ANNOTATION-FREE DEEP LEARNING OF LARGE-SCALE NUCLEAR SEGMENTATION AND SPATIAL NEIGHBORHOOD ANALYSIS ON MULTIPLEXED FLUORESCENCE IMAGES

by
Xiaoyang Li

A Dissertation submitted to the Electrical and Computer Engineering Department,
The Cullen College of Engineering
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in Image Processing and Data Analysis

Chair of Committee: Dr. Badrinath Roysam

Committee Member: Dr. Hien Nguyen

Committee Member: Dr. Saurabh Prasad

Committee Member: Dr. Jason Eriksen

Committee Member: Dr. Dragan Maric

University of Houston
December 2020

ACKNOWLEDGEMENTS

National Institute of Health funds the research of this thesis.

First, I would like to express my deepest appreciation to my PhD advisor Dr. Badri Roysam for his continuous guidance. His solid experience and extensive knowledge motivated my academic progress. His serious attitude to science, tremendous enthusiasm for research, and endless patience with instructions positively impacted my whole life.

I would also like to extend my deepest gratitude to the committee members. Dr. Hien Nguyen provided me with invaluable contributions throughout the duration of the zero-human effort nuclear segmentation project. Dr. Dragan Maric was instrumental in advising my internship at NINDS, NIH, and my thesis projects, including watershed segmentation and image registration. Dr. Saurabh Prasad brought his valuable suggestion into my neighborhood analysis project. I have also learned a lot from his series of machine learning courses. Dr. Jason Eriksen advised me on the side of biological applications.

I am so grateful to be surrounded by the fun and smart spirits in my study place because of my labmates Dr. Jahandar Jahanipour, Dr. Hengyang Lu, Aditi Singh, and Rachel Mills. Many challenges cannot be conquered without their thoughtful inspirations and encouraging motivations. I also appreciate my senior labmates Dr. Kedar Grama, Dr. Yan Xu, Dr. Murad Megjhani, Dr. Leila Saadatifard, and collaborators Dr. Dash Pramod, Dr. Amit Amritkar, Dr. Navin Varadarajan, Pengyu Yuan and Dr. Aryan mobiny.

My sincere appreciation also goes to Dr. Ping Lu and Dr. Nikolaos Mitsakos, Yufei Yuan, to instruct on my summer intern industrial projects that largely extended my diverse industrial AI application experiences. I also appreciate my hiring manager Dr. Geoffrey Oxholm at Adobe, for offering me the research engineer position after I graduate.

Many thanks to my friends Dr. Li Huang, Dr. Jiabing Li, Dr. Dandan Zhu, Rufeng Li, Jiajie Chen, Davar Daeinejad, Dr. Jesus Cruz-Garza and Dr. Zhenxin Yan. Their encouragement helped me get through many hard times, and they made my past five years full of joy and memories. And I mourned my good friend Charles Lee who lost his young life in a car accident. He let me know that one can have a limited life but an unlimited colorful experience.

Special thanks to my boyfriend, Dr. Alex Joachim. He is not only a hero as a first responder working in the healthcare system during the Pandemic, but also a gift to my life, providing me the endless support and encouragement.

Last but not least, the completion of my study would not have been possible without spiritual support from my most precious family. My father, as a good example of a researcher, encouraged me to keep making progress. And my mother, taught me to be optimistic even in the most difficult time. I also hope my Grandma can stay healthy and happy as she always does.

ABSTRACT

Deep neural networks (DNNs) offer state-of-the-art performance for cell nucleus detection and segmentation. However, they require many manual annotations from skilled biologists for robust algorithm training, which is labor-intensive and not easily scalable. We propose an unsupervised expectation driving pipeline for Brain Cell Analysis using noisy Labels with minimal human input. 1) It uses a parametric method to generate noisy labels for cell nuclei and refines through an iterative training process. 2) We introduce a background recovery technique to enhance the detection and estimation of segmentation accuracy, especially in densely packed brain regions. 3) A novel sparse decomposition method is used to identify anomalous cell detections and automatically correct them to improve the accuracy further. We also provide extensive experiments evaluated on both supervised and unsupervised measurements to demonstrate our method's high effectiveness. The results of segmentation can be further used for phenotyping and cell localization. Besides, we proposed a spatial model to analyze the neuron-glia cells neighborhoods by cumulative influence of all neuron-glial pairs from the same circular surrounding area centered at the neuronal nuclei. A fast co-location analysis is applied to profile cell spatial neighborhoods in the healthy brain efficiently. LASSO-based feature selection methods are adopted to reveal their changes in different tissue conditions. Finally, we developed an accurate, fast-speed, and scalable method to align large-scale images from multi-round to pixel-level accuracy.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS i
ABSTRACT iii
TABLE OF CONTENTS iv
LIST OF TABLES viii
LIST OF FIGURES ix
CHAPTER 1 INTRODUCTION 1
1.1 Experiment Preparation 1
1.2 Descriptions of the Dataset 4
1.3 Objectives 4
CHAPTER 2 AUTOMATIC NUCLEAR SEGMENTATION BY WATERSHED . 6
2.1 Automatic Nuclei Segmentation 6
2.1.1. Image Filling 8
2.1.2. Seed Detection 8
2.1.3. Watershed Segmentation 9
2.2 Results 10
2.3 Conclusions 13
CHAPTER 3 EXPECTATION DRIVEN NUCLEAR SEGMENTATION 14
3.1 Introduction 15
3.2 Preliminary Studies 16
3.2.1. Background Review 16
3.2.2. Train Deep Neural Networks with Label Noise 17
3.2.3. Mask R-CNN Model 18
3.3 Proposed Methods 20
3.3.1. Iterative Training 21

3.3.2. Background Recovery for MRCNN Detection 23
3.3.3. Cell Splitting and Phenotyping using Sparse Decomposition 25
3.3.4. Large Scale Image Merging 31
3.4 Evaluation Measurements 33
3.4.1. Supervised Measurements 33
3.4.2. Unsupervised Measurements 35
3.5 Performance 39
3.5.1. Dataset Setting and Run Time Summary 39
3.5.2. Discussions on Background Recovery 40
3.5.3. Performance on Supervised Measurements 42
3.5.4. Performance on Unsupervised Measurements 45
3.5.5. Visualization Results on Large Scale Datasets 45
3.6 Conclusions 47
3.7 Future Works 48
CHAPTER 4 PHENOTYPE 49
4.1 Phenotyping from Segmentation Sparse Decomposition 49
4.2 Classification with Deep Learning Networks 50
4.3 Interactive Gating 50
4.3.1. Flow Cytometry Standard 50
4.3.2. Image Cytometry Experiment 51
CHAPTER 5 CELLULAR NEIGHBORHOOD COLOCATION ANALYSIS 54
5.1 Introduction 55
5.1.1. Neuron Neighborhood Biological Background 56
5.1.2. Dataset Description 57
5.1.3. Boolean Spatial Features 60
5.2 Measure the Attraction of Neuron Neighborhoods 60

5.2.1. Construct Graphical Representation for Local Neighborhoods 61
5.3 Colocation Rule of Neuronal Neighborhood 63
5.3.1. Association Rule 63
5.3.2. Colocation Rule 64
5.4 Result Visualization 66
5.5 Conclusions 68
CHAPTER 6 CELLULAR NEIGHBORHOOD CORRELATION ANALYSIS 69
6.1 Introduction 70
6.1.1. Related Works 70
6.1.2. Challenges 72
6.1.3. Contributions 72
6.2 Graphical Representation of Neuron neighborhood 73
6.2.1. Hierarchical Structure of Neuron Neighborhood 73
6.2.2. Limitations of Common Feature Representations 74
6.2.3. Measure the Attraction of Neuron Neighborhood 76
6.3 Supervised Feature Selection with Respect to Different Classes 78
6.3.1. Using LASSO to Select Features 79
6.3.2. Using Sparse Group-LASSO to Select Features 80
6.3.3. Using Tree-LASSO to Select Features 82
6.4 Feature Selection Measurements 84
6.4.1. Jaccard Similarity Measure 85
6.4.2. Spearman's Rank Correlation Coefficient 86
6.5 Results Evaluations 87
6.5.1. Discussions on the KNA Parameters 88
6.5.2. EXPERIMENT 1: Neuron Status Comparison 89
6.5.3. EXPERIMENT 2: Tissue Type Comparison 92

