Microscopy Dataset Segmentation and Object Detection

Neelu Nerella, Pranav Mishra, Pranav Bhardwaj, Mehar Gambhir

December 3 2023

Dataset

The Electron Microscopy Dataset offers a comprehensive view of a 5x5x5 um section from the CA1 hippocampus region of the brain, translating to a 1065x2048x1536 volume. With a voxel resolution of approximately 5x5x5 nm, the dataset has been graciously provided as structured TIF files for training and validation. Both training and test dataset are one TIF file each consisting of 165 2D slices each. In addition the dataset also has binary ground truth images that shows proper segmentation and act as output value for training and testing. Notably, our focus was drawn to this dataset due to its meticulous annotations of mitochondria in two sub-volumes, making it an instrumental tool in our line of research.

Objectives

Primary Objective: Develop a segmentation model capable of effectively working with the dataset, striving to achieve the optimal Intersection over Union (IoU) score.
Secondary Objective: Progress to semantic segmentation with the aim to track specific cell bodies throughout the dataset, adding bounding boxes for object detection.

1 Methodology & Implementation

1.1 Basic Histogram Segmentation

We kick-started our project with a basic histogram segmentation leveraging Otsu's threshold. This approach set the foundational framework upon which subsequent methodologies were implemented. The image was converted to grayscale, threshold value was calculated and a segmented binary image obtained. We added post-processing techniques such as morphological operations, hole filling, skeletonize, etc.

1.1.1 Description

Otsu's method is a popular technique employed for image thresholding. It separates an image into two classes, foreground, and background, based on the grayscale intensity values of its pixels. Furthermore, Otsu's method uses the grayscale histogram of an image to detect an optimal threshold value that separates two regions with maximum inter-class variance.

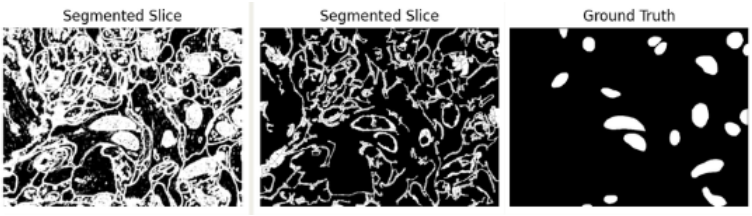

Figure 2: Thresholding Predicted Masks

1.1.2 Results

We started with a simple pixel level accuracy measure and an IOU score. The table below shows the results from using the thresholding mentod before and before each round of morphological post processing. The image shows the predicted masks before and after the post-processing compared with the ground truth

Metric	No post processing	First round of post processing	Second round of post processing
Train Accuracy	58%	66%	79.6%
Test Accuracy	59%	66.2%	79.8%
Train IOU score	0.115	0.137	0.085
Test IOU score	0.110	0.130	0.077

Figure 1: Thresholding Results

1.2 UNet Segmentation (Base)

After working on the Basic Histogram Segmentation, we moved on to UNet Segmentation. We used deep learning starting with creating an untrained unet architecture training from scratch. We got poor scores even after parameter tuning.

1.2.1 Description

The U-Net Segmentation is a Convolutional Neural Network (CNN) designed for biomedical image segmentation. We implemented a basic U-Net Model with a minimal set of encoders and decoders. This architecture was chosen for its proven track record in segmentation tasks.

1.2.2 Results

The image below shows that the U-Net Segmentation from scratch is not good at detecting cells which resulted in overfitting. The Training and Validation Loss shows that the data is overfitting because while the training loss approaches 0, validation loss plateaus, which means that the method isn't optimal for performance.

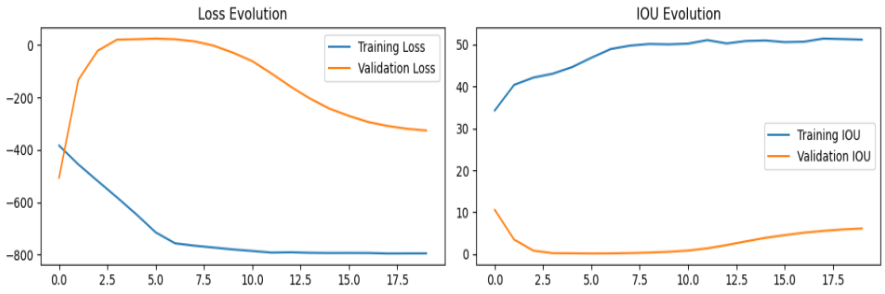

Figure 3: UNET (Base) Graphs for Loss and IOU Evolution

We can also see overfitting when looking at the IOU Evolution graph. As the number of epochs increase, the Training IOU increases and the Validation IOU plateaus near 0 which means that the model followed the training data too closely.

