3D segmentation of Mitochondria

Vasu Swaroop

April 2023

1. Introduction

This study discusses the semantic segmentation implementations on the Electron Microscopy Dataset obtained by Graham Knott and Marco Cantoni at EPFL:- Dataset. Semantic segmentation refers to forming partitions of a 3D volume which essentially classifies them into classes. This report includes the best results I was able to generate. All my previous/discarded attempts can also be referred to in the link.

2. Dataset Extraction and Data Cleaning

The dataset consisted of a 5x5x5 µm^3 section taken from the CA1 hippocampus region of the brain, corresponding to a 1065x2048x1536 volume. The resolution of each voxel is approximately 5x5x5 nm^3. The dataset consisted of TIFF files. This dataset was not annotated, and hence could not be used. However, it helped with qualitative understanding of the data.

The training and testing dataset each consisted of 165 slices of the 1065x2048x1536 image stack which had ground truth annotation. The training dataset was corresponding to 165x1024x768 voxels where each voxel corresponded to the size 5x5x5 nm. After taking reference from the shared paper, a patch size of 512 was chosen as the optimal.

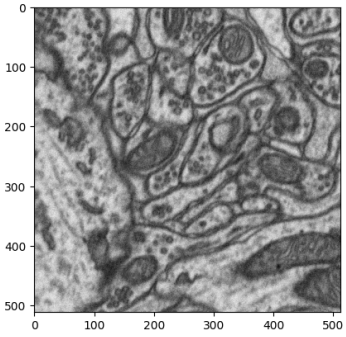

Figure 1: The sample image slices extracted from the dataset

3. Preprocessing and dataloading

Data augmentation techniques, like horizontal flip, vertica flip, and gaussian noise were used. It was observed that rotation or such augmentation techniques would lead to artifacts around the edges while training. A train-validation split of 80:20 was made. The data was augmented to both training and validation set.

4. Building the model

My initial attempts included using 3D UNet to segment the 3D volumes. This was inefficient and led to images being saturated. The model would fail to generalize and the training would be slow. The 3D UNet also suffered from insufficient dataset as 3D volume was quantitatively lesser.

Upon advice, I then switched to 2D UNet to segment the 3D volumes. This was easier to train and would train and learn the training features well. But this would fail to generalize on the validation set. The validation set would become all black/white. I tried various variants of the UNet architecture. I tried changing the number of feature layers, number of encoders, adding Res-Net like skip layers, different loss and learning rate, but training the model from scratch proved to be arduous and time consuming. The models were training, but tuning the parameters to generalize over the validation set was difficult across the implementations.

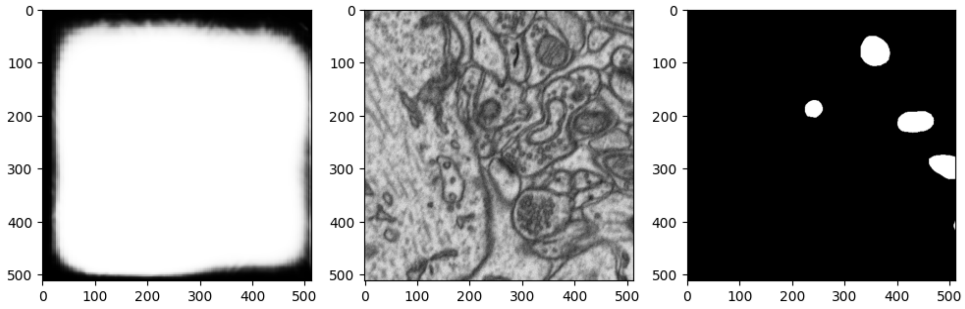

Figure 2: The sample validation set output which was saturated during training

Hence I started to look towards transfer learning approaches such that I could use pretrained weights to extract the features and use that as the encoder/compressive path of the UNet. The decoder would then correspond to the individual convolution blocks of the encoder. For the purpose of the project I chose ResNet34 as the encoder model. The base architecture was UNet. This replaced the encoder/compressive path with ResNet34. This enables the U-Net to use the high-level characteristics acquired by the ResNet network while also gathering fine-grained spatial information. The pretrained weights of ResNet are freezed which enable faster convergence and feature extraction. The decode network

layer weights are learned via back propagation. Due to positive sample imbalance, I decided to use modified DiceLoss which is better suited for skewed data.