6.5.1. EXPERIMENT 3: Brain Side Comparison 96
6.6 Conclusions 98
6.7 Future Work 99
CHAPTER 7 LARGE SCALE MULTI-PLEX IMAGE REGISTRATION 101
7.1 Previous Methods 102
7.2 Basic ORB Based Image Registration Pipeline 103
7.2.1. Feature Detection and Extraction 104
7.2.2. Feature Matching 106
7.2.3. Transformation Estimation 107
7.3 Improvements on Large Scale Texture Featured Image Registration 108
7.3.1. Generate Tile Ranges 108
7.3.2. Tile-based Feature Detection and Extraction 111
7.3.3. Tile-based Feature Matching 112
7.3.4. Tile Sampled Hierarchical RANSAC 114
7.3.5. Bootstrapped Regional Refinement 116
7.4 Experiment Discussion 118
7.4.1. Multiprocessing Acceleration 118
7.4.1. Other Parameter Settings 119
7.5 Conclusions 120
REFERENCES 122
APPENDIX 139
A. Loss Functions of MRCNN 139
B. Figures for MRCNN Modules 140
C. Recommended System Requirements 141
D. Loss Performance of MRCNN 142

LIST OF TABLES

Table 2-1. Detection Performance on Crops of Cortex Regions. 12
Table 2-2. Improvement of Shape Fidelity in Whole Brain. 13
Table 3-1. Supervised Performance Metrics Comparisons..... 43
Table 3-2. Unsupervised Performance Metrics Comparisons 45
Table 5-1. Boolean Table of 10 Cell Phenotype Channels..... 59
Table 5-2. Boolean Spatial Features 60
Table 5-3. Illustration of Typical Models of Co-Location 65
Table 6-1. Common Neighborhood Measurement and their Limitations..... 75
Table 6-2. Comparing the Neighborhood Attraction with Two Other Measurements..... 78
Table 6-3. An Example of Indexes and Ranks of LASSO Coefficients Result..... 85
Table 6-4. Evaluations of the Feature Selection Results 91
Table 6-5. Stability and Computational Cost of 10 Top Feature Selection Results 95
Table 7-1. Multiplex Image Registration Parameters..... 120

LIST OF FIGURES

Figure 1-1. Healthy Rat Brain Sample in UT Health. 1
Figure 1-2. Tissue Staining in the Wet Lab at NINDS, NIH. 2
Figure 1-3. Microscopy Imaging in the Dry Lab at NINDS, NIH. 2
Figure 1-4. Whole Rat Brain Tissue from Multiplex Imaging. 3
Figure 1-5. Close-up Images from Five 10-plex Imaging Rounds. 3
Figure 2-1. The Pipeline for Automatic Segmentation 7
Figure 2-2. Visualization of Image Filling For Segmentation. 8
Figure 2-3. The Comparison of Watershed Segmentation with Compactness Constraint in A Cropped Region of Hippocampus. 10
Figure 2-4. The Comparison of Nucleus Detection on DAPI Channel and Composite Channel. 11
Figure 3-1. Diagram of Mask RCNN 18
Figure 3-2. The Detailed Structure of MRCNN For an Example of Nuclei Image. 19
Figure 3-3. Fully Automatic Pipeline for Brain Cell Nuclear Segmentation 20
Figure 3-4. Background Recovery Example for an Image Patch. 24
Figure 3-5. Sparse Decomposition of Nuclear Masks. 26
Figure 3-6. Example of a Nucleus Segmentation Split into Two Nuclei. 28
Figure 3-7. Cell Type Mask Generation Examples 29
Figure 3-8. Examples of Nuclei Segmentation Split into Different Cells. 30
Figure 3-9. Large Scale Segmentation Solution. 31
Figure 3-10. Intersection-over-union 33
Figure 3-11. Raw Image and Splitting Cells. 37
Figure 3-12. Examples of Low Confidence Objects. 38
Figure 3-13. An Example of Background Boosting Stopping at Step Three. 41
Figure 3-14. Cell Detection Plots as a Function of Background Recovery Iterations. 41

Figure 3-15. Segmentation Result Comparisons. 42
Figure 3-16. Performance on Iterative Training for Three Image Samples. 43
Figure 3-17. Overall F1score-IoU Curve 44
Figure 3-18. Testing Results on Healthy Rat Brain 46
Figure 3-19. Testing Results on Alzheimer Dataset 46
Figure 4-1. Phenotype Result of Neuron Seeds Superimposed on the NeuN Channel. . 49
Figure 4-2. Visualize the FCS Outputs on Kaluza Software. 51
Figure 4-3. Neuron Phenotyping on a Cropped Image by FCS express. 52
Figure 4-4. Whole Brain Image Interactive Gating. 52
Figure 5-1. Neurons and Neuroglia Cells Neurons and Neuroglia Cells 55
Figure 5-2. Multi-channel Imaging of mFPI-injured Rat Brain. 58
Figure 5-3. Multiplex Fluorescence Immunohistochemistry Illustrates Complex Tissue after Treatment. 61
Figure 5-4. The Diagram of Two Neuron Neighborhoods 61
Figure 5-5. Close-ups for the Zoom-in Regions in Figure 5-3. 62
Figure 5-6. Diagrams of Items and Transactions. 63
Figure 5-7. Fast Co-location Mining Example 65
Figure 5-8. The Radar Map Results of Colocation Rules. 68
Figure 6-1. Cell Status Hierarchy of Two Kind of Neuronal Neighborhood. 74
Figure 6-2. Example of Calculating the Distance Weights over all Neuron-glial Pairs. 77
Figure 6-3. Ridge Regression in $L2$ Norm and LASSO in $L1$ Norm Penalty 80
Figure 6-4. Comparing of LASSO, Group LASSO and Sparse Group LASSO. 81
Figure 6-5. Illustration of Tree Guided Group LASSO. 83
Figure 6-6. Kernelled Neuronal Neighborhood Attraction over the Whole Brain. 88
Figure 6-7. The Feature Selection Result of Neuron Status Comparisons by Using Sparse Group LASSO Algorithm. 90

Figure 6-8. Color-Coded Heatmap Display of LASSO Coefficients. 95
Figure 6-9. (A) Multiple IHC Image (B) Nuclei Seed Data and Analysis of ROIs..... 96
Figure 6-10. Sparse Group LASSO Coefficients in Stroke Dataset. 97
Figure 7-1. Register the Source Image to Target Image at Small-scale. 103
Figure 7-2. ORB Feature Keypoint Detection of an Example Patch. 105
Figure 7-3. Tile Range Generation. 110
Figure 7-4. The Comparison of Keypoint Extraction of the Original and Tiled-based ORB Method. 112
Figure 7-5. The Comparison of Keypoint Matching by the Original and Tiled-based Method. 114
Figure 7-6. The Comparison of the Original RANSAC and Hierarchical RANSAC. . 116
Figure 7-7. An Example of Fixing a Folded Area by Regional Refinement 117
Figure 7-8. An Example Difference Map for Registered Image. 118
Figure 7-9. Run Time Summary of Registration. Example of registering a pair of images with the size of 40,000 px by 30,000 px. 119

CHAPTER 1 INTRODUCTION

Computational analysis of multi-channel fluorescence microscopy images is widely used for tissue analysis. Building a comprehensive map for each single cell can help us reveal and quantify the profiles of all relevant cellular and molecular players spreading across brain cell types (e.g., neuron, astrocyte, microglia, oligodendrocyte, endothelial cell, etc.) and regions.

1.1 Experiment Preparation

Figure 1-1. Healthy Rat Brain Sample in UT Health.

Dr. John Redell's team at The University of Texas Health Science Center at Houston has cut healthy rat brains into slides of 10 μm-thick coronal cryo-sections (see **Figure 1-1**). Each section is incubated using a cocktail mixture of 10 non-cross reactive and spectrally compatible biomarkers. To generate readouts of a rich panel of biomarkers, the sample slides are stained in precisely designed fluorescent protocol in Dr. Dragan Maric's Lab at

National Institute of Neurological Disorders and Stroke, National Institute of Health (see **Figure 1-2**). Then, the full sets of multiplex images of the rat brains are scanned by microscopies (see **Figure 1-3**).

Figure 1-2. Tissue Staining in the Wet Lab at NINDS, NIH.

Figure 1-3. Microscopy Imaging in the Dry Lab at NINDS, NIH.

