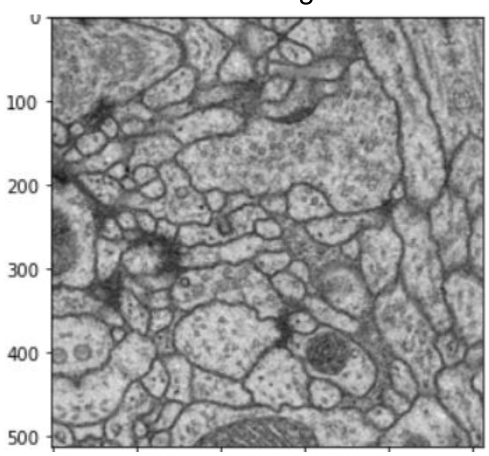
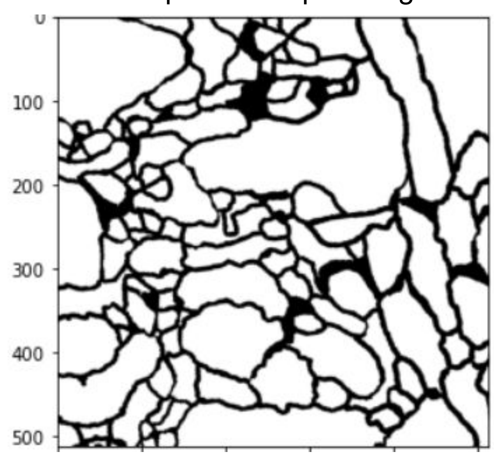
MIC Project Report
Neuronal Segmentation from Electron Microscopy Image

Neel Aryan Gupta 180050067
Tathagat Verma 180050111

INPUT image



Expected output image



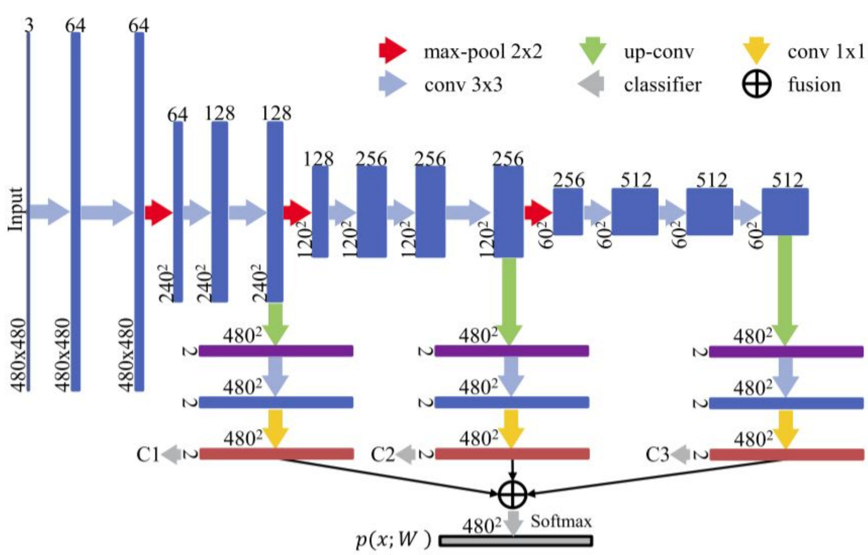
Dataset used - ISBI 2012 cell segmentation challenge - 30 images size 512x512 - entire dataset is present in the python notebook submitted

Two methods used:

