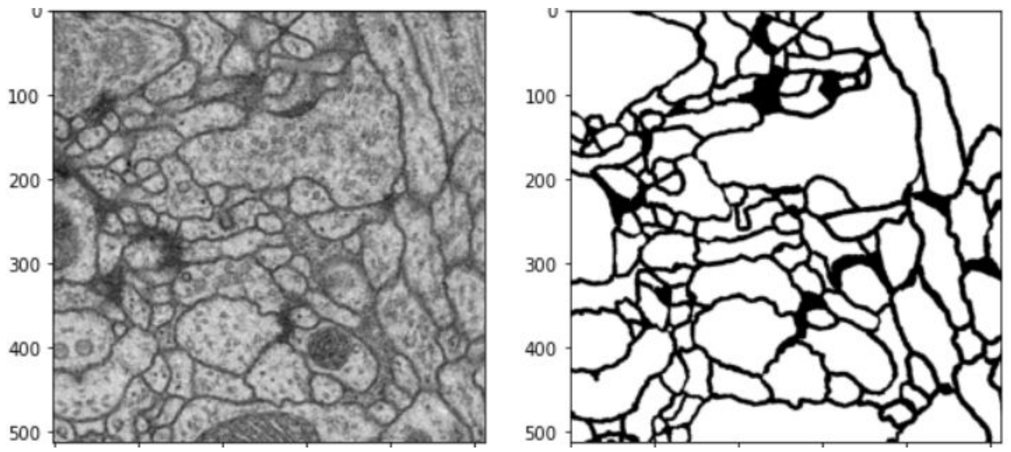
Neuronal Structure Segmentation in Electron Microscopy images

Neel Aryan Gupta 180050067
Tathagat Verma 180050111

Introduction



Why is this a problem ?

1. Image deformation during acquisition can blur membrane boundaries
2. Neuron membrane thickness can vary a lot, ranging from solid dark curves to grazed gray swaths
3. Presence of intracellular structures make edge detection and region growing based methods ineffective to identify neuron membranes

Where is this useful?

This segmentation is used in neuronal circuit reconstruction which is used in functional analysis of brain and other nervous systems

Strategy

We broadly use 2 deep learning models with variations

1. Deep Multilevel Contextual Network
2. U-Net
   a. Weight maps
   b. Dropout nodes
   c. Patch method

1. Chen, H., Dou, Q., Qi, X., Cheng, J. Z., & Heng, P. A. (2020). Deep multilevel contextual networks for biomedical image segmentation. In *Handbook of Medical Image Computing and Computer Assisted Intervention* (pp. 231-247). Academic Press.

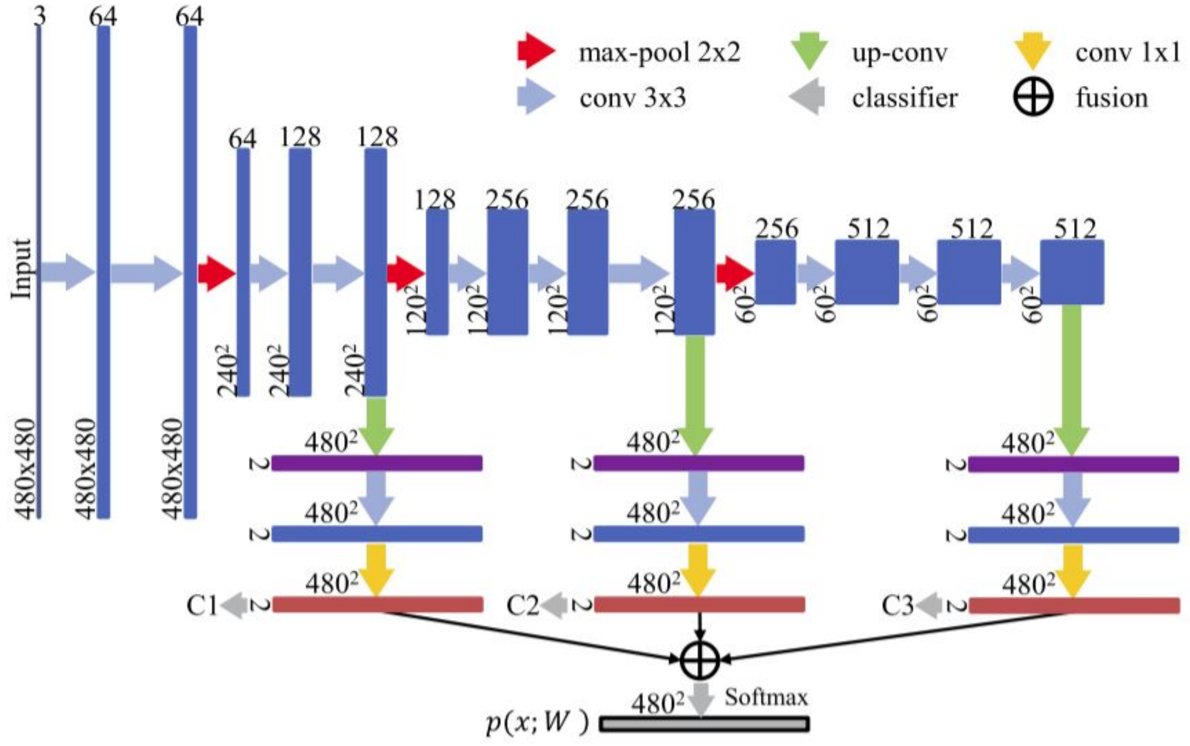
2. Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.