1.3 Unet Segmentation with a Pre-trained Backbone

After working with an untrained architecture, we decided to work on UNet with a pre-trained VGG19 backbone in our UNet architecture, which resulted in better scores. We also added in post-processing techniques as well for better prediction scores.

1.3.1 Description

Leveraged the VGGNet architecture as the backbone of our U-Net model, initializing it with pre-trained ImageNet weights for enhanced feature extraction.

1.3.2 Results

With this new approach, we stopped training the model to avoid overfitting after epochs 13.

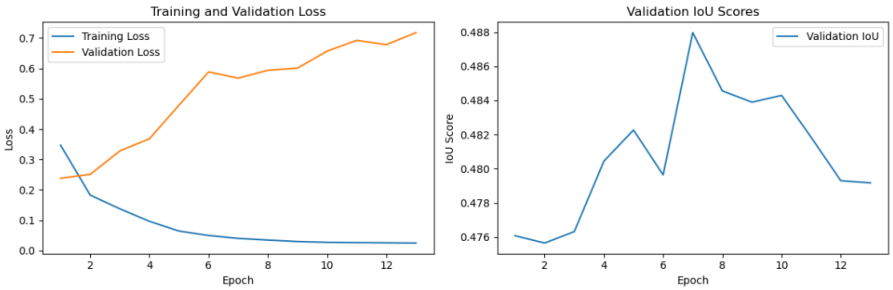

Figure 4: UNET evaluation Graphs for Loss and IOU Evolution

In the image below, the predicted mask shows where the predicted cells are in the whole image. The Post-Processed Predicted Mask improves the final result using Binary Masks and Morphological Operations.

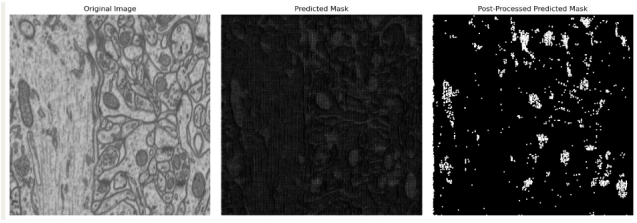

Figure 5: UNET Segmentation of Pretrained Backbone Masks

1.4 Single Shot Object Detection

We then switched approaches to Object Detection. With the SSD, we created a function to automatically and accurately annotate bounding boxes, which is crucial for object detection models. We implemented SSD using TensorFlow and PyTorch libraries. We got good evaluation metrics and meaningful predictions.

1.4.1 Description

The annotation function uses a dynamic kernel size adaptation method, which determines the kernel size for the dilation operation based on the contour area of detected objects within the binary mask. After this, we recompute the contours, representing more accurately separated objects and calculate the bounding box coordinates..To verify its accuracy , we implement a visualization function function that randomly selects a mask from the dataset, draws the calculated bounding boxes on the mask, and displays the original mask alongside the mask with bounding boxes.

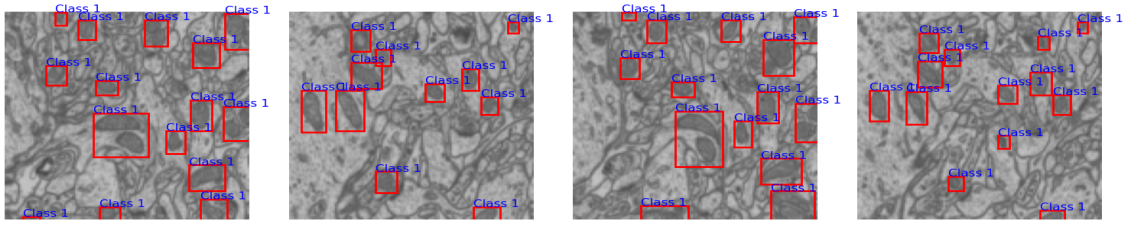

Figure 6: Evaluation of bounding box annotations

SSD is a fast and efficient way to detect multiple objects in a class within a image in one go, hence the name "Single Shot". It uses a single neural network with an appropriate backbone to compute bounding boxes for object of interest and hence is computationally inexpensive.