1. Deep Multilevel Contextual Network link to research paper
2. U-Net (used with variations) link to research paper

<u>Deep Multilevel Contextual Network (DMCN)</u>

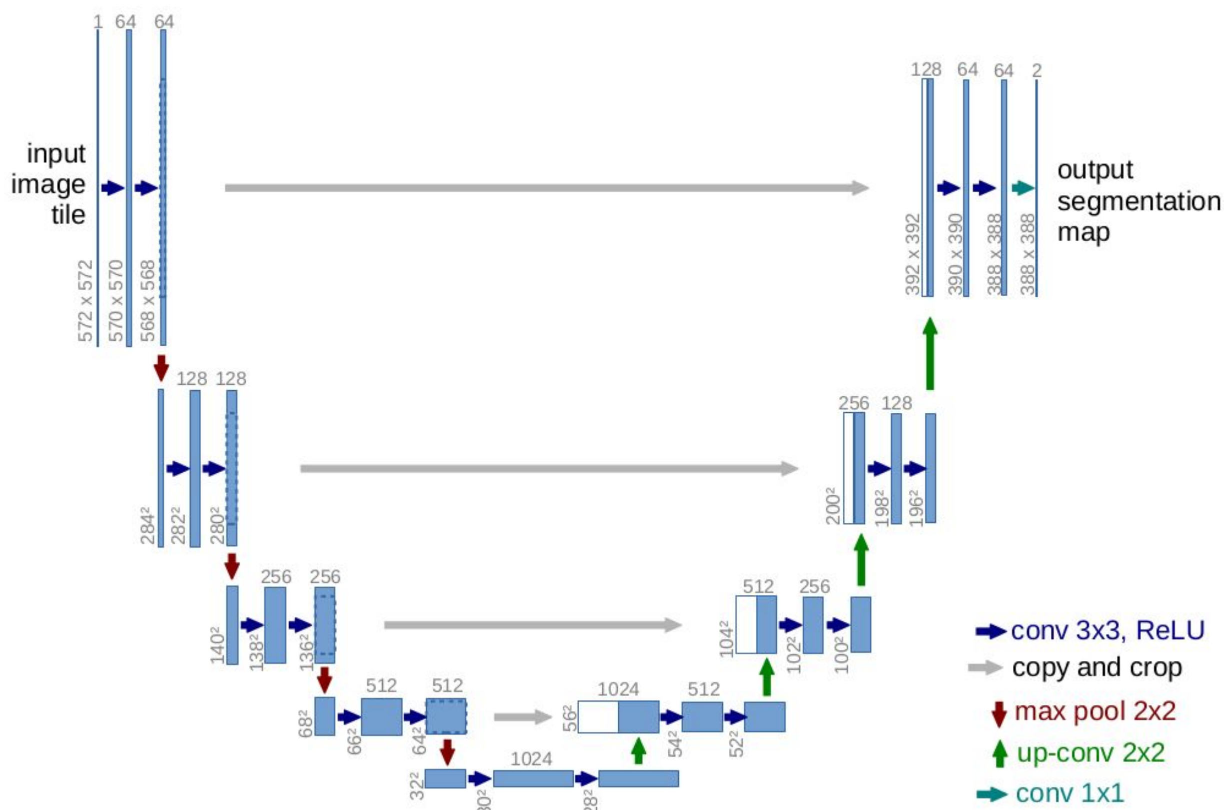
Model architecture



- This model has 2 modules essentially
    - Downsampling with max-pooling and convolutional layers
    - Upsampling with convolutional and de-convolutional(backward strided convolution) layers
- Abstract and global information from higher layers helps in classification and local information from lower layers helps in localization accuracy like identifying boundaries
- ReLU activation used after each layer
- Overall we have used 16 convolutional layers, 3 max-pooling layers for downsampling and 3 de-convolutional layers for upsampling
- Finally this multilevel contextual information is fused with a concatenation operation and fed into a classification layer
- 7.7 million parameters and all being trainable

U-Net

Model Architecture



Variations used in U-nets

- Unet1
    - no weight maps (explained later), with no dropouts
    - ‘valid’ padding used hence input size 512x512, output size 388x388
    - used overlap-tile strategy
    - average and resize methods (to get same sized output image as input), explained later
- Weighted Unet
    - used weight maps, and dropouts
    - ‘same’ padding used model input size 512x512 and output also 512x512
- Unet2
    - no weight maps, ‘same’ padding and dropouts
    - reduced number of filters in convolutional layers reducing number of parameters from 31M to 8.6M to reduce training time and overfitting possibility

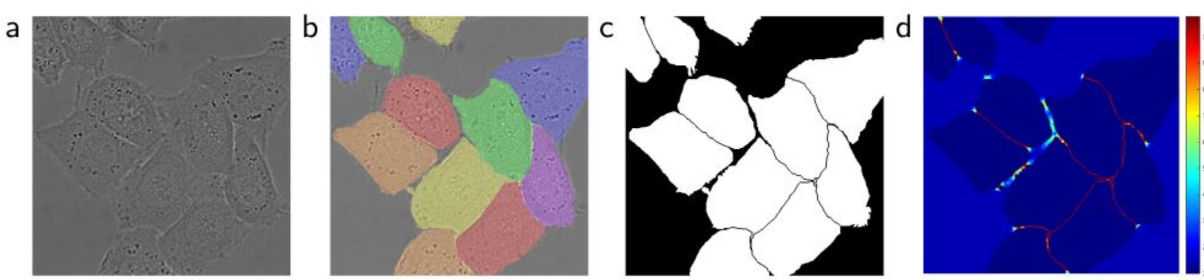
Weight Maps

In cell segmentation, one challenge is separation of touching objects of same class