Deep Multilevel Contextual Network

- 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

Deep Multilevel Contextual Network

Model architecture

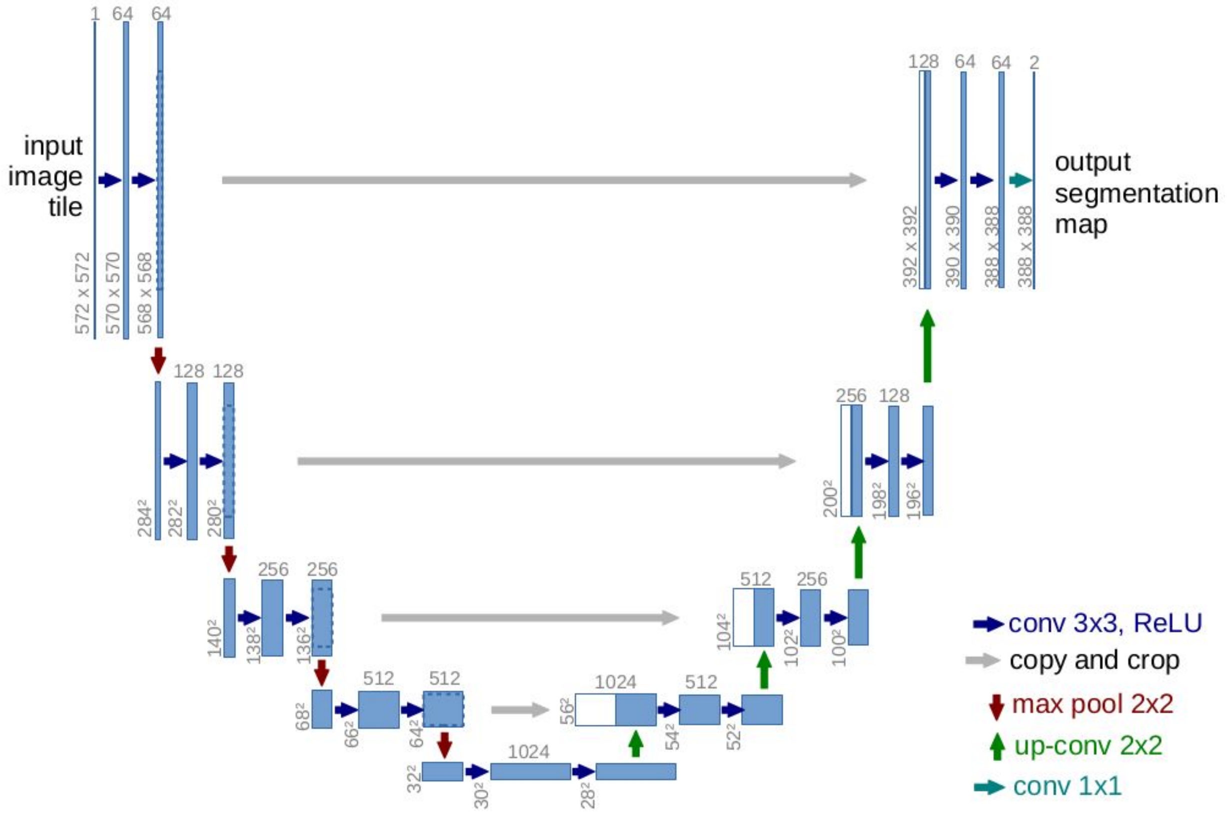


U-Net

- General approach is using sliding-window method, for each pixel use a patch surrounding it to determine its label
- Benefits of the sliding window approach
  - The network can localize
  - Training data is much larger than training images
- Drawbacks of the sliding window approach
  - Network is slow as it runs for every patch and redundancy due to overlapping patches
  - Trade-off between localization accuracy and use of context
- In U-Net, we have downsampling and upsampling operators which are able to use context as well as local information and use feature channels to propagate context information to higher resolution layers
- 23 convolutional, 4 max-pooling, 4 up-convolution layers used in total