1.4.2 Results

We trained the annotated data using SSD for 25 epochs after which it converged to a loss value of 8.04 and average IOU of 0.589.

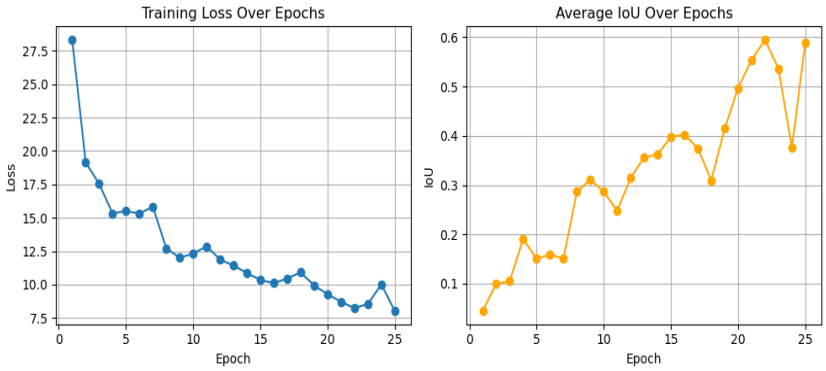

Figure 7: Model Evaluation of Single Shot Detection

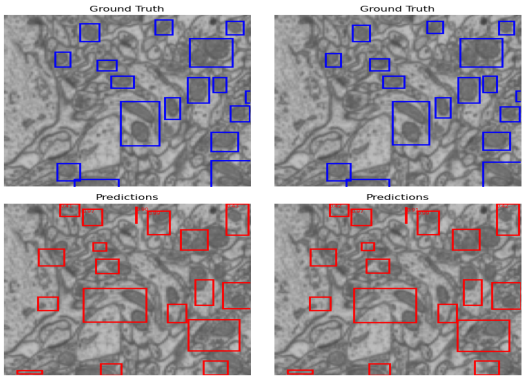

Figure 8: Example predictions of Single Shot Detection

We made predictions for the bounding boxes and compared them to the ground truth. Based on the area match, a truth value is assigned to each of the annotations using a threshold value (0,1). We calculated precision and recall values for threshold values [0.1 - 0.9] and plotted a curve.

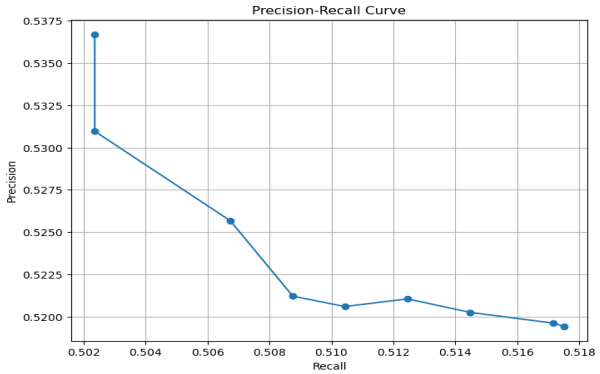

Figure 9: Precision Recall Curve predictions for Single Shot Detection

1.5 Fast R-CNN Models

In the first attempt, we built a Fast R-CNN model with a basic Region Proposal Network (RPN). Unfortunately, this model exhibited a stagnated loss of 29 after more than 20 epochs of training.

Subsequently, we took a different approach by constructing a new model. This model featured a pretrained Feature Pyramid Network (FPN) and a trainable RPN integrated into the Fast R-CNN architecture. As a result, this modification led to significantly improved scores and achieved the best predictions to date.

1.5.1 First Implementation

Our implementation starts with the creation of a custom dataset to load data efficiently in batches, ensuring that the model receives properly formatted inputs during training and evaluation.