For this we use weighted loss where separating background labels between touching cells obtain large weight in loss function. Using weight maps forces network to learn the border pixels

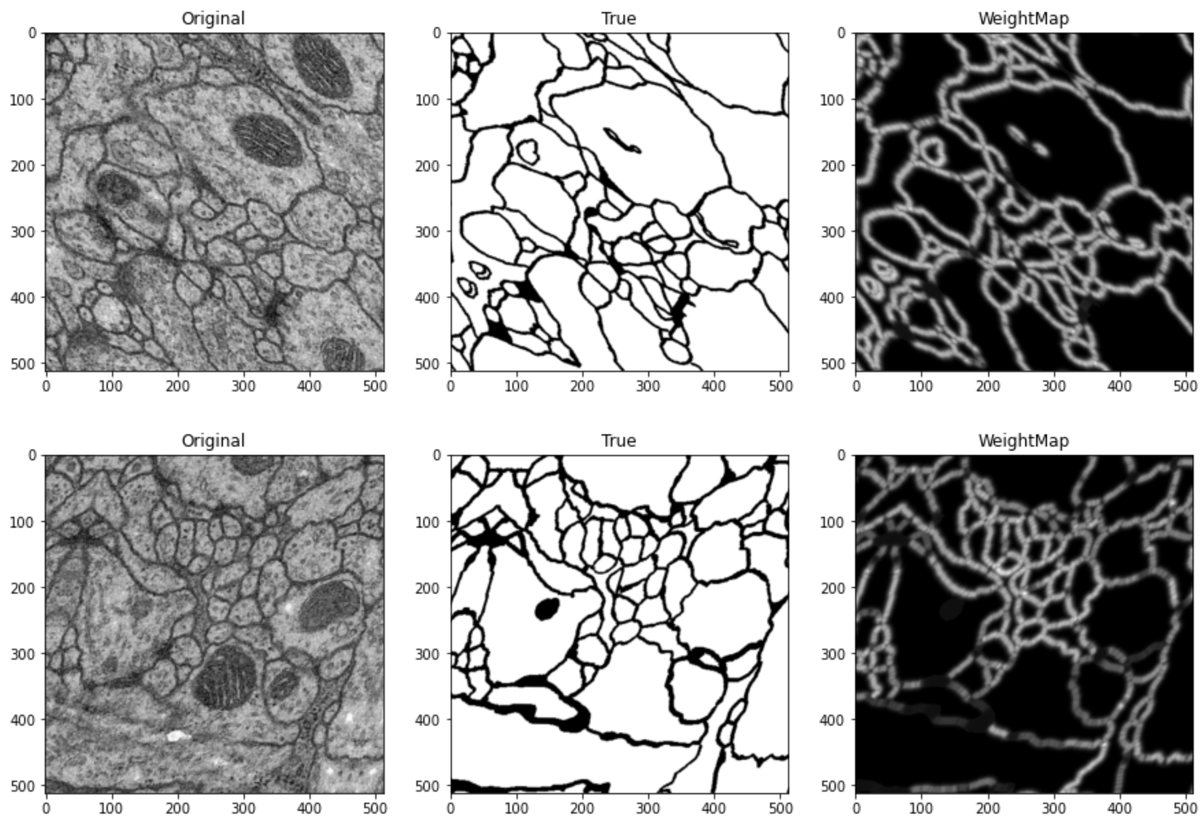
$w_c : \Omega \rightarrow R$ is the weight map to balance the class frequencies, $d1 : \Omega \rightarrow R$ denotes the distance to the border of the nearest cell and $d2 : \Omega \rightarrow R$ the distance to the border of the second nearest cell