U-Net

Model architecture



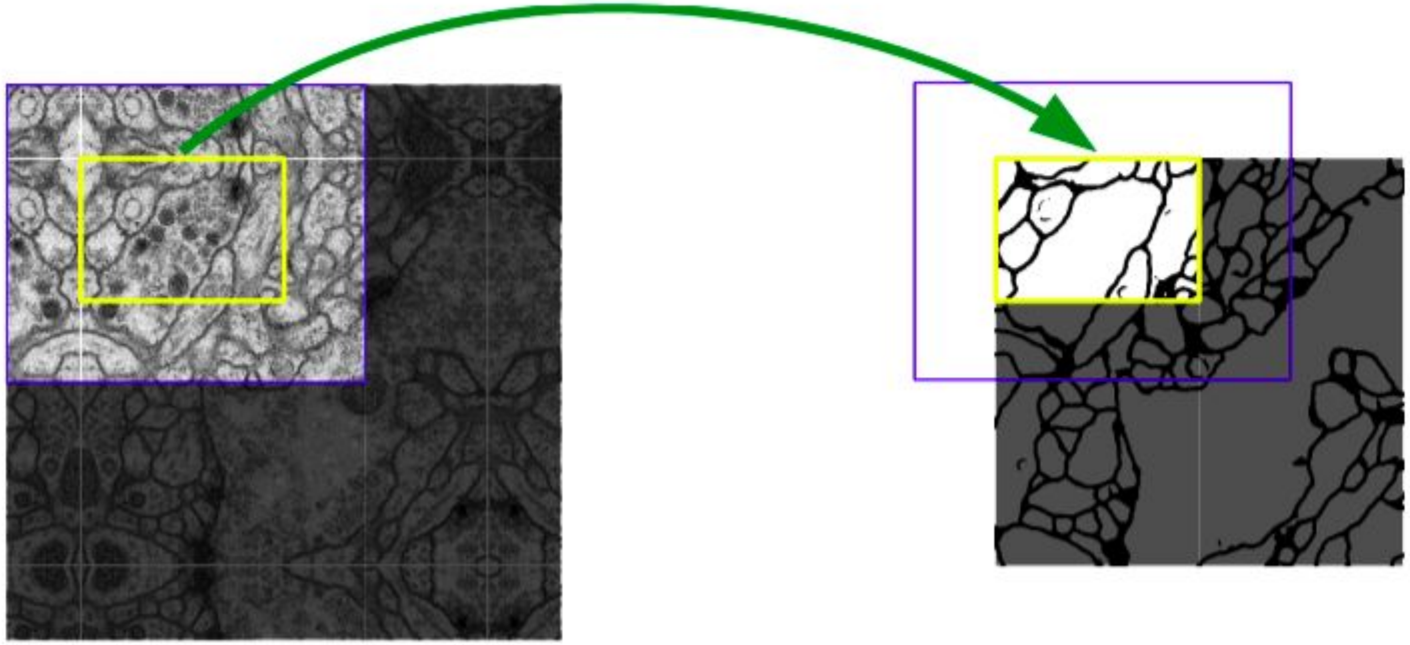
U-Net

Some key points

- 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

U-Net

Overlap Tile strategy



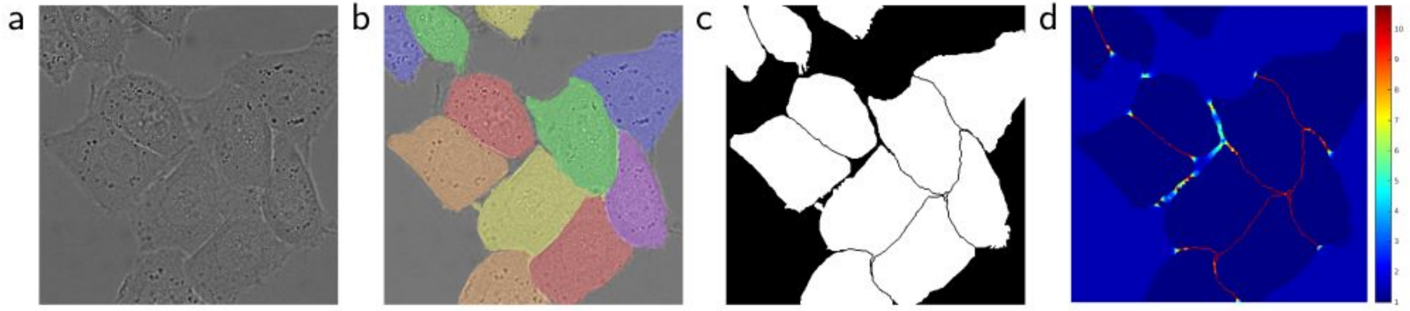
Weight Maps

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

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

w_c : Ω → R is the weight map to balance the class frequencies, d1 : Ω → R denotes the distance to the border of the nearest cell and d2 : Ω → R the distance to the border of the second nearest cell