Model Architecture

The core of our Fast R-CNN implementation is the model architecture. We load a pre-trained VGG16 model and modify it to serve as our feature extractor. We remove the last max-pooling layer to retain richer feature maps. Our model consists of the following components:

- Feature Extractor: Utilizing the pre-trained VGG16 model, we extract feature maps from input images.
- Region of Interest (RoI) Pooling: We employ RoI pooling to adaptively transform regions of varying sizes into a fixed-sized feature map (7x7). This step aligns RoIs with the feature maps, preserving spatial relationships.
- Bounding Box Regressor: Similar to the classifier, the bounding box regressor comprises fully connected layers, generating bounding box deltas for precise localization.
- Classifier: Our classifier consists of fully connected layers, including ReLU activations and dropout, ultimately producing class logits for object classification.
- Multi-Task Loss: To optimize the model, we employ a multi-task loss function, combining classification and bounding box regression losses. The classification loss is computed using cross-entropy, while the regression loss utilizes the Smooth L1 loss. These losses jointly guide the model to classify objects accurately and refine bounding box predictions.
- Non-Maximum Suppression (NMS) During inference, we apply Non-Maximum Suppression to filter overlapping bounding boxes and retain the most confident predictions. NMS helps eliminate redundancy in the output, producing cleaner and more reliable results.

Training challenges

Despite these unique features, our initial training results did not yield the expected performance. The loss curve stagnated at a relatively high value of 29, and precision and recall metrics hovered around 0.6. This phenomenon suggests that the model may face convergence issues, and further investigation is required to address this challenge effectively.

1.5.2 Second Implementation

In this research, we present an enhanced implementation of the Fast Region-based Convolutional Neural Network (Fast R-CNN) for object detection tasks. We leverage the advantages of a pre-trained Faster R-CNN model with a Feature Pyramid Network (FPN) and a ResNet50 backbone. This model architecture enhances feature extraction, object detection, and region proposal generation, leading to improved object detection performance.
By incorporating a pre-trained Faster R-CNN model with FPN, we address the challenges of multi-scale object detection and feature extraction. The FPN enables the model to handle objects of varying sizes effectively. Moreover, the Faster R-CNN model integrates a Region Proposal Network (RPN) for generating region proposals, streamlining the model's architecture and reducing the need for custom RPN implementations.

Model Architecture

Our enhanced Fast R-CNN model consists of the following components:

- Pre-trained Faster R-CNN with FPN: This model incorporates a ResNet50 backbone for feature extraction, utilizing a Feature Pyramid Network (FPN) for improved multi-scale feature handling.
- Replacement of ROI Classifier and Box Predictor: While retaining the Faster R-CNN's feature extraction and RPN components, we replace the ROI heads' classifier and box predictor. This step customizes the model for our specific object detection task, ensuring it tailors its final classification and bounding box prediction layers accordingly.

1.5.3 Results

During training, our model exhibited significant progress, with the loss consistently decreasing over the course of 60 epochs with final loss being 4.8532. These training metrics demonstrate the model's ability to converge effectively and efficiently during the training process. We evaluated the model's performance on a validation set, yielding the following metrics:
Best F1 Score: 0.8635
Best Precision: 0.8713
Corresponding Recall: 0.8558

These metrics reflect the model's strong precision and recall rates, highlighting its suitability for various object detection tasks.

The following is the metrics and graphs from the second implementation of Fast R CNN model as discussed above