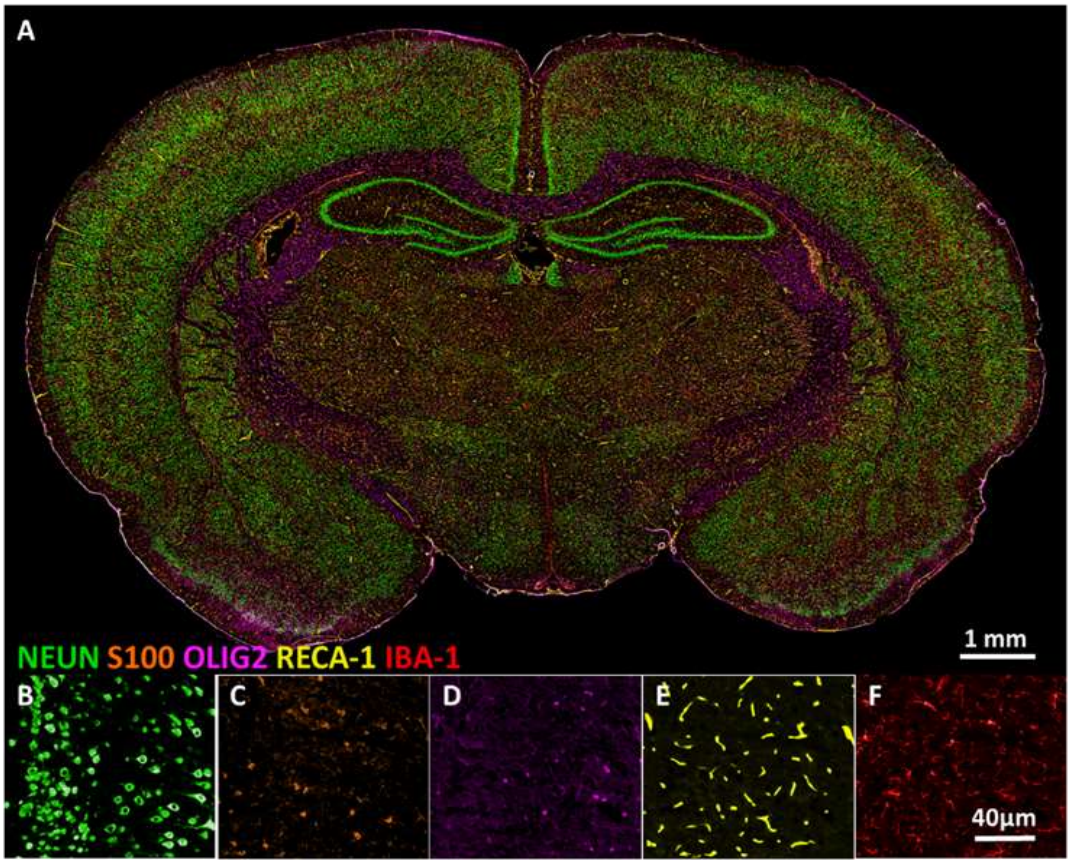


Figure 1-4. Whole Rat Brain Tissue from Multiplex Imaging.

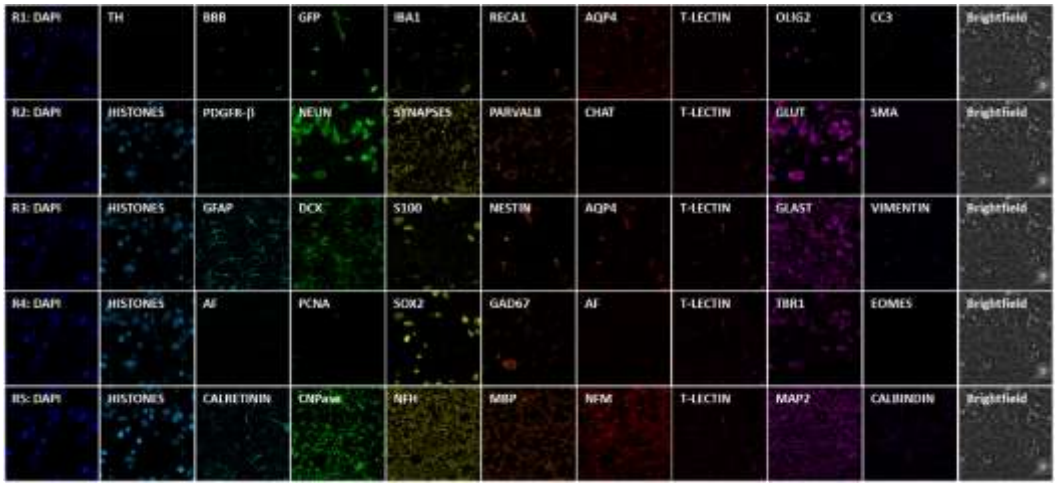


Figure 1-5. Close-up Images from Five 10-plex Imaging Rounds.

1.2 Descriptions of the Dataset

There are five rounds of 10-plex immune-labeling images for each rat brain tissue (see **Figure 1-4 (A)** and **Figure 1-5**). The major channels are DAPI and Histone channels representing the nuclei, and main cell types channels for Neurons, Astrocytes, Oligodendrocytes, Endothelial Cells and Microglia (shown in **Figure 1-4** (**B)-(F)**).

Traditional nuclei images stained are performed only using DAPI, a DNA binding dye. However, uniquely to the rat brain, some cells' nuclei are barely visible in the DAPI channel. To overcome this problem, Dr. Dragan Mari added a complementary channel using Histone antibodies, enabling more reliable nuclei detection and segmentation results. We will use DAPI and Histone channels for automatic nuclei segmentation by watershed; DAPI, Histone and five main biomarkers for expectation driven segmentation and phenotyping; the main biomarkers and the cell status/ cell subpopulation channels for neighborhood analysis; all 50-plex channels for image registration.

1.3 Objectives

This dissertation is formulated as follows:

CHAPTER 2 introduces a parametric nuclear segmentation method using the watershed method. The result will be used as the noisy training labels for the next chapter,

CHAPTER 3 proposes a novel deep learning based nuclear segmentation pipeline, achieving a reliable segmentation with zero human effort.

CHAPTER 4 displays the phenotype interactive interface.

CHAPTER 5 models the spatial representation of neuronal neighborhoods, and analyzes the cellular neighborhood using co-location method.

CHAPTER 6 reveals the alterations of neuronal neighborhoods over different conditions using the feature selection method.

CHAPTER 7 proposes an advanced solution to large scale multiplex image registration.

CHAPTER 2 AUTOMATIC NUCLEAR SEGMENTATION BY WATERSHED

Reliable nuclei detection and segmentation is crucial to precise cell counting and capturing nuclear morphology. Classical nuclei detection is typically performed only on images stained using DAPI, a DNA binding dye. However, uniquely in the rat brain, the nuclei in some regions are barely visible in the DAPI channel, due to the rat brain image peculiarities by vacant staining. To overcome this problem, we added a complementary channel using histone antibodies. With this modification, we developed a complete pipeline for cell segmentation: from image binarization and seed detection to border generation. Thus, two main problems caused by rat brain image peculiarities are solved: 1) the impact of vacant staining in the center of nuclei on segmentation is diminished by a Gaussian filter and a border completing technique; and 2) densely distributed nuclei blobs largely observed in Hippocampus region are detected by seed controlled watershed segmentation method via compactness consideration. The results show that both the nuclei detection accuracy and the shape fidelity of segmentation are improved.

2.1 Automatic Nuclei Segmentation

Traditional label-free nuclei segmentation requires parameter sourced computer vision algorithms to delineate nuclei borders sourced on their morphology, granularity, shape and intensity distribution pattern. To accomplish this, we leveraged Yousef ‘s method which uses graph cuts methods for binarization and multi-scale Laplacian of Gaussian (LoG) for

seed detection (LoG) [1] for seed detection [2]. However, we improve the simplified binarization and seed detection procedures with a modified border generation strategy. The pipeline is shown in **Figure 2-1**.

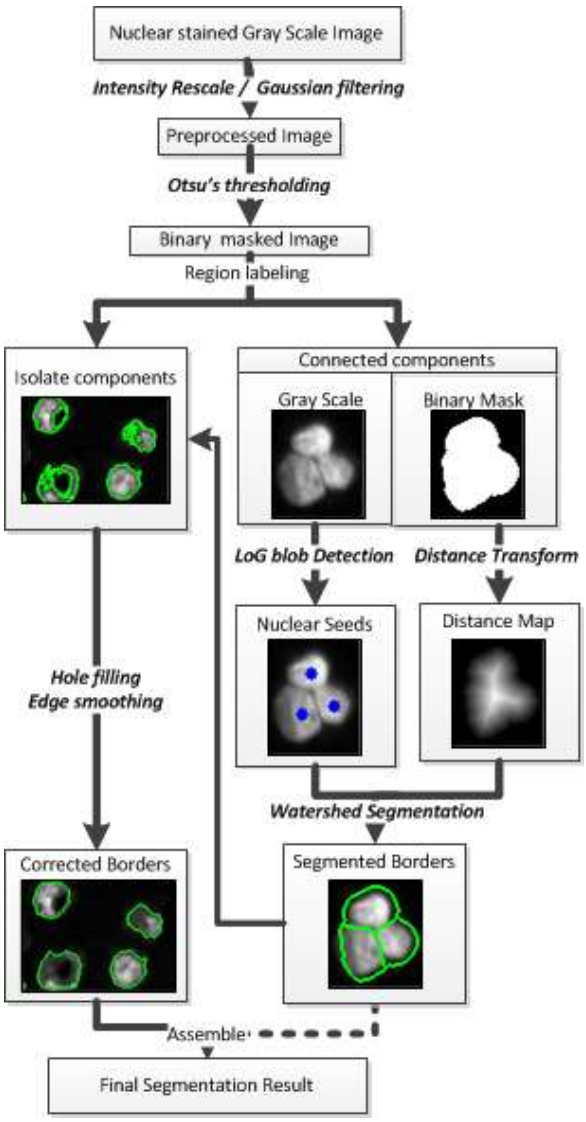


Figure 2-1. The Pipeline for Automatic Segmentation

2.1.1. Image Filling

In fluorescent images of the rat brain, some nuclei are observed with an absence of staining in their centers, visually expressed as circular black holes. Ignoring these holes during image processing result in severe under-segmentation problems and failure to recognize the correct seeds for subsequent steps. Therefore, image filling is applied on the raw image to fill the holes through Gaussian smoothed adjacent pixels. As shown in the **Figure 2-2**, this hole-filling method is effective filling the cell center when the nucleus is recognizable (blue and yellow arrows) and eliminates the over-segmentation problem in generating correct seeds (blue arrow).