Implementation details

Dataset - 2012 ISBI neuronal structure segmentation challenge. 30 images 512x512 for training

Data Augmentation - created 1000 images by making random transformations such as zoom, rotations, flips, etc.

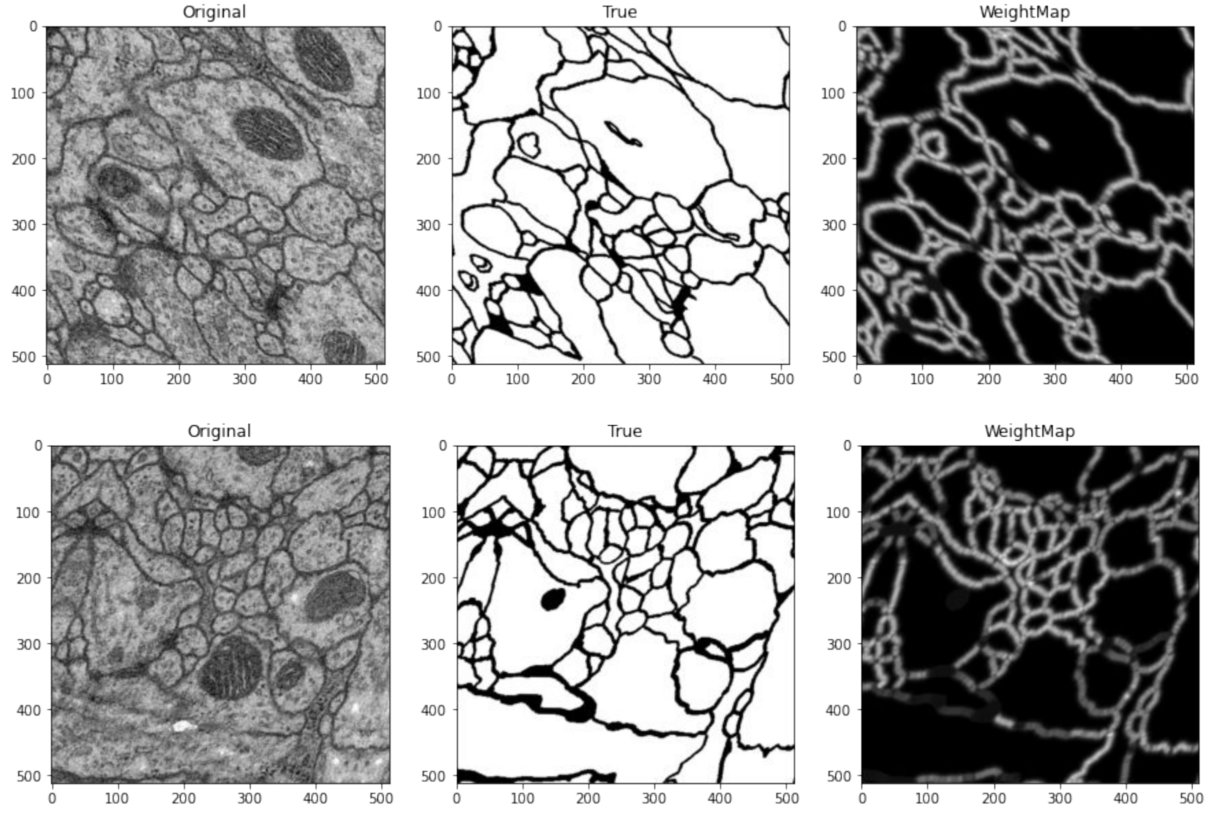
Used Keras library for the models and data augmentation

Used He normal initialization for the layers activated with ReLU - draws samples from a truncated normal distribution centered on 0 with `variance = 2 / fan_in` where `fan_in` is the number of input units in the weight tensor

Used Glorot uniform initialization for the layers activated with sigmoid and softmax - draws samples from a uniform distribution within `-limit, limit` where `limit` is `sqrt(6 / (fan_in + fan_out))` where `fan_in` is the number of input units in the weight tensor and `fan_out` is the number of output units in the weight tensor