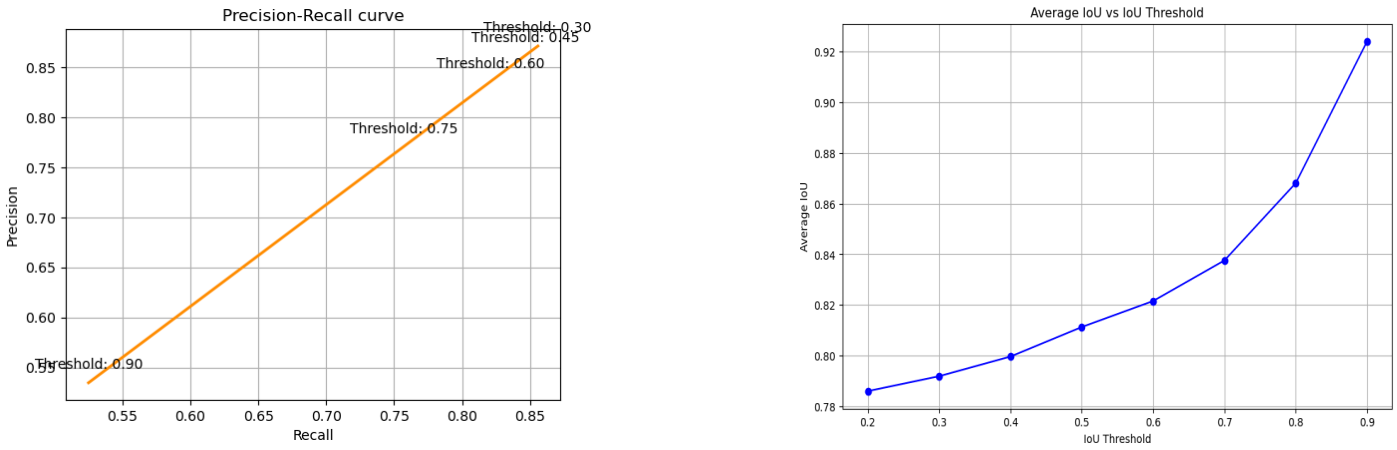

Figure 11: Model Evaluation for Fast R-CNN

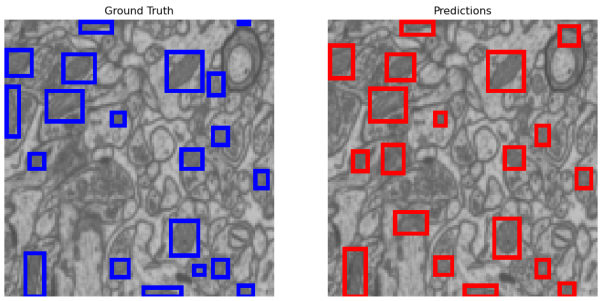

Figure 10: Comparison of ground truth and Predictions from Fast R-CNN model

1.5.4 Use of backbones

In this section, we explore the performance of the Fast Region-based Convolutional Neural Network (Fast R-CNN) with different backbone architectures. Specifically, we utilize the MobileNetV2 and DenseNet121 backbones to evaluate the model's object detection capabilities. The comparative analysis provides insights into the impact of backbone selection on object detection performance.

- MobileNetV2
  Best Precision: 0.8756
  Corresponding Recall: 0.8684
- DenseNet121
  Best Precision: 0.7785
  Corresponding Recall: 0.8794

MODEL	Total Loss	Precision (best)	Recall (at best precision)	IOU (Threshold = 0.9)
SSD Model	8.04 (25 epochs)	53.6%	50.2%	58.96%
Fast RCNN ResNet50 bb	**4.85 (60 epochs)**	87.2%	85.5%	92.41%
Fast RCNN MobileNet bb	5.6 (60 epochs)	**87.5%**	86.8%	92.47%
Fast RCNN DenseNet bb	11.33 (60 epochs)	77.8%	**87.9%**	**92.63**

Figure 12: Metrics comparision for object detection models

The performance evaluation of object detection models is encapsulated in a summarized table, which contrasts the SSD Model against three Fast R-CNN variants with different backbones: ResNet50, MobileNet, and DenseNet. The SSD model trails with higher total loss and lower precision and recall, while Fast R-CNN models demonstrate enhanced precision and recall rates, with the DenseNet backbone slightly outperforming the others. All Fast R-CNN models achieve remarkable IOU scores at a high threshold of 0.9, indicating precise object localization capabilities.

Conclusion

Finally, our research successfully implemented and analyzed several segmentation methodologies and object detection architectures, demonstrating that Fast R-CNN models, particularly with DenseNet backbones, provide robust detection and classification with excellent precision and recall. The models' constant high IOU scores at severe thresholds highlight their outstanding ability to reliably position objects, demonstrating their potential for hard object recognition tasks in complicated visual data. This comprehensive analysis not only confirms the efficacy of advanced convolutional networks in object detection but also provides a solid foundation for future explorations into more intricate or domain-specific applications.