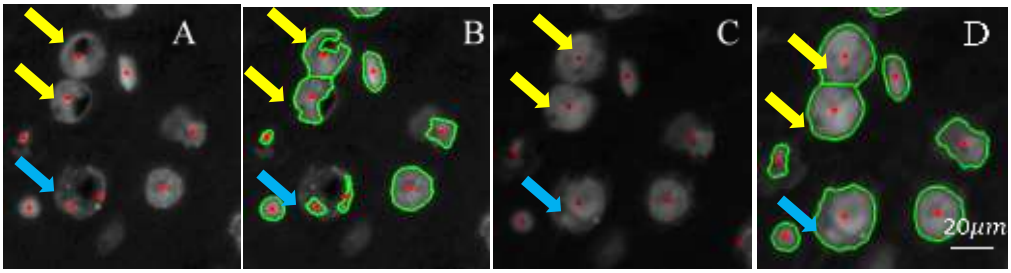


Figure 2-2. Visualization of Image Filling For Segmentation.

2.1.2. Seed Detection

Seed detection is required to detect the existence of cell objects for generating labelled markers for segmentation. We adopted a common blob detection method “Laplacian of Gaussian” (LoG) to identify cell blob objects. Since centroids of blobs are located on local maxima and the pixel intensity gradually decreases from centroid to border for each object,

we consider each cell nucleus a blob object because it fits the required texture properties. When the general Nuclear Radius r is set to $\sigma\sqrt{2}$, LoG is shown as [3],

$$L(x, y;\ \sigma) = \sigma^2\left(\frac{\partial^2 G(x,y;\sigma)}{\partial x^2} + \frac{\partial^2 G(x,y;\sigma)}{\partial y^2}\right)$$

$$\boldsymbol{\sigma} = [\boldsymbol{\sigma_{min}}, \ldots\ldots \boldsymbol{\sigma_{max}}]. \tag{1}$$

2.1.3. Watershed Segmentation

We utilize the seed-controlled watershed algorithm to separate the connected nuclei masks and outline the exact border of each object. The analogy behind the watershed segmentation algorithm is to flood a basin from the background image to the foreground image along the water flow until the water from different origins converge to form the borders of the objects [4].

n the seed-controlled watershed segmentation, the black and white image gained from the binaries thresholding method (we used Otsu's thresholding method) is used as the background; the dilation of detected nucleus seeds with respective markers are used as the foreground, and the intensity image is used as the directional water flow. Since nucleus clumps (especially in hippocampus regions of rat brain images) clustered tightly are difficult to separate as **Figure 2-3** shows. Thus, we adopted an advanced modification of

the watershed algorithm, which incorporates a compactness consideration to impose separated components to have a similar size and shape [3].

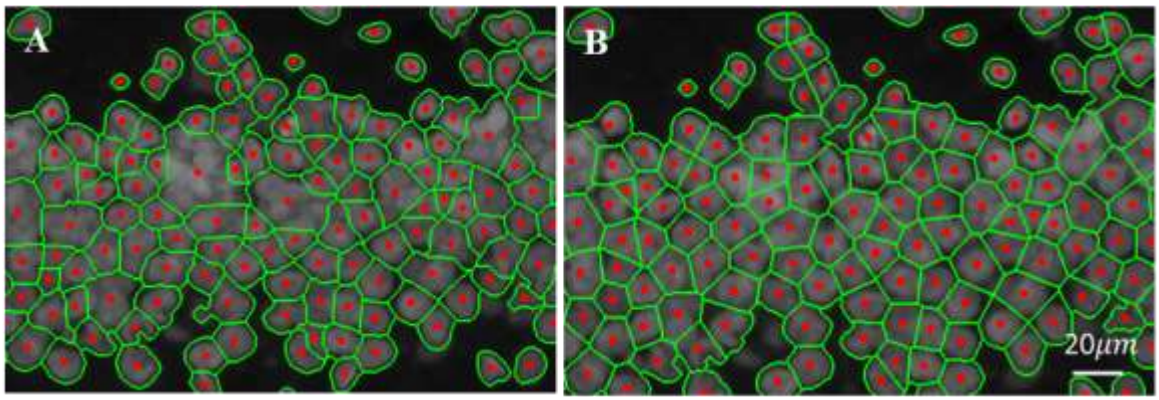


Figure 2-3. The Comparison of Watershed Segmentation with Compactness Constraint in A Cropped Region of Hippocampus. A and B show the result before and after the constraint.

2.2 Results

Table 2-1 (1) shows, a cropped region with 3000 by 7000 pixels, 3891 cells have been detected. This indicates that a high detection precision is achieved in the DAPI channel after the image filling step. Also, the highest recall is achieved in the composite nucleus image after image filling. Therefore, the image filling technique reduces the false positive and the composite channel reduces the false negative. By combining the application of histone stains as a complementary channel for DAPI and using hole filling techniques, the nucleus detection accuracy in cortex region has improved.

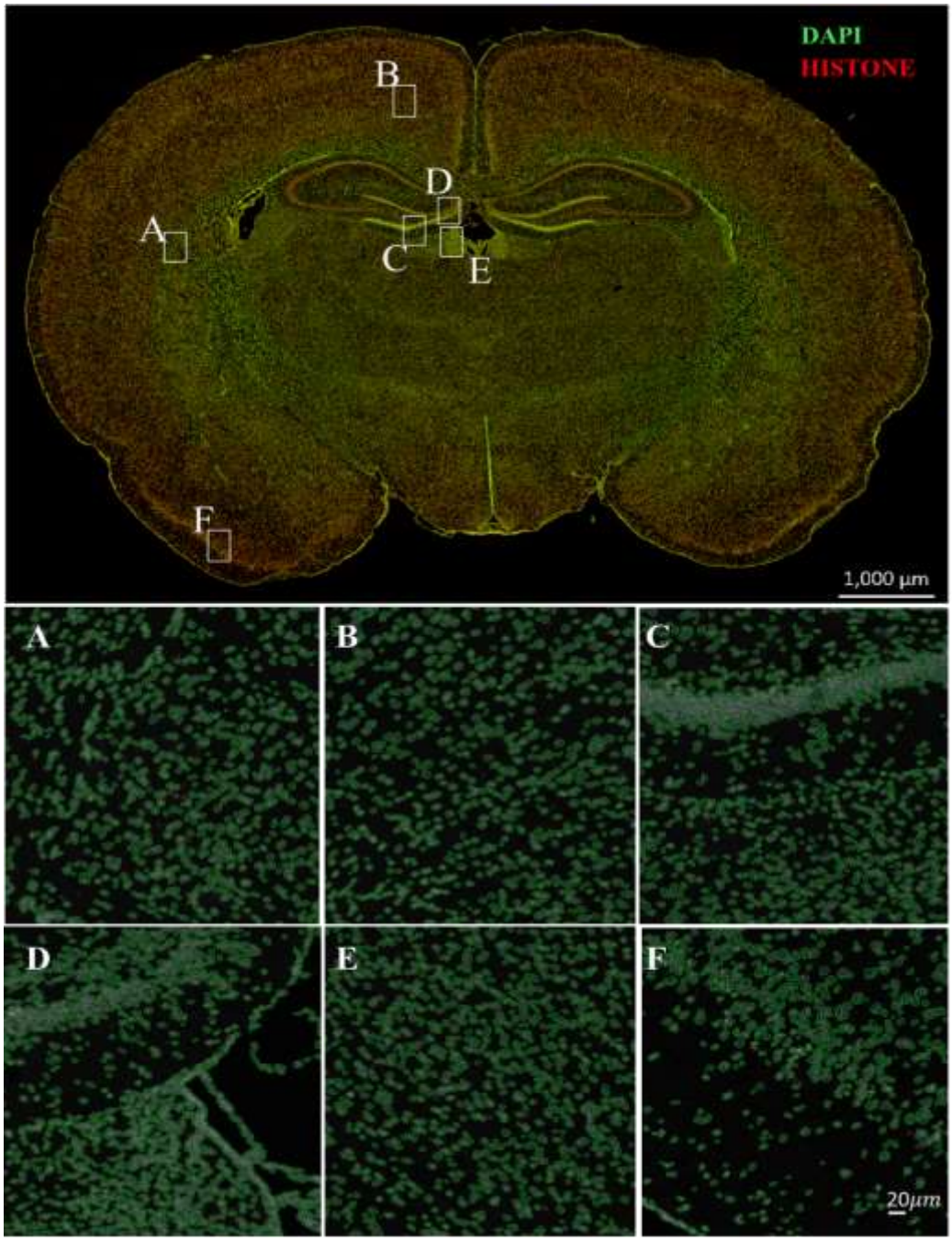


Figure 2-4. The Comparison of Nucleus Detection on DAPI Channel and Composite Channel.