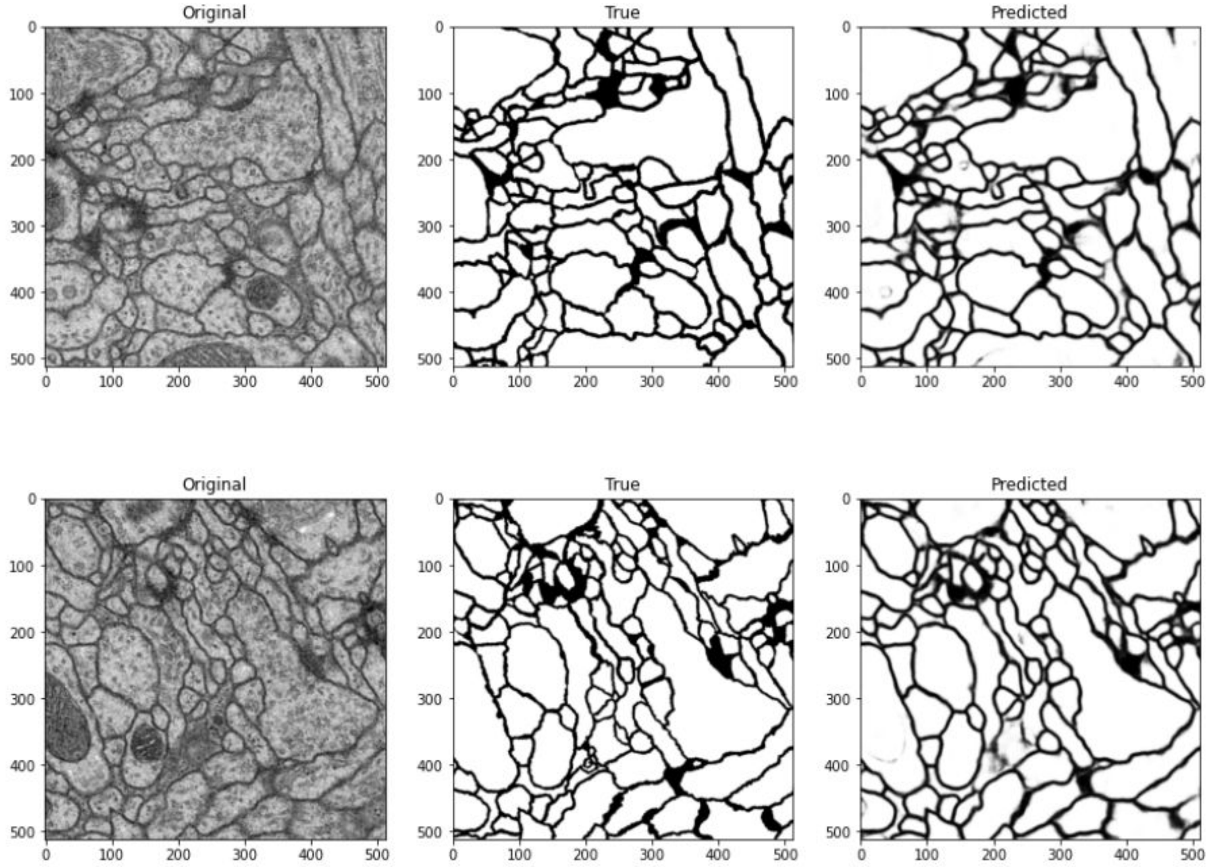
Data Visualization

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



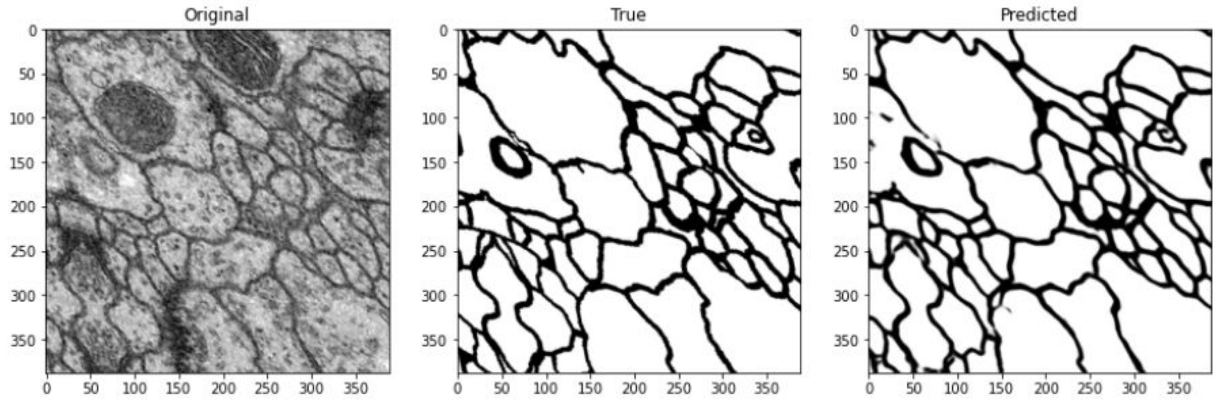
Results

Predictions from the DMCN model

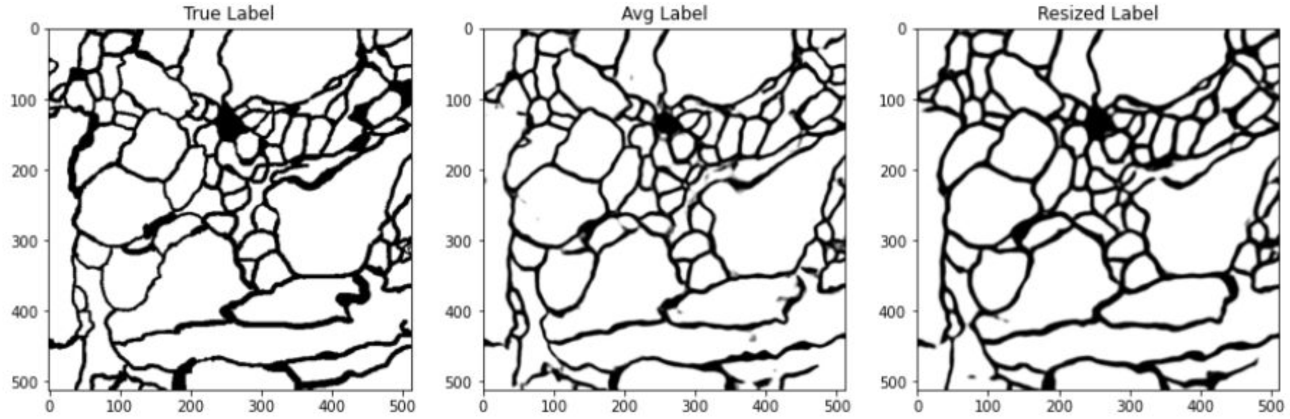


