.4
R50-FPN	3x	0.261	0.043	3.4	41.0	37.2

Table 6: Bbox and Segm AP of the Model

Name	AP	AP50	AP75	APs	APm	APl
Bbox	48.462	74.751	55.144	43.714	77.452	nan
Segm	49.211	73.776	55.046	44.371	78.880	nan

The model's detection performance is shown in Figure 2, showing acceptable performance. According to the AP, this could be attributed to inaccuracies in data labeling and inaccurate detection of edges and small objects. Overall, the model can roughly locate and segment objects, but struggles to segment their edges accurately.

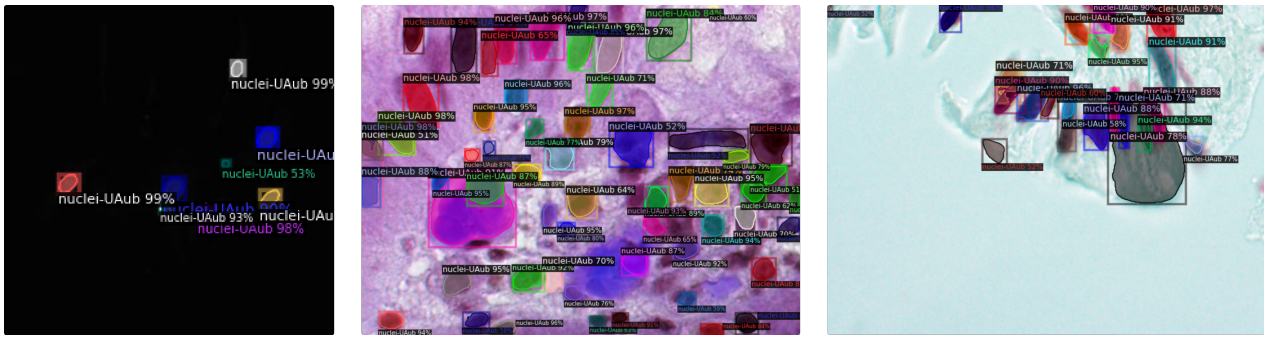


Figure 9: Detectron2 Results

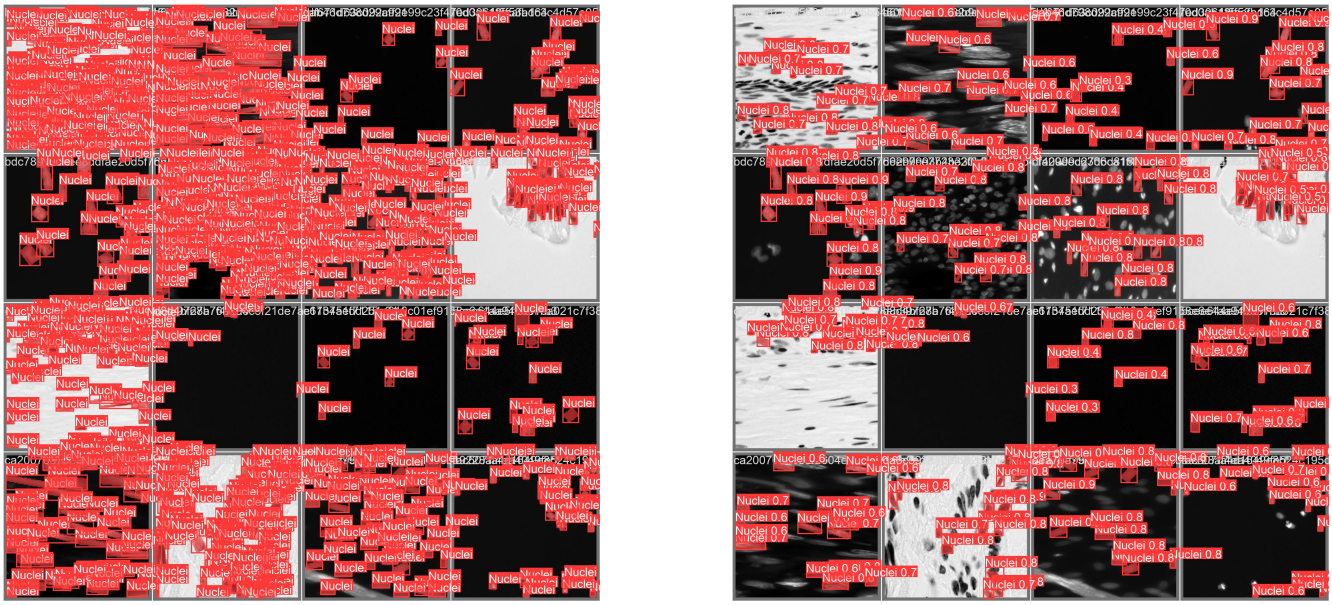
4.3 YOLOv8

The performance of the fine-tuned YOLO models on the test data is shown in Table 7. The model is very mediocre at detecting and segmenting nuclei. At IoU 0.50, the best YOLO model reaches an AP of 67.8%, which seems decent. However, the AP is only 23.6% at an IoU of 0.75. This implies that the model fails to capture the edge details with precision. Looking at the test outputs, in Figure 10, the problems become more apparent. Each of the 4 runs was done with different batch sizes and data augmentation techniques used on the training set. Despite improvement with each run, the final performance is still lackluster.

YOLOv8 is the first version of YOLO where segmentation is possible. The YOLO architecture is more optimized toward objection detection, which may be a good reason why edge details on the pixel level are often missed by it. It also struggles with small object size and high density like that of nuclei images, for the same reason. The shift in the type of images in the test set means that the model is overfitting to the training set, since there is a noticeable difference in performance on the validation and the test sets. For example, 47.2% is