Table 2-1. Detection Performance on Crops of Cortex Regions.

Exp	Region	Channel	Precision	Recall
(1)	**Cortex Sample**	DAPI	98.7%	95.5%
		DAPI (after image filling)	99.5%	97.9%
		DAPI + Histones	99.2%	98.2%
		DAPI + Histones (after image filling)	99.4%	99.1%
(2)	**A**	DAPI + Histones (auto segmentation)	99.9%	96.5%
	B		98.8%	98.4%
	C		99.8%	94.7%
	D		99.9%	94.7%
	E		99.4%	98.1%
	F		99.4%	98.6%

This whole-segmentation pipeline can be implemented on entire rat brain images. For example, in a 16-bit gray scale image of size of 29200×42900 pixels, 223610 nuclei are detected. The segmentation evaluation results of six cropped images (**Figure 2-4**) are illustrated in **Table 2-1 (2)**. This table shows that our method performs quite well in most of the regions except regions C and D where the nuclei are tightly clustered together which forms large connected components.

Not only does the detection accuracy improve by adopting a combination of DAPI and Histones channels, but it also improves the shape fidelity. As shown in **Table 2-2**, comparing single DAPI and Histone channels the composition of the two produces the best nuclear segmentation.

Table 2-2. Improvement of Shape Fidelity in Whole Brain.

Channel	Area Occupancy $\mu \pm \delta$	Cell Population (Missing Cell %)
DAPI	75.7 % $\pm$ 29.5%	193960 (7.1%)
Histone	87.0 % $\pm$ 21.5%	202890 (2.8%)
DAPI + Histone	100% $\pm$ 0%	208835 (0%)

2.3 Conclusions

Conclusively, applying histone antibody stains as a complementary channel for DAPI allows for dim nucleus objects, which are barely visible in the DAPI channel, to be recognized and segmented well. Our segmentation pipeline can produce comprehensive analysis on whole brain images and performs well on most of the cortex regions.

CHAPTER 3 EXPECTATION DRIVEN NUCLEAR SEGMENTATION

Deep neural networks (DNNs) offer state-of-the-art performance for cell nucleus detection and segmentation, which are a foundational step for assessing cell localization, classification, and enumeration in biological environments. However, DNNs require a myriad of manual annotations from skilled biologists for robust algorithm training, which is labor-intensive, expensive and not easily scalable. This study proposes a novel framework for building deep networks to perform Brain Cell Analysis using Fuzzy Labels (BrainCAFL) with minimal human input. Our framework consists of three main components. First, it uses unsupervised methods to generate and refine noisy (fuzzy) labels for cell nuclei through an iterative training process. Second, we introduce a background recovery technique to enhance the accuracy of detection and estimation of segmentation, especially in densely packed brain regions with numerous overlapping cells. Finally, a novel sparse decomposition method is used to identify anomalous cell detection results and automatically correct them to improve the accuracy further. We also provide extensive experiments to demonstrate the high effectiveness of BrainCAFL. Our method achieves mean Intersection over Union (mIoU) of 79.1% compared with 74.2% of the popular MaskRCNN.

3.1 Introduction

Nuclear detection and segmentation are challenging in large-scale brain images because of the high degree of cellular and nuclear heterogeneity, including widely variable spatial distributions and a large diversity of cellular/nuclear morphologies [5]. Classical parameter-based nucleus segmentation methods such as graph-cuts [2] and watershed cuts [3] typically perform well only on sparsely distributed regions but fail in densely distributed regions such as the hippocampus. Recently, deep learning methods have been demonstrated to capture sophisticated image features and generalize well given a large number of data samples [6]. Several well-designed networks, such as UNet [7] and MaskRCNN [8], achieved state-of-the-art accuracies on instance segmentation tasks [9]. Our work on nuclei segmentation by MaskRCNN [10] have proved the feasibility of reliable segmentation in Nano-wells with very few number of cells. There are also some research works combining the network with traditional methods. However, the performance of these supervised learning networks heavily relies on the quality of training samples. Furthermore, for biological-image analysis, human annotations are prohibitively labor-intensive and difficult to obtain due to the strict requirements in domain knowledge. Thus, being able to build a reliable deep network with minimal human annotations is essential.

Motivated by the needs above, we propose an efficient unsupervised learning framework for robust instance segmentation of large nuclear images with zero expert labels. Our framework makes the following novel contributions:

1) Using an iterative training process to improve segmentation quality without human labels;
2) Introducing a background recovery technique to enhance the segmentation accuracy;
3) Developing a novel sparse decomposition method to identify and rectify abnormal nuclei automatically.

3.2 Preliminary Studies

3.2.1. Background Review

The goal of deep learning is to learn from a set of training samples and produce a model capable of predicting a similar outcome. However, when high-quality training samples are not present, the model suffers from overfitting and produces detections with the same error profile as the inputs. This inexact supervision problem has been addressed by weakly supervised learning [11], e.g., training a segmentation network by using bounding boxes as inputs [12, 13]. Our work focuses on using a noisy (fuzzy) labeled training set from a parametric based method to produce reliable segmentation of nuclei.

An early stopping strategy [14] has been shown to make deep neural networks more robust to noisy labels, whereas it requires a good validation set to determine termination. Bootstrapping technique has been proposed for repeatedly training the network using its predicted labels from the previous iteration [15]. This technique was robust to noisy labels in image classification tasks. Unfortunately, our experiments show that this method is inadequate for biological cell segmentation since it does not eliminate non-random errors. Another recent study implements iterative training and graph cut refinements to the output predictions [16]. Our work is inspired by these methods but offers novel mechanisms for correcting the network's errors between iterations.

3.2.2. Train Deep Neural Networks with Label Noise

The training of supervised deep Neural Network demands massive labeled dataset, generally highly relying on extensive and skilled annotations from human efforts. Due to the fact that comprehensive high-quality labels are difficult to obtain, practically, annotations are more often generated from traditional parametric methods, which exist an inevitable large amount of label noise. Former studies [17, 18] have showed that most of the machine learning methods in big data-driven data mining field are robotic to label noise. Recent experiments including [19, 20, 21] also indicate the same resistance on deep learning.

Some researchers specifically focus on the discussion of which patterns of label noise greatly influence the performance of deep learning models. The work [22] claimed that "concentrated noise", where errors are not locally concentrated in different parts, causes dramatic declines in accuracy performance, and random noise causes the least influence. The reason behind it could be similarly explained of K-nearest neighbor classification algorithm: when the distribution and the amount of concentrated noise are large enough to change the centroid of clusters, the final classification result makes great turbulence, whereas random noise, in the form of sparsely distrusted samples in each cluster, impacts less to the centroids. This theory gives us strong support that deep learning could tolerate non-random noise as long as the noise does not show concentrated patterns.

3.2.3. Mask R-CNN Model

We use MRCNN model as the unit model for our iteration training pipeline. As a deep learning instance segmentation for detection and segmentation, MRCNN do the 3 tasks at the same time, object detection, classification and instance segmentation, as shown in **Figure 3-1**.

Figure 3-1. Diagram of Mask RCNN.