$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right)$$



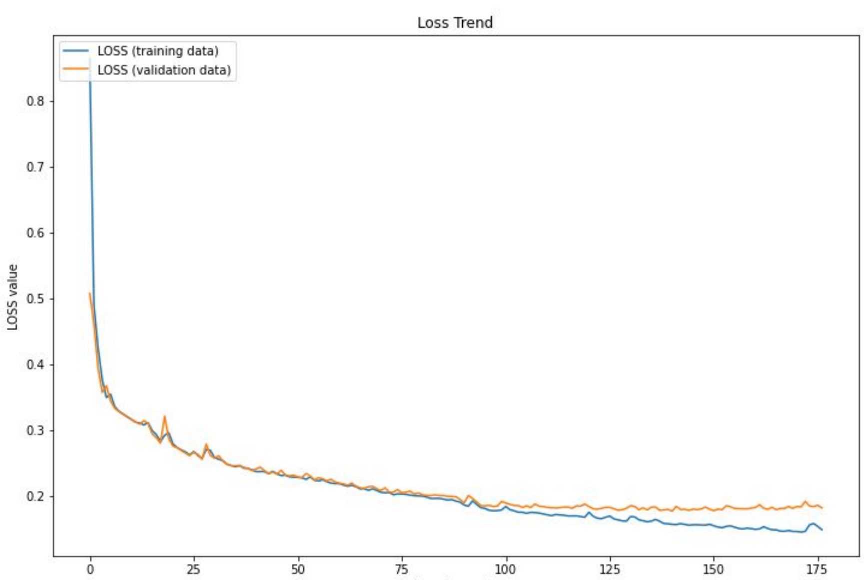
Data Visualization

Original and True images given in the ISBI dataset. WeightMap generated by us

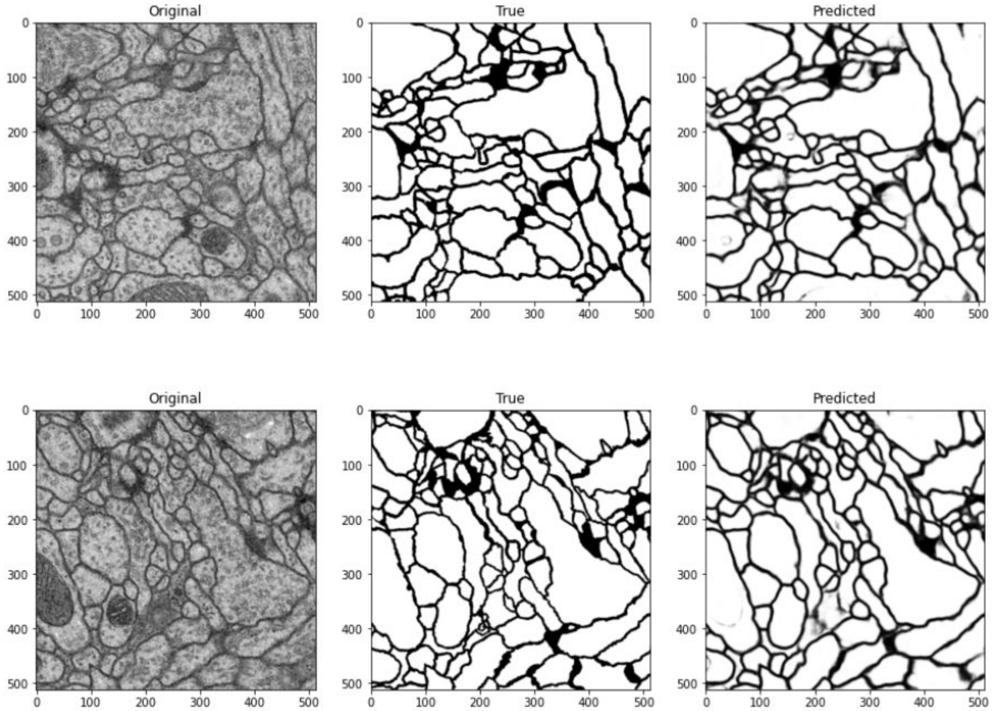


Results