the mean average precision (mAP50-95) for the validation set for the Large-200epochs model, whereas, it is only 31.4% for the test set. Therefore, the limited training data available in this set might also be a reason for this performance. Figure 11 has more training and validation metrics for the best-performing YOLO model.

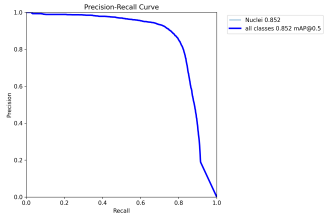
Table 7: YOLOv8 Segmentation Metrics on Test set

Model	conf	Precision	Recall	mAP50	mAP75	mAP50-95
Large-200epochs	0.5	0.887	0.472	0.678	0.236	0.314
XL-150epochs	0.5	0.863	0.487	0.672	0.208	0.298
XL-100epochs	0.5	0.844	0.409	0.627	0.192	0.269
Tiny-50epochs	0.5	0.852	0.310	0.578	0.194	0.261

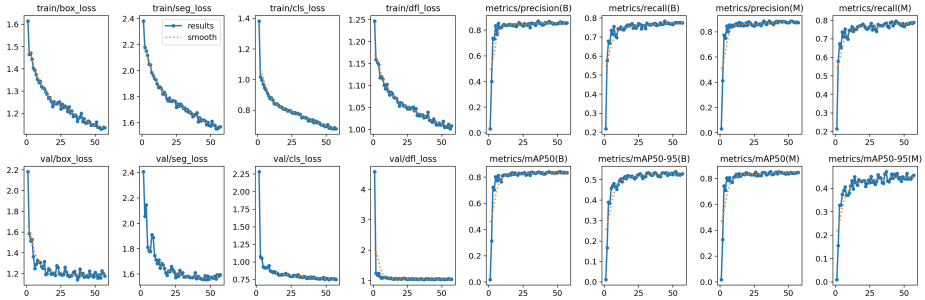


(a) Labeled (b) Predicted

Figure 10: YOLO test samples - Labeled images vs Predicted labels



(a) Mask Precision-Recall curve



(b) Training losses

Figure 11: YOLO training

5 Conclusions

The models used in this project are considered state-of-the-art in their domains. However, that does not mean they can adapt well to every domain. The Segment Anything Model (SAM) demonstrated promising results for binary segmentation nuclei, with good overlap between predicted masks and ground truth. As SAM uses a Transformer based approach it is not good fit for images with low resolution due to its heavy dependence of points and bounding boxes as prompt. While these results are encouraging, SAM is not specifically designed for instance segmentation and may require extensive post-processing. YOLOv8 could not adapt well to this dataset and its performance was not at all comparable to our baseline/benchmark. Detectron2 outperforms SAM and YOLO, with a top 20% finish in the Kaggle leaderboard. However, it is still trailing much behind our baseline. Figures 12 and 13 compare the predictions made by the 3 approaches.