MRCNN is based on Faster RCNN [8], and shares the similar modules for object detection, classification. At the same time, MRCNN adds a mask header and uses ROI Align rather than ROI Pooling (Faster RCNN used), which returns a more accurate image feature map location to the original image. There are three main components of the MRCNN model (see **Figure 3-2**): backbone represents the high dimensional features of raw images by kernels; Region Proposal Network (RPN) generates the Region of Interests (ROI) of suspected region crops; and the three headers learn and reduce the loss for location, classification, and masks. The detailed explanation of loss functions of MRCNN can be seen in **Appendix A,** and the modules are in **Appendix B.**

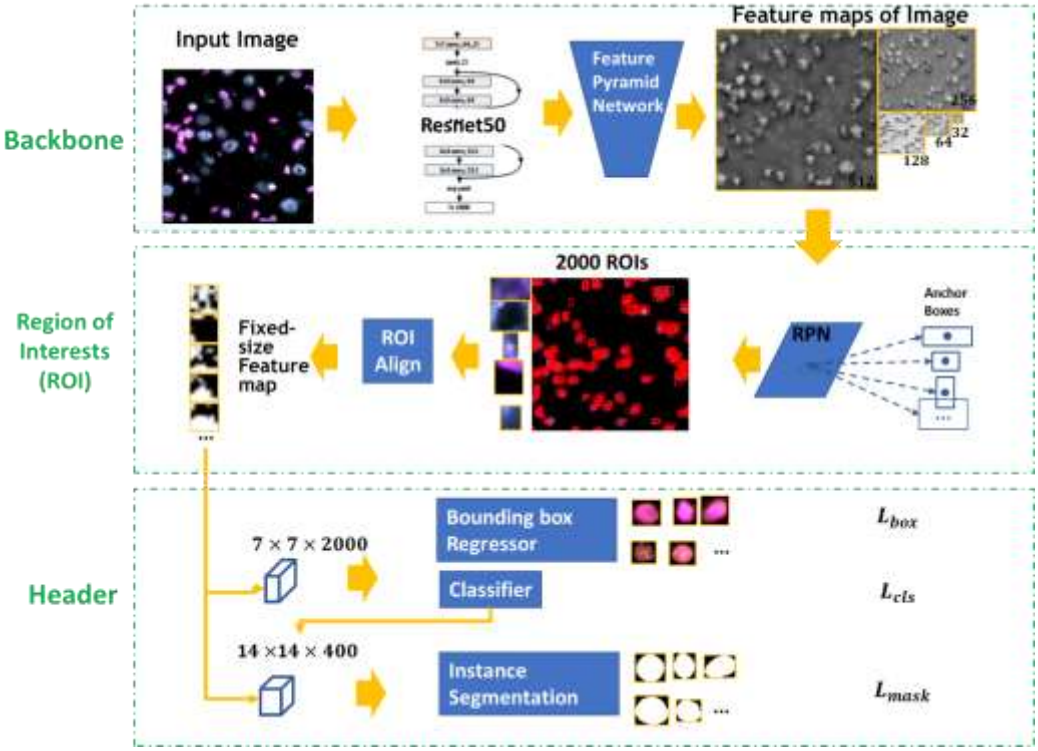


Figure 3-2. The Detailed Structure of MRCNN For an Example of Nuclei Image.

3.3 Proposed Methods

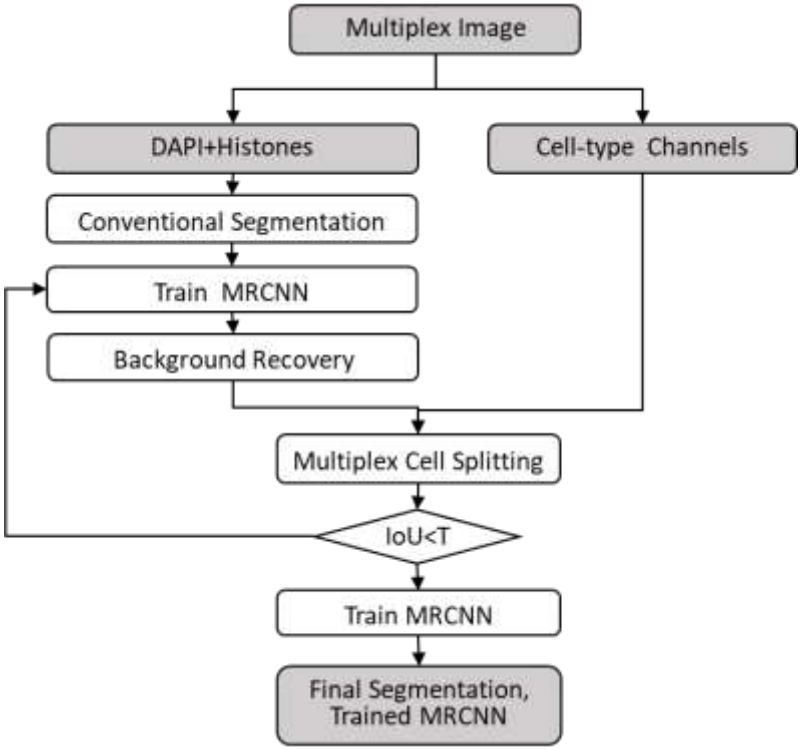


Figure 3-3. Fully Automatic Pipeline for Brain Cell Nuclear Segmentation.

The **Figure 3-3** shows an overview of our framework for training a cell segmentation deep network with zero human annotation. Given an unlabeled training dataset, a watershed-based method is used to generate noisy segmentation labels from a DAPI stained grayscale image. These noisy labels serve as the input to training the initial instance-segmentation deep network using the MaskRCNN algorithm. Our pipeline then uses a background recovery technique to enhance the output of MaskRCNN, especially for

densely packed regions with many closely apposed cells. We observe that most of our model's errors are due to detecting multiple overlapping cells as a single cell. For this reason, our pipeline introduces a novel sparse decomposition for splitting these overlapping cells automatically. The rectified MaskRCNN's output then serves as the new input to updating itself. Our experiments show that this iterative training process can significantly reduce noises and improve the model's performance. In what follows, we will explain each component of our method in detail.

3.3.1. Iterative Training

For an image X with M pixels, its segmentation labels at pixel x_m is denoted by y_m, where L is the total number of cells in the image. For a deep learning network with parameters θ , the predicted label is estimated as $Y = f(X; \theta)$. Supervised learning approximates Y to Z where Z is the known ground truth. However, in the weakly supervised learning scenario, there is no ground truth but a known noisy label Z. So, we assume, reasonably, that Y will converge to the real ground truth after a certain number of iterations T.We initialize Y^t at iteration $t = 0$ by well-known automatic seed-controlled watershed segmentation [16]. For the iterations more than 0, we update y^t by the predicted labels from their previous iterations, i.e.,

$$Y^t = \begin{cases} Z & if\ t = 0 \\ f\,(X, Y^{t-1};\, \theta^t) & if\ t \geq 1 \end{cases} . \quad (2)$$

To estimate the final prediction value $Y^* = Y^T$, we adopt an Expectation-Maximization (EM) algorithm, as shown in **Algorithm 1**. In the E step, we estimate the labels from the current model using **Equation 2**. We denote the posterior probability of the latent variable Y^t, θ^t as $p(\,Y^t|\mathrm{X};\theta^t)$ .Then, in the M step we update the parameters to maximizes the following posterior probability,

$$\theta^{t+1} = \underset{\theta^t}{\operatorname{argmax}}\, p(Y^t|X;\theta^t)\,. \quad (3)$$

There are a couple of ways to determine the stopping criteria, and we provide a straight-forward one: the overall intersection-over-union (IoU) difference between the current and previous iteration falls below a predefined threshold ε (set to 0.3 from the original MRCNN architecture in [23]).

Algorithm 1. Iterative learning with Noisy Label

Inputs: Initial Training Mask Z^0, Initial MRCNN Parameters θ^0, Image X,Maximum iterations T, Threshold ε

Outputs: Final estimate segmentation result $Y^* = Y^T$

for (t = 0 to T):
  E step: Test the network to get Y^t by **Equation** 2
  M step: Train the network to get θ^{t+1} by **Equation** 3
  $d = \mathrm{IoU}(Y^{t+1}, Y) - \mathrm{IoU}(Y^t, Y^{t-1})$
If ($d < \varepsilon$) : *stop;*
$\theta^t = \theta^{t+1}$

3.3.2. Background Recovery for MRCNN Detection

In weakly supervised training, there commonly exist objects that are either not labeled or are poorly-labeled such that the network is unable to detect them. These miss-detected objects cannot be corrected by adjusting the detection threshold or data augmentation techniques because the network never learns the correct inputs. The not-labeled training data usually come from regions with no segmentation output, and the poorly labeled objects come from the vicinity of segmented objects. To address this issue, we define the background region as the pixels that have not been labeled. Based on the assumption that background regions contain all missed objects, we introduce a Background Recovery algorithm to deploy further detections, as shown in **Algorithm 2.**

Algorithm 2. Background Recovery

Inputs: Detection model $\mathcal{F}$, input Image X
Outputs: Segmentation result $\sum \boldsymbol{y}^{(k)}$

$Y^{(0)}, b^{(0)} \leftarrow \mathcal{F}(X)$
$k = 0$
While $(b^{(k)} \neq \emptyset \text{ or } k = 0)$:
    $B^{(k)} = \bigcup_1^k b^{(i)}$
    $Y^{(k)} \leftarrow X \odot \left(1 - B^{(k)}\right)$
    $Y^{(k)}, b^{(k)} \leftarrow \mathcal{F}(X^{(k)})$
    $k \leftarrow k + 1$

Given an input image X, and a trained model $\mathcal{F}$, the output of the model at step k is denoted $Y^{(k)}$. Then we use a binary mask $b^{(k)}$ to indicate the location of the detected pixels of $Y^{(k)}$ at step k. For the mask at pixel m, $b_m^{(k)}$ is *True* only when $Y_m^{(k)} > 0$. An accumulated binary $B^{(k)}$ is set to represent the union of the binary masks until step k,

$$B^{(k)} = \bigcup_1^k b^{(i)} \quad . \tag{4}$$

Background Recovery runs the same detection model on the background image. The iterative loop ends when there are no longer objects detected in the background region. It is clear that this technique does not require training the network again, but rather, it aims to make use of the already trained model as much as possible.