Parameter	**Value**
Encoder	ResNet34
Number of Epochs	70
Optimizer	Adam
Learning Rate	0.001
Loss	DiceBCELoss= DiceLoss+BCELosswithLogits

Table 1: Model Parameters

5. Results

IoU/Jaccard's index and DiceBCELoss are the metrics used for computing the model performance.

Jaccard's index or IoU (Intersection over union) is used to evaluate the performance of image segmentation. The IoU metric is based on the overlap between the predicted and ground truth segmentations. The Io U is calculated as the ratio of the number of pixels in the intersection of the predicted and ground truth segmentations to the number of pixels in their union. Mathematically, it can be expressed as:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

Equation 1: Intersection over Union

DiceLoss is the loss function used to score ranges from 0 to 1, with a value of 1 indicating a perfect overlap between the predicted segmentation and the ground truth segmentation. The DICE score is calculated as follows:

$$\mathbf{L}_{dice} = \frac{2 * \sum p_{true} * p_{pred}}{\sum p_{true}^2 + \sum p_{pred}^2 + \epsilon}$$

Equation 2: Dice Loss

DiceBCELoss is the sum of DiceLoss and BCEWithLogitsLoss

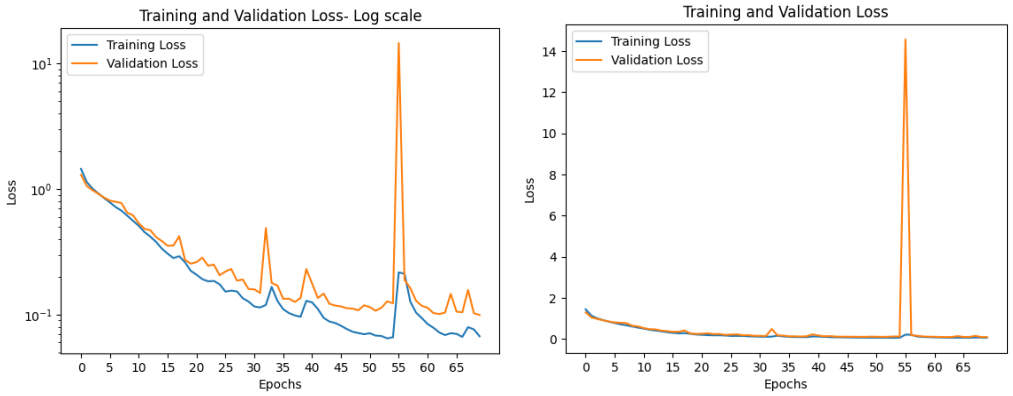

Figure 3: Loss curves. Loss vs Epochs for 70 epochs

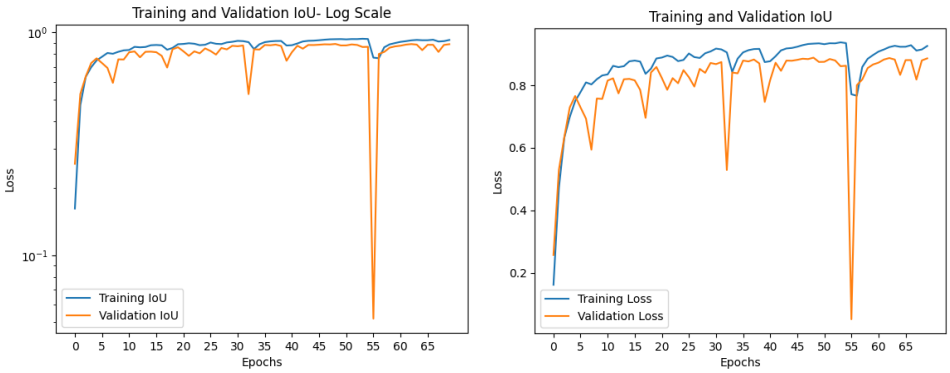

Figure 4: IoU curves. IoU vs Epocs for 70 epochs

	DiceBCELoss	Jaccard's Index/IoU
Training Set	0.164	0.934
Validation Set	0.678	0.882
Testing Set	0.542	0.713