Detectron2 was the better performer here because it was designed specifically with Instance Segmentation in mind. As a Mask R-CNN based approach, the edge detection precision is slightly lacking compared to encoder-decoder architectures like Unet[8] and its variants such as Unet++[11] and DeeplabV3+[1], both of which were used in some of the top solutions as the first step(for binary segmentation), followed by the watershed algorithm to separate the nuclei. The Unet-based architectures use skip connections to preserve more pixel-level detail, which is why those approaches might show higher MAP scores. However, Detectron2 can be an excellent starting point for beginners, given how easy it is to pick up. The top submissions on Kaggle often use advanced techniques like ensembling using a voting mechanism for better performance. Using external data to enrich the small training set would also allow the models to learn and generalize better over a variety of nuclei in images. Using these techniques with Detectron2, paired with better computational resources, to enable using larger batch sizes and training for more epochs, we should achieve comparable performance to our benchmark.[Code]

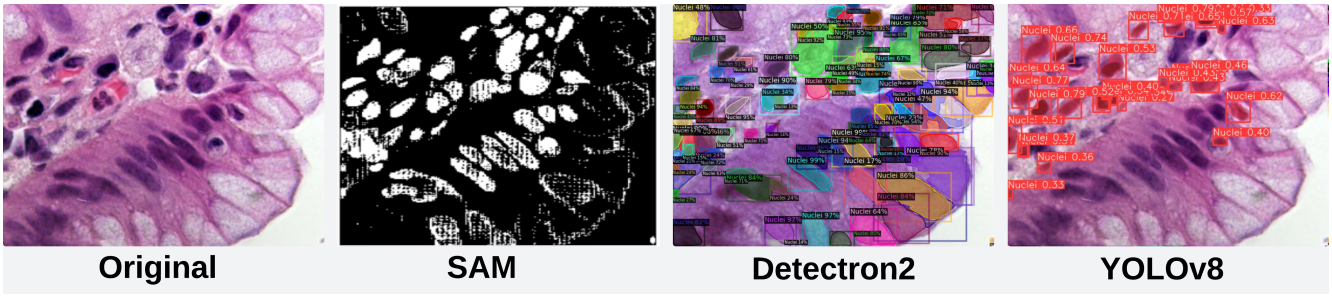


Figure 12: Models compared on the test set

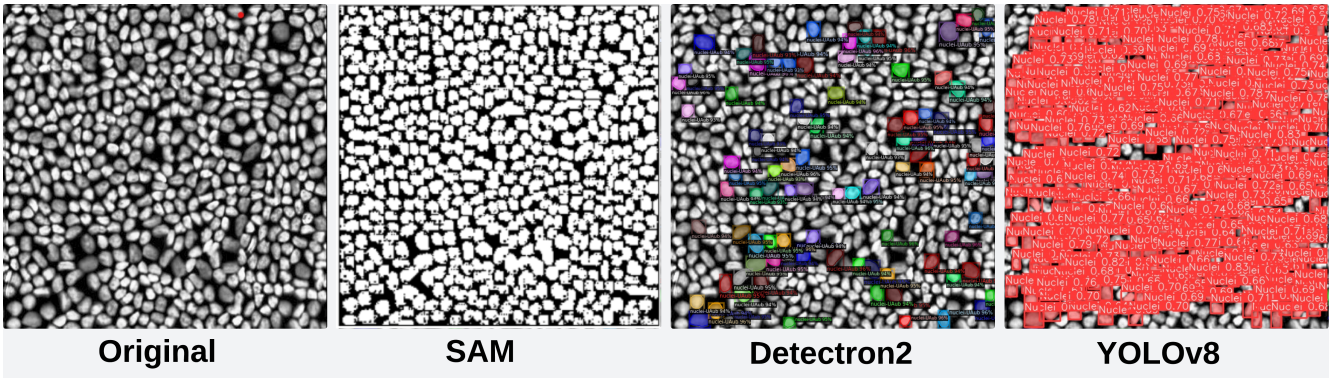


Figure 13: Models compared on the other data

References

[1] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[2] Edgar G. Fischer. Nuclear morphology and the biology of cancer cells. *Acta Cytologica*, 64:511–519, 2020.

[3] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.

[4] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[5] N. Malpica et al. Applying watershed algorithms to the segmentation of clustered nuclei. *Cytom. A*, 28:289–297, 1998.

[6] C. McQuin, A. Goodman, V. Chernyshev, L. Kamentsky, B.A. Cimini, K.W. Karhohs, et al. Cellprofiler 3.0: Next-generation image processing for biology. *PLoS Biol*, 16(7):e2005970, 2018.

[7] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man. Cybern.*, 9:62–66, 1979.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[9] J. Schindelin, I. Arganda-Carreras, E. Frise, et al. Fiji: an open-source platform for biological-image analysis. *Nat Methods*, 9:676–682, 2012.

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

[11] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*, pages 3–11. Springer, 2018.