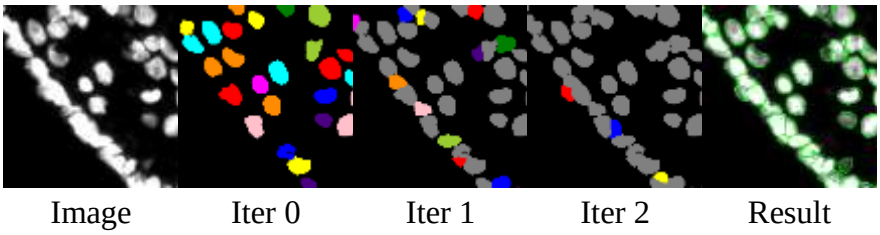


Figure 3-4. Background Recovery Example for an Image Patch.

Handing nuclear clumps, i.e., groups of closely adjacent or partially overlapped cells, is one of the main challenges in nuclear segmentation (as shown in the first panel of **Figure 3-4**). Our method can tackle the nuclear clumping problem because it is able to segment a subset of the objects in clumps, even though the network was never trained on big clumps. A region proposal network tends to look for strong candidates, given a fixed number of

proposals. In this way, the network favors the correctly segmented objects and ignores the wrong ones.

When we applied background Recovery, the detected objects from previous steps are removed; thus, the network focuses on detecting objects in the rest of the image. Examples of the removed objects from previous steps are shown as the gray areas in **Figure 3-4**. In general cases, big clumps are thus broken into smaller clumps that become easier to analyze.

3.3.3. Cell Splitting and Phenotyping using Sparse Decomposition

We define a segmentation anomaly as having more than one nucleus inside the mask and therefore require splitting. Although Background Recovery can separate most of the big clumps, there remain some small clumps as anomaly segmentation. Assuming nuclear masks of all cell types are given (starting now referred to as cell type masks), intuitively, we can regard a nuclear mask in the DAPI channel (a channel that detects nuclear DNA) as the union of all cell type masks. With this in mind, we perform a sparse decomposition to find the significant cell type masks, as described below.

I. Segmentation Anomaly Detection

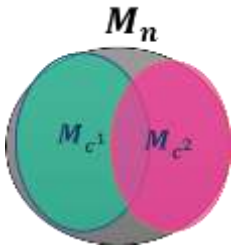


Figure 3-5. Sparse Decomposition of Nuclear Masks.

For the sparse decomposition problem, a nuclear mask M_n is modeled as being composed of C different cell type masks M_c of which only N of them are significant (M_n *and* M_c are the vectorized binary masks), as **Figure 3-5** shown,

$$M_n = \sum_{c=1}^{C} (\alpha_c \ M_c)$$
$$N = \sum_{c=1}^{C} \|\alpha_c \geq \tau\|_0, \quad 0 < \alpha_c < 1. \tag{5}$$

The sparsity coefficient α_c defines the weights of each M_c and c is claimed as a significant channel if α_c is larger than a predefined threshold τ. In this problem, a cell can be split into 3 cells at most, i.e., $N = 3$ .To efficiently solve this sparse decomposition problem, we used a simple greedy approximation algorithm Orthogonal Matching Pursuit as described in [24] where it aims to minimize the difference of M_c to each M_n,

$$\hat{\alpha} = \underset{\alpha_c}{\operatorname{argmin}} \| M_n - \sum_{c=1}^{C} (\alpha_c M_c) \|_2^2, \ s.t. \|\alpha_c\|_0 \leq N . \tag{6}$$

However, the equation above needs also consider the following constraints:

1) **Valid cell type channel nuclear masks should not contain cytoplasm and processes**

   Given that $\|M_n\|_0 \geq \|\cup M_c\|_0$, we can no longer use subtraction of M_c to M_n to measure their difference. Instead, we come up with a ReLu style of difference function to guarantee that the pixels outside M_n are exclusive. We define the asymmetric difference function of M_n to M_c as

$$(M_n - M_c)^+ = \begin{cases} M_n - M_c, & if\ (M_n - M_c) > 0 \\ 0, & else \end{cases} . \quad (7)$$

2) **Multi-channel decomposition should require a non-negative selection of the coefficients.** Thus, the sub-optimal solution of the equation should guarantee the non-negativity of the sparse coefficients. A more suitable solution such as Non-Negative OMP [18] must be applied such that the optimization **Equatio**n **(7** is changed to

$$\hat{\alpha} = \underset{0<\alpha_c<1}{\operatorname{argmin}} \left\| \left(M_n - \sum_{c=1}^{C} (\alpha_c M_c) \right)^+ \right\|_2^2, \ \text{s.t.} \|\alpha_c\|_0 \leq N. \quad (8)$$

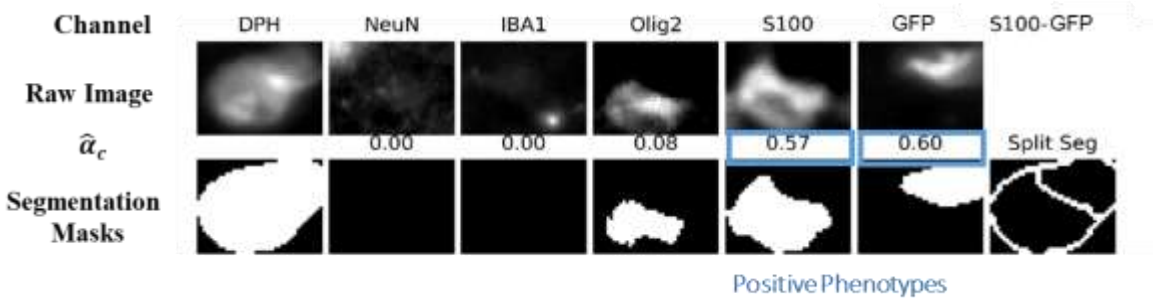


Figure 3-6. Example of a Nucleus Segmentation Split into Two Nuclei. The astrocyte is stained in S100 and the endothelial cell is strained in GFP.

An example of sparse decomposition of nuclei is illustrated in **Figure 3-6.** The sparsity coefficients α_c of all five channels are [0, 0, 0.08, 0.57, 0.6]. After thresholding by 0.3, only S100 (a biomarker of astrocytes) and GFP (a marker of rat endothelial cells) channels are significant channels, i.e., this mask needs to be split into an astrocyte and an endothelial cell. The coefficient threshold can be set to [0,1]. The lower it is, the more low confidence objects (detailed in **3.4.2**) will be detected. It set to 0.3 based on the experimental testing.

II. Cell Type Nuclear Mask Generation.

There are numerous methods to binarize the cell type image and create cell type specific nuclear masks M_c. Here we adopt a straightforward method, which produces reliable results on most of the cell-type images identified using pan-specific biomarkers, including NeuN for neurons, IBA1 for microglia, Olig2 for oligodendrocytes, S100 for astrocytes and GFP for endothelial cells.