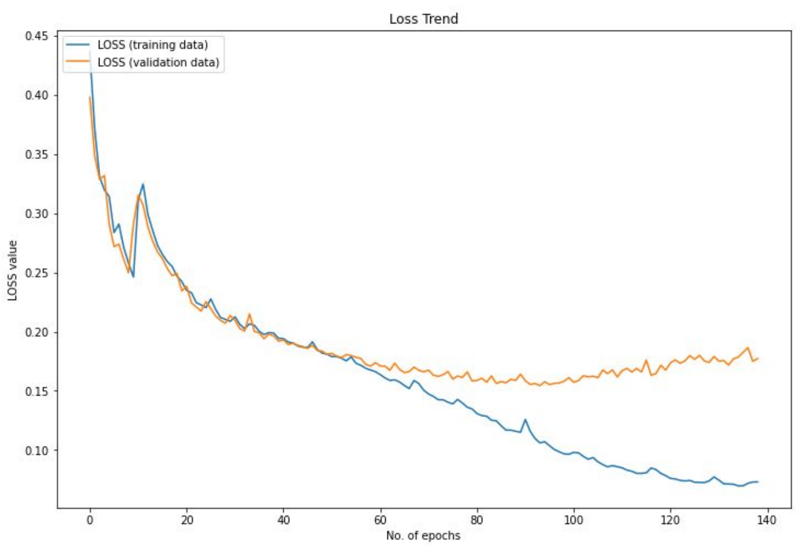
1. DMCN



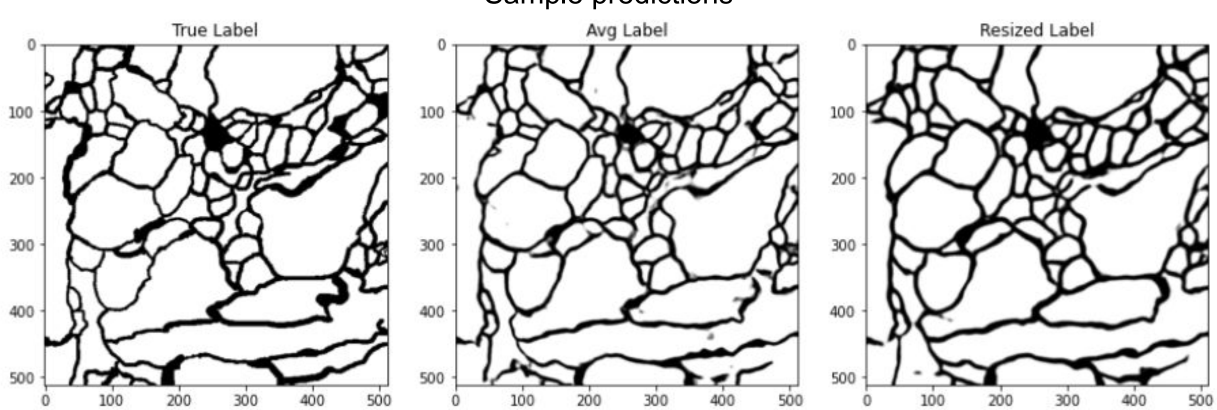
Sample predictions



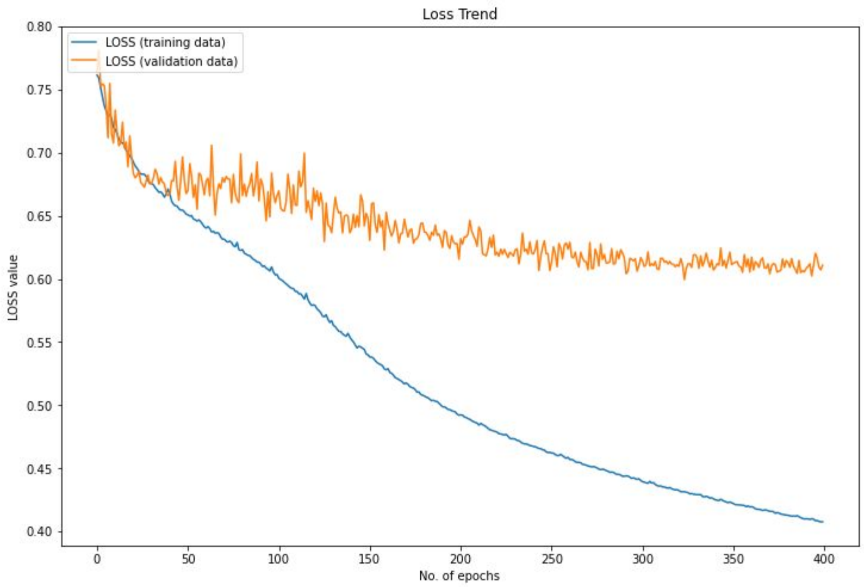
2. Unet1



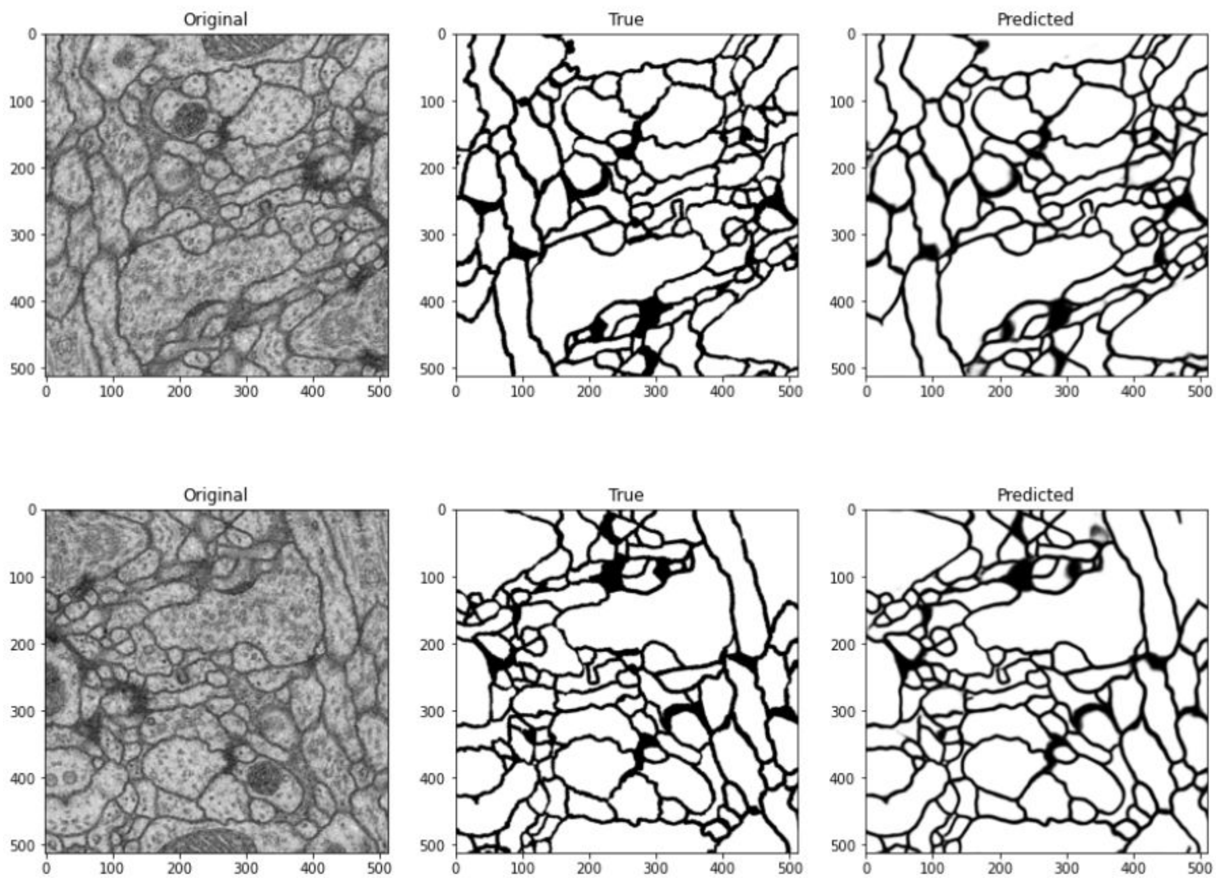
Sample predictions



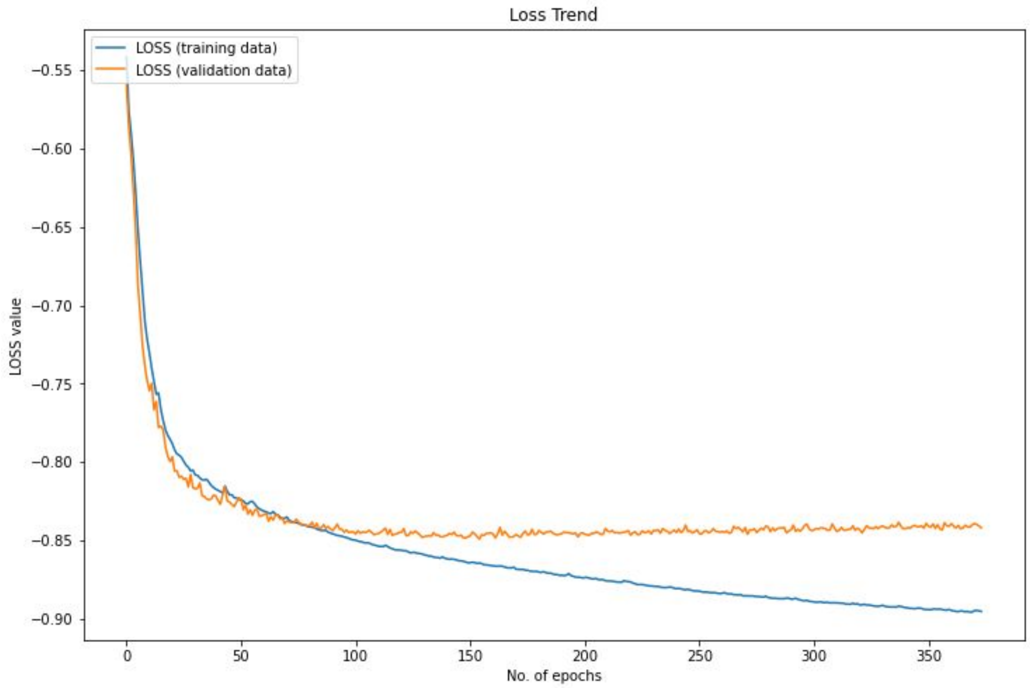
3. Weighted Unet