Table 2: Losses, IoU for training, validation and testing set

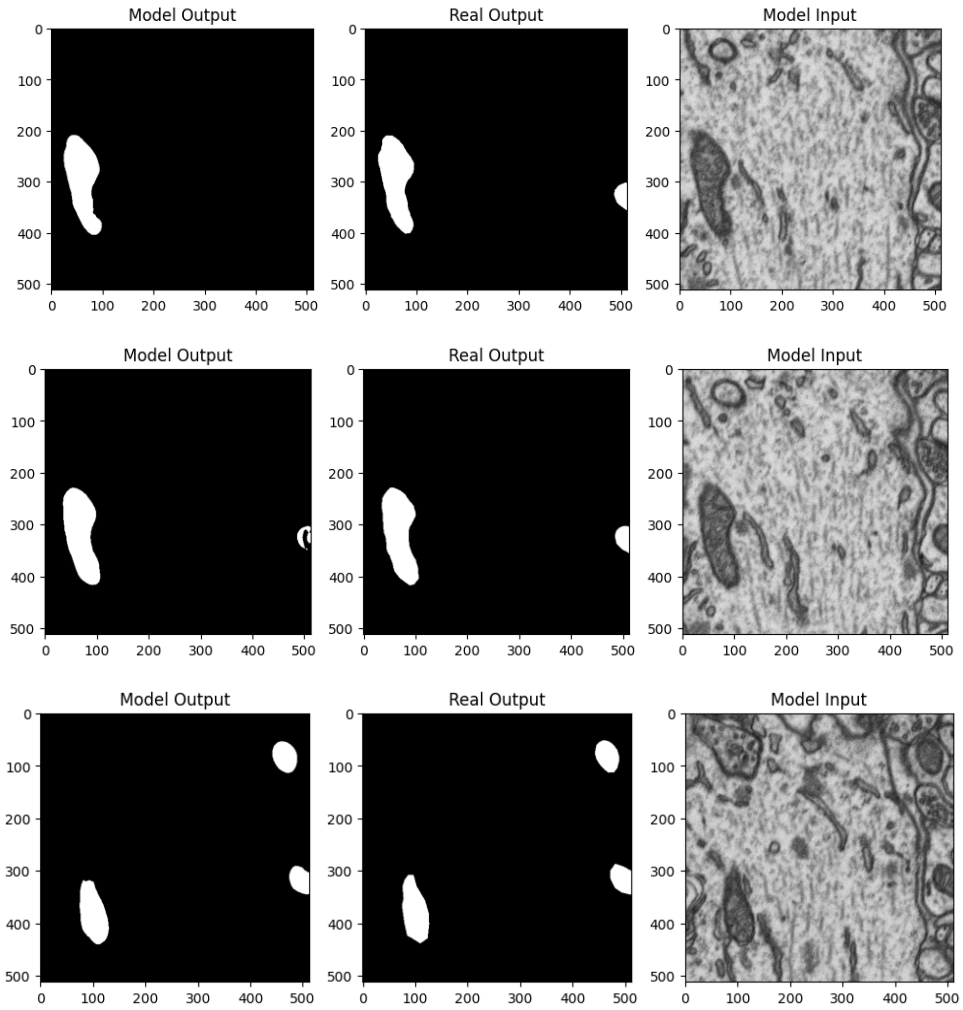

Figure 5: Sample inference over testing set. Image on the left is the

6. Conclusion

The use of transfer learning leads to a good application specific development of models that can give quick and accurate results. The application of ResNet34Unet has shown promising results. The model converges quickly and is able to show good results across the datasets. Training the model from scratch leads to loss of generalization and was time consuming, hence transfer learning worked better for this application. The Code for the architecture used can be found.

7. Discussion

The task took considerable time because of the initial approach to building models. 3D images would take considerably longer to train and lead to lesser data. It proved to be difficult to faithfully learn the features of 2D images using 2D UNet as the model always

failed to generalize. The IoU would arbitrarily drop to zero in the middle of the training. Adding data augmentations would be giving better results, but adding training overhead. Eventually, effective training over the collab file proved worrisome. Also, while training the model from scratch, hyperparameter tuning was time consuming, because both feature extraction and decoder network weights would be learned simultaneously. The loss of generalization might have been due to the use of DiceBCEloss with adam. Along with the inability to find the adequately complex model for the amount of training set available. Which always led to underfitting/overfitting or large training time.