Outline:

Data source

Otsu and concavity for comparison

CNN crop method description

Data generation

Model Structure & Training

Classification

Loss function method and structure

Tie everything together

[1]

D. C. Ciresan, L. M. Gambardella, and A. Giusti, “Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images,” p. 9.

This section provides an in-depth description of the methodology and procedure used to segment and differentiate pericyte and smooth muscle cells in blood brain barrier neurovascular microscopy scans. It outlines the acquisition of the data, a baseline shallow-learning approach, and a deep neural network approach which uses a convolutional neural network, sliding window approach, as presented in [Ciresan].

**Data**

First, labelled data was obtained from Dr. Yongsoo Kim containing 120 512x512 images, along with segmented black and white masks containing separately labelled pericyte and smooth muscle cell features. The images were captured using serial two-photon tomography and are greyscale. They are all manually labelled by a neuroscience expert. 20 images were randomly chosen as test data, and 100 were kept for training data.

**Baseline**

To provide a baseline for the experiment, it was necessary for the sake of result evaluation to develop a baseline segmentation method that used no deep learning. For the sake of simplicity, techniques outlined in [Sharma] were predominantly analyzed. Ultimately, a combination of thresholding and connected component area analysis was used. These techniques are outlined in [Sharma] and [Zhou]. Specifically, Otsu thresholding was used, followed by a connected component analysis removing features below 10 pixels in area and above 250 pixels in area. These techniques yielded a set of preliminary images segmented to identify both smooth muscle cells and pericytes in the dataset. This segmentation served as a baseline of comparison for the deep methods described shortly.

**CNN**

The deep neural network implemented to identify pericyte and smooth muscle cells in the dataset will be structured in the following ways. The model is written using Python, with the Keras library and the Tensorflow backend. The model has the following layers:

* 2D convolution, kernel size: 3, stride: 3, activation function: relu, input size: 65x65
* 2D max pooling, size (2x2)
* 2D convolution, kernel size: 3, stride: 3, activation function: relu
* 20% dropout
* 2D convolution, kernel size: 3, stride: 3, activation function: relu
* 20% dropout
* Dense layer, activation function: sigmoid, output: probability that a cell exists in the center pixel of the input image (between 0 and 1)
* Loss function: Binary crossentropy

The methodology of the model can be deciphered from its structure. The model takes as input an 65x65 greyscale image, in the form of a tensor (numpy array). The model uses dropout to avoid overfitting [Ronneberger]. The model uses max-pooling to reduce data size but still retain important features from a large initial data size [Ciresan]. The final layer is a dense layer with a sigmoid activation function, along with a binary crossentropy loss function, because the model is being trained on a binary decision. The objective of the model is to determine whether or not the central pixel in the image is part of a region of interest in the image, as determined by the ground truth segmented images. This is the approach taken by [Ciresan] and implemented with success. In other words, the model determines whether or not the central pixel of the input image is a pericyte or smooth muscle cell. The model is trained using the same set of images, but different sets of masks, and thus can identify pericytes and smooth muscle cells together, and distinguish between them. The idea behind this model is that the small image size provides the model with the capability to process a large amount of data, along with allowing it to process many images quickly. Additionally, the center-pixel identification strategy allows the CNN to learn to use the context around the center pixel to determine if that center pixel contains a cell, just as the human eye would [Ciresan]. This model allows the small amount of data to be increased by only training on a small portion of each image, and still allows the CNN to distinguish patterns and features by learning them.

**Data Preprocessing**

The source images are 512x512 greyscale, and the ground truth masks are very sparse in terms of the amount of identified, segmented cells.

***INSERT DATA EXAMPLE HERE***

Due to the sparse nature of the data, the data generation and preprocessing must be done in such a way that maximum features can be extracted and learned. Thus, choosing a random 65x65 patch and training on it is not optimal, due to the fact that there is a low probability that this fragment will contain a cell. Thus, an approach was taken akin to [Ciresan]. Every pixel in every cell in the ground truth image set was used to generate a 65x65 image for training in the positive category. Subsequently, random sections of the training dataset portion not containing cells in the center pixel were chosen for the negative category such that the positive and negative examples were equal. In instances where the center pixel lies within the window size (65x65) from the edge of the image, the image is mirrored in that area to create the slice, as outlined in [Ciresan].

The model was trained for each cell type over 1000 epocs with a batch size of 25 images. The model was evaluated using a custom loss function, which is described in the following subsection. This loss function is designed to maximize cellular feature extraction, and not emphasize exact cellular accuracy.

**Data Augmentation**

In [Ciresan], Ciresan et al. had success with random 90 \degree rotations on images when generating training data. The data in this experiment were augmented accordingly. Additionally, [Ronneberger] proposed various augmentation techniques in their CNN paper. These augmentations are vital to generating a large amount of training data from a small training dataset. Thus, to generate an even larger amount of potential data, these techniques, including shifting and random elastic deformations were also used to augment the training data.

**Accuracy/Loss Function**

A custom loss function was written to evaluate and train the CNN model, and also to evaluate the baseline thresholding performance. This loss function is also used to compare the performance of the various segmentation techniques used to segment the data images into pericytes and smooth muscle cells. The loss function calculates the F-score of a segmentation image and its ground truth mask by calculating the false positive, true positive, and true negative rate as follows. The algorithm first locates the connected pixel areas in both images. Each of these connected pixel areas is considered a ‘cell’, with a location.

**True Positive:** Cell in generated segmentation image with at least 1 pixel from this cell touching a cell in the ground truth image.

**False Positive:** Cell in generated segmentation image that does not have at least 1 pixel from this cell touching a cell in the ground truth image.  
**False Negative:** Cell in ground truth image that does not have at least 1 pixel from this cell touching a cell in the generated segmentation image.

F-score is calculated as usual, by calculating and using precision and recall [Macawile]:

***TODO: INSERT F-SCORE FORMULA, PRECISION AND RECALL FORMULA***

The idea of this accuracy/loss function is that, rather than calculating pixelwise accuracy, this function calculates cellwise accuracy. As long as the segmentation method generates a feature that is overlapping with the desired feature it is locating a cell that is, at the very least, extremely close to the right classification area. This is useful for two reasons: 1) Ultimately, the application of this model is in the counting of cells in certain regions of the brain, and 2) Ultimately, even when trying to locate specific cell locations, it is better to classify all cells very close to their proper location than to classify only some cells with exact pixel-by-pixel accuracy, because exact pixel-by-pixel accuracy is impossible due to variance in expert opinion and, thus, variance in human ground truth. Therefore, a very close approximation is always desirable in cell segmentation, so this loss function was crafted and implemented to evaluate performance based on this fact.

In conclusion, a shallow and deep learning approach was taken to identify cellular features in brain microscopy scans, and a custom F-score cell-wise loss function was used to evaluate the accuracy and usefulness of these features.