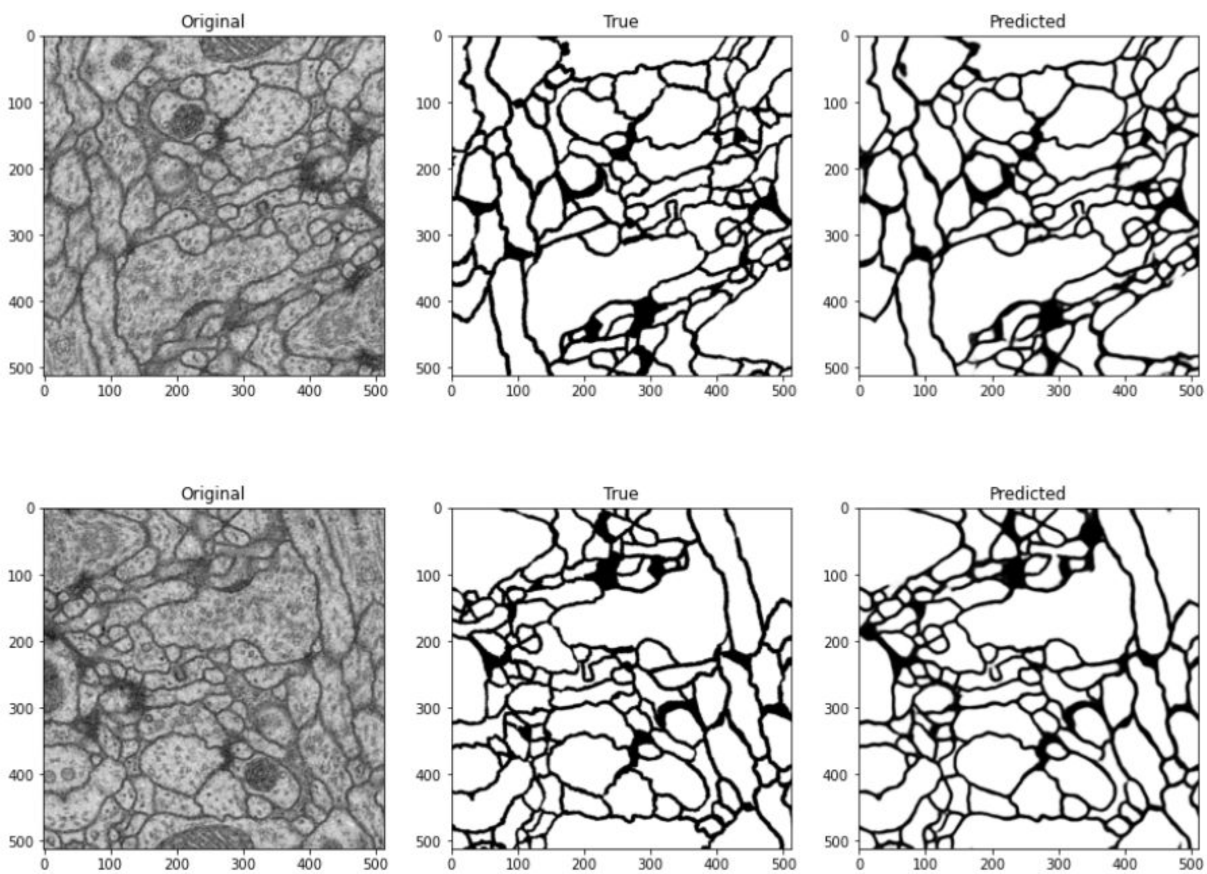
Sample predictions



4. Unet2



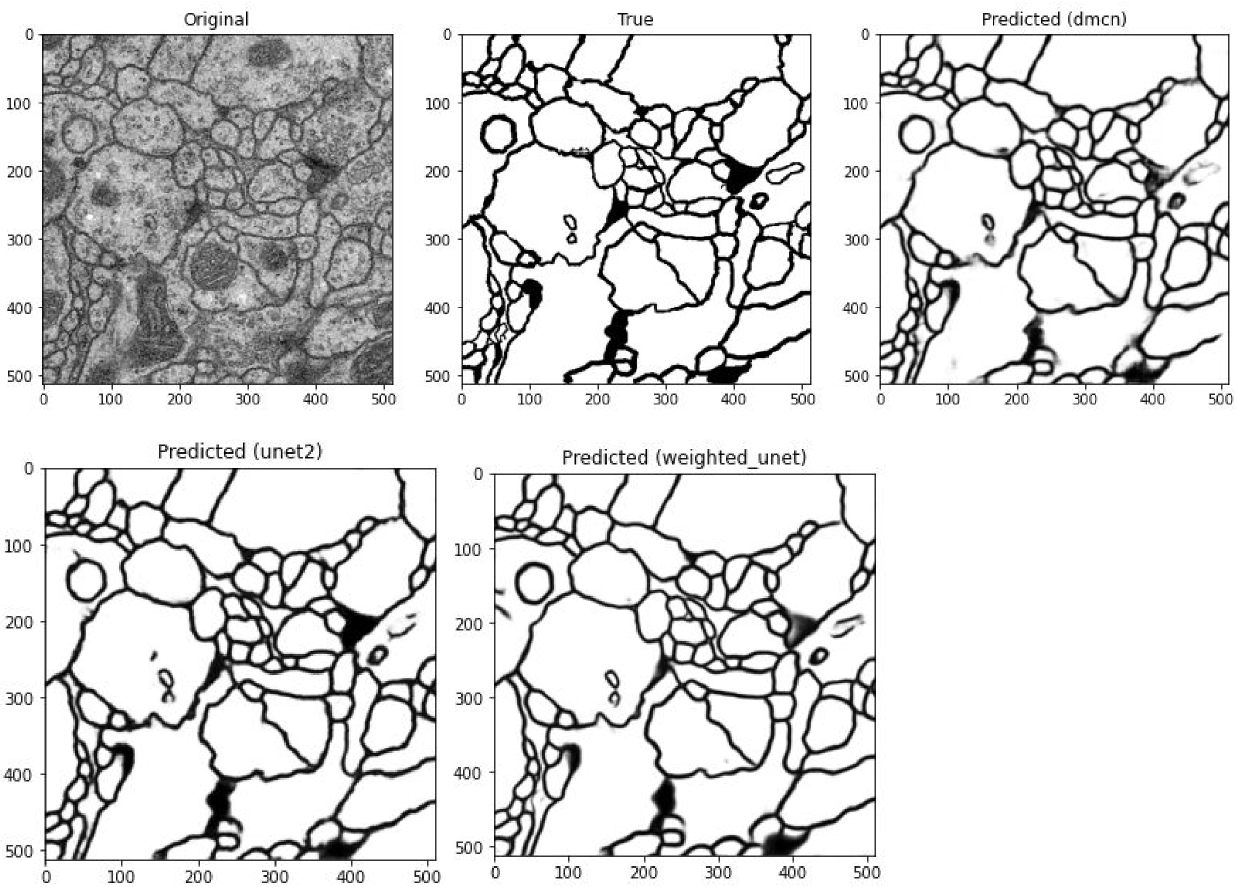
Sample predictions

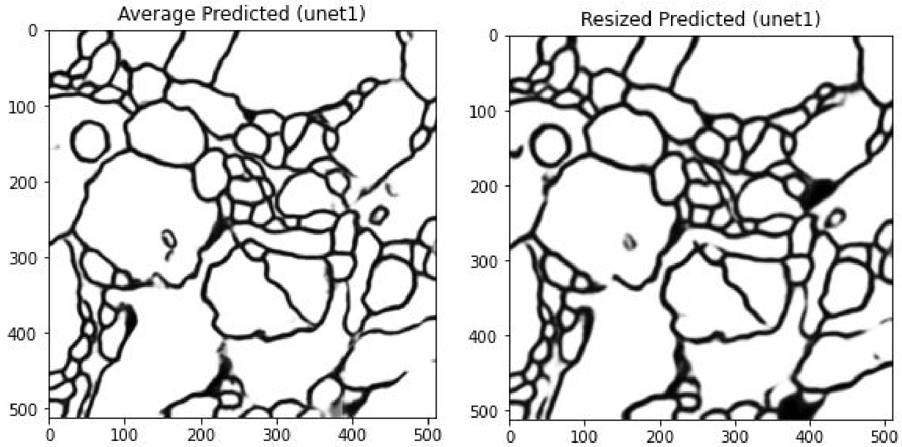


Result Metrics

	IOU score (Jaccard index)	DICE score (F1 Score)	Pixel error	Rand error	No. of params (millions)
DMCN	0.8760	0.9528	0.0732	0.1355	7.7
weighted_unet	0.9020	0.9617	0.0599	0.1124	31
unet1	0.8980	0.9623	0.0578	0.1087	31
unet2	0.8840	0.9556	0.0690	0.1282	8.6

Predictions of all models on a common image





Conclusion

- ‘valid’ paddings are better than ‘same’ paddings because we are able to use the contextual/local information to classify each pixel
- Introduction of weight maps result in improvement of performance because they force the model to learn the border pixels, and takes into account class imbalance