Results

Predictions from the U-Net1 model



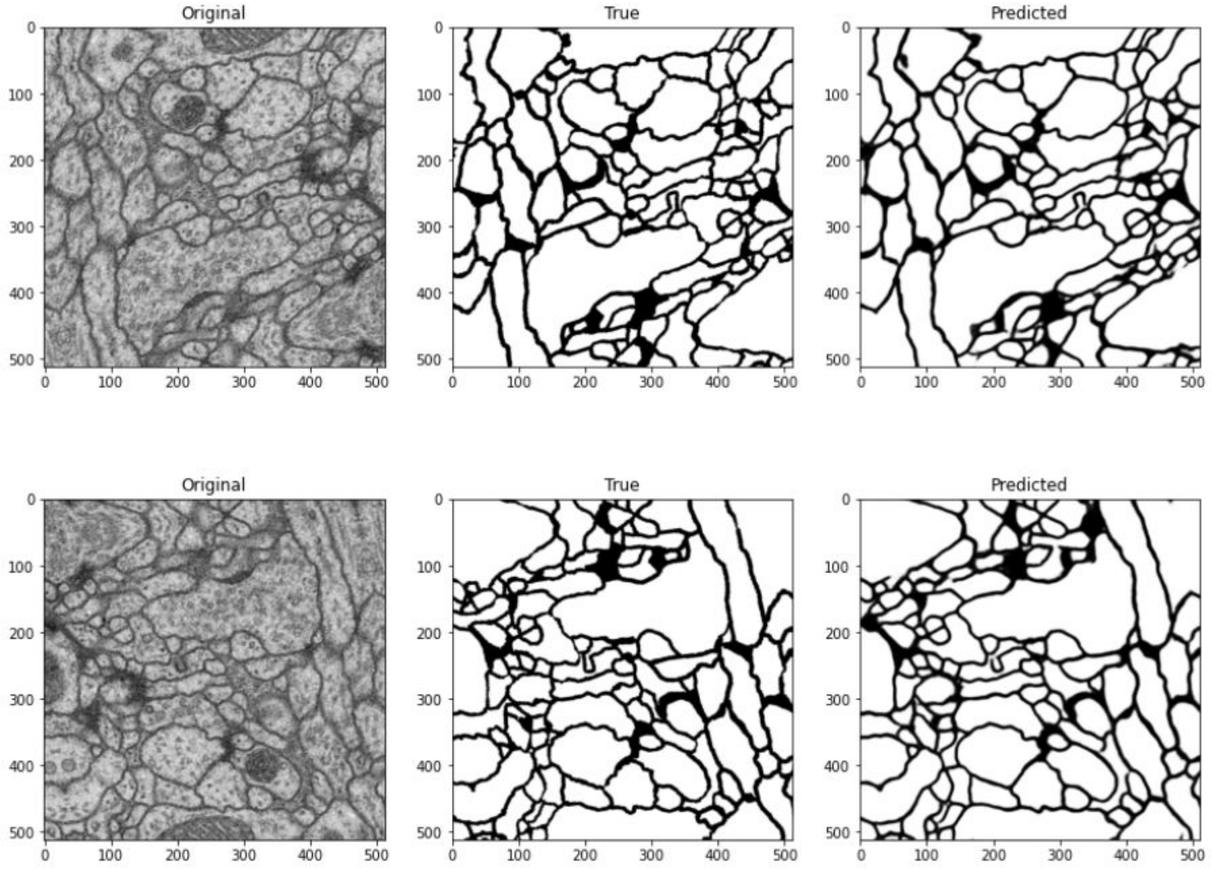
Prediction on 388 x 388 patch



Prediction on original image 512 x 512 using average and resize methods

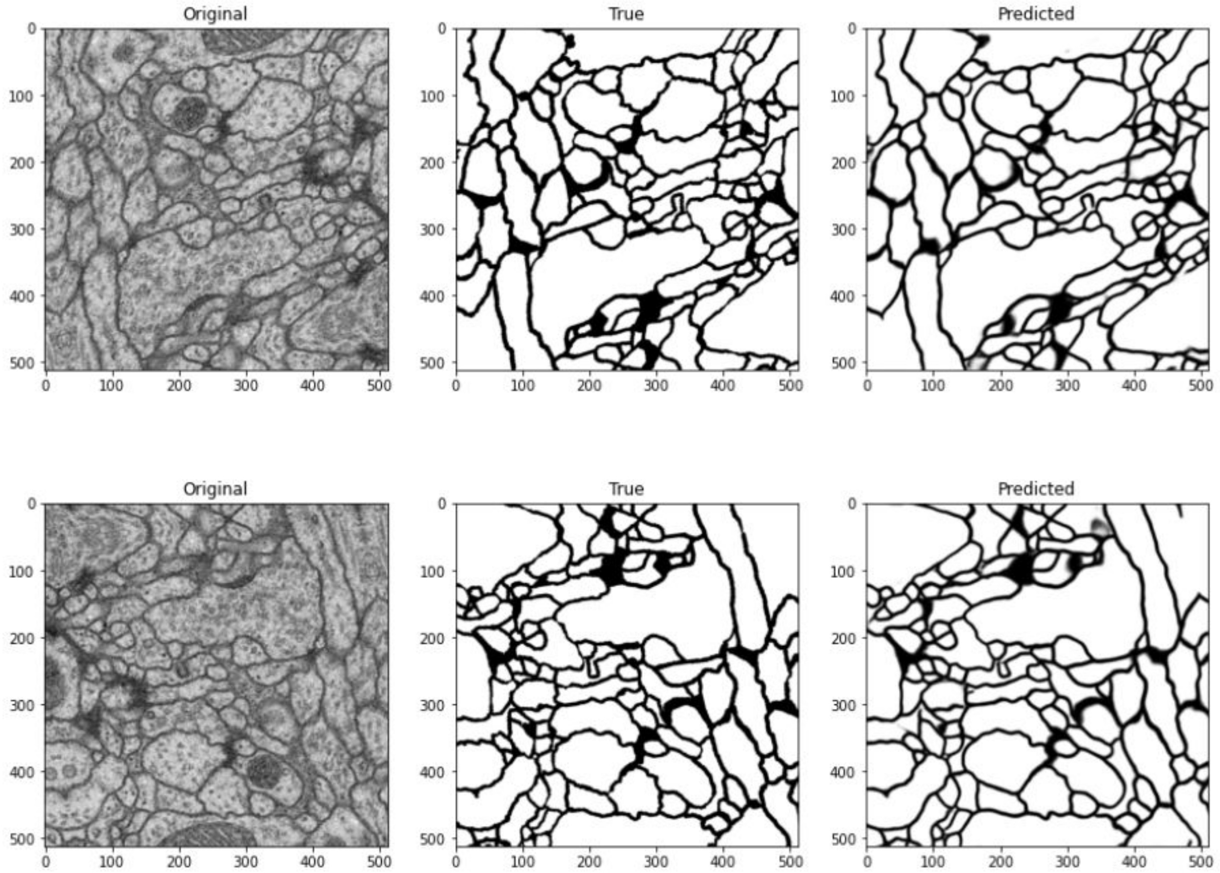
Results

Predictions from the U-Net2 model



Results

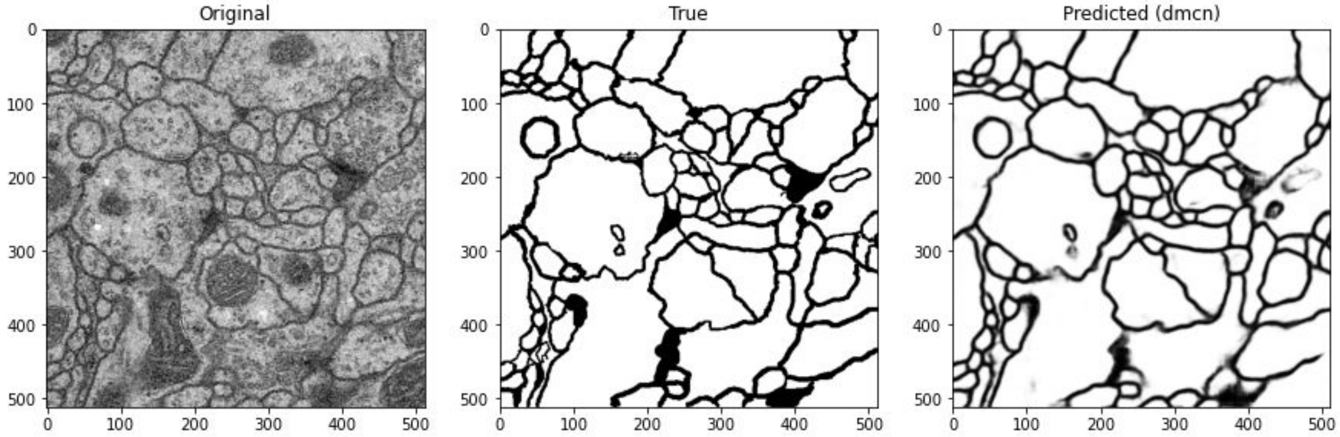
Predictions from the weighted U-Net model

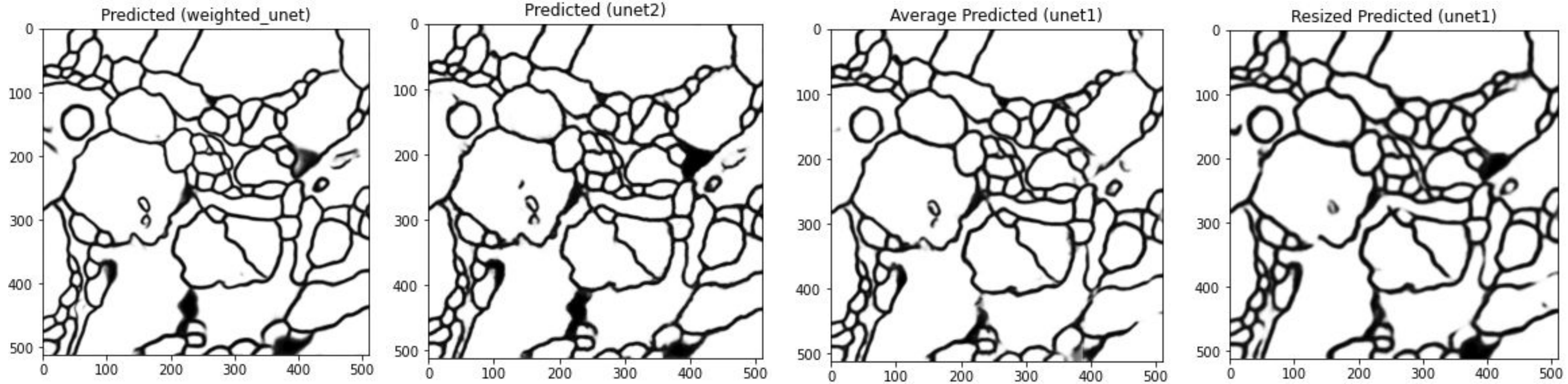


Results

	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

Results


Original
True
Predicted (dmcn)
0
100
200
300
400
500


Predicted (weighted_unet)
Predicted (unet2)
Average Predicted (unet1)
Resized Predicted (unet1)
0
100
200
300
400
500

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