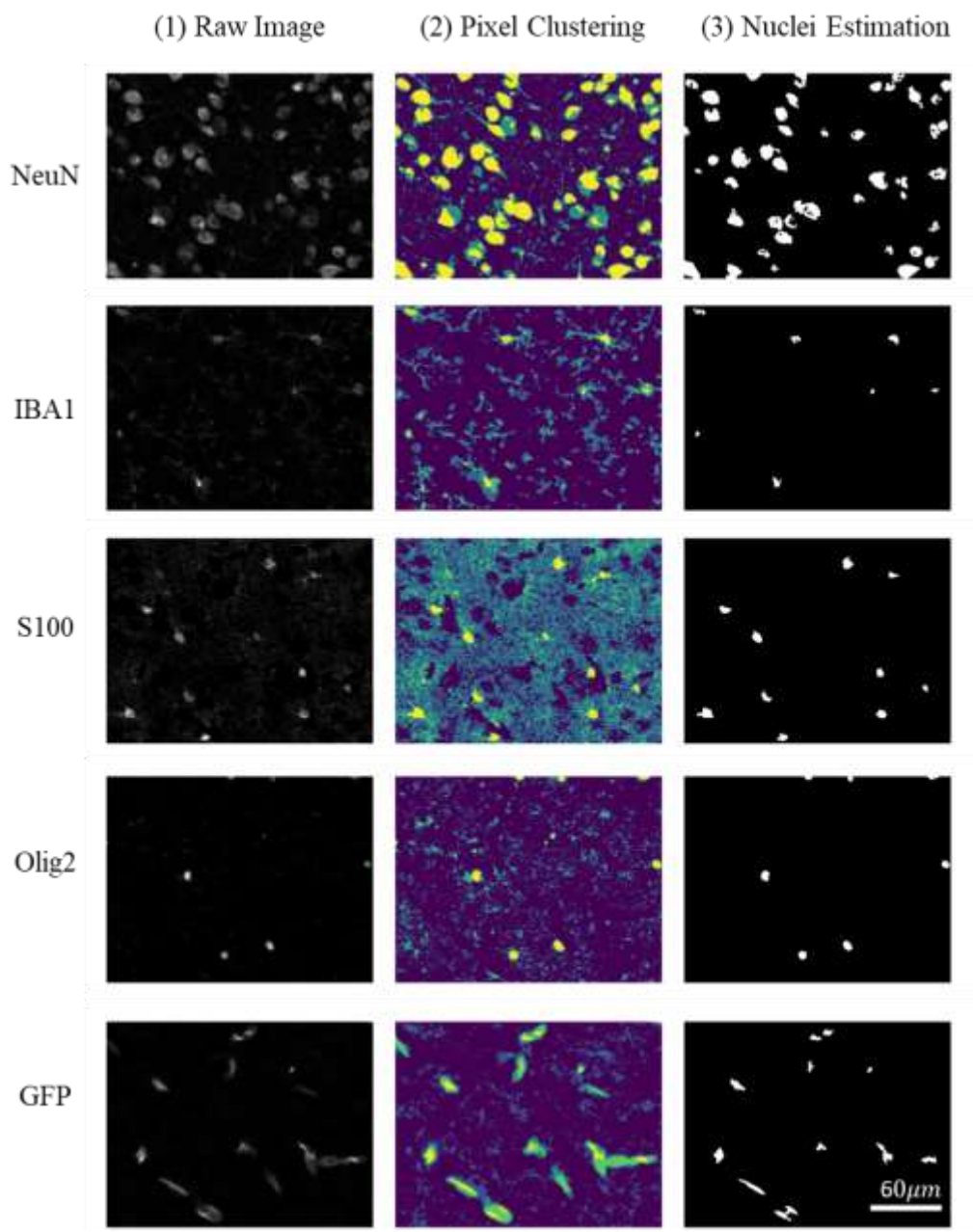


Figure 3-7. Cell Type Mask Generation Examples. (1) The original images. (2) The clustering results: background in purple, auto-fluorescence and cytoplasm in cyan; nuclear of cell Type $\boldsymbol{M_c}$ in yellow, also in (3).

We deploy a three-cluster Gaussian Mixture Model (GMM) on all the pixels in the cell type channel and extract the cluster with the highest mean as the nuclear mask, as shown in **Figure 3-7**. This method performs better than bimodal based methods as it can also filter out the cytoplasm, processes and, auto-fluorescence signals.

III. Split Nuclear Mask Re-generation

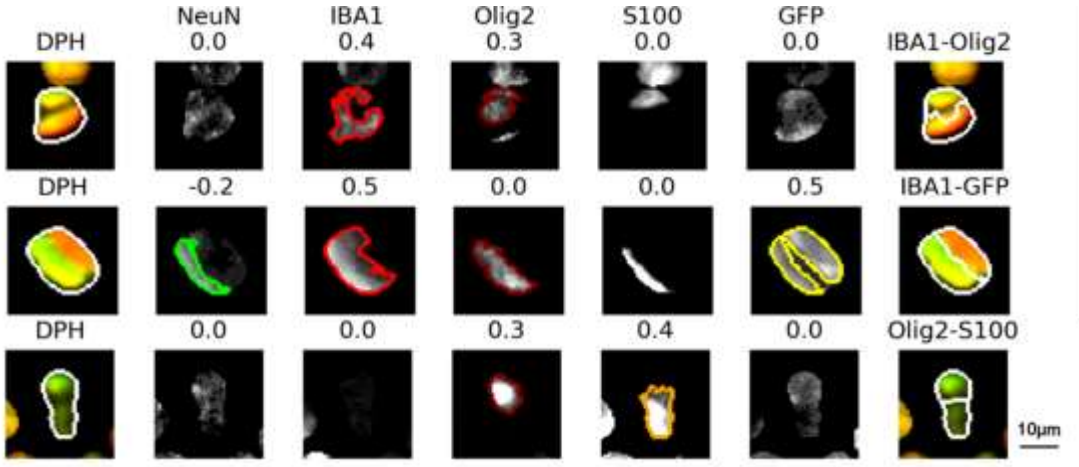


Figure 3-8. Examples of Nuclei Segmentation Split into Different Cells.

After the significant channels are selected, the corrected mask can be replaced as either the individual M_c or the split mask. Considering the characteristic of molecular staining in the cell-type indicating channels, M_k only indicate the existence of the cell and an estimation of the nuclear region. They are not ideal for direct use when the unions of M_c are much smaller than M_n or their intersections are too large. Thus, we apply the watershed segmentation using significant M_c as starting regions to split all the pixels of M_n. It can

guarantee the completeness for the shape to M_n , as shown in the lower right result in **Figure 3-8**.

3.3.4. Large Scale Image Merging

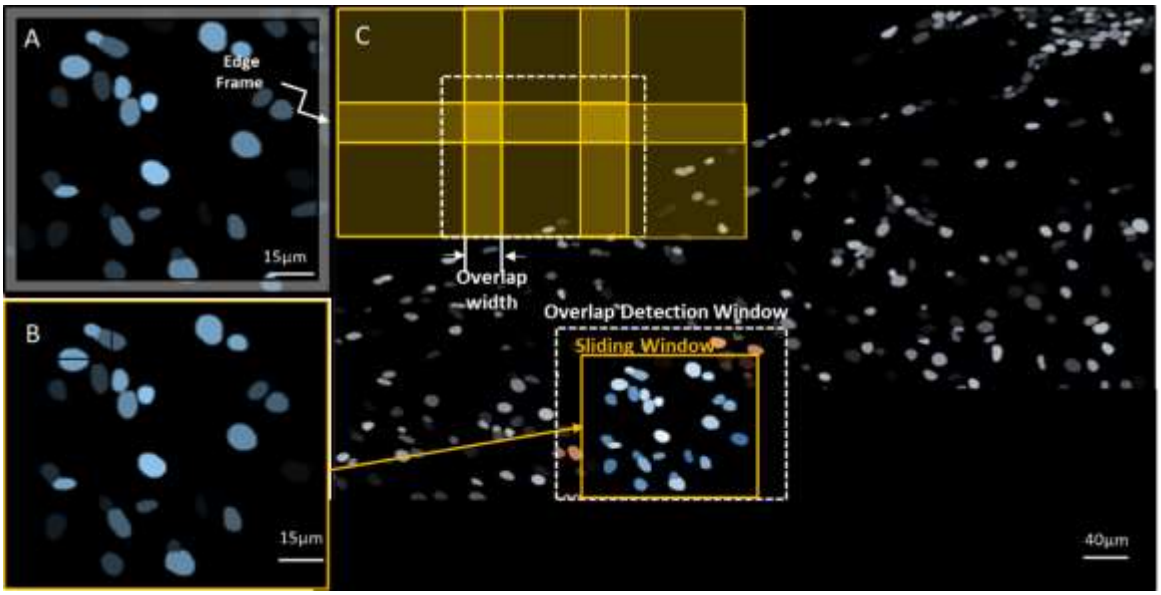


Figure 3-9. Large Scale Segmentation Solution. (A) Sliding window. Detected objects are in blue. (B) Updated sliding window. (C) Composition of sliding windows and checking windows.

One of the key limitations for large scale image segmentation all at once is that the model cannot run on the whole large image because of the memory limitation. Thus, detection and segmentation are often implemented on small images and then merging all the local results together to form a complete whole image segmentation. The type of merging techniques is also called agglomeration [25]. The designing of the merging

algorithm should be cautious about the setting of local cropped window, edge effects and overlapped objects. We designed the algorithm as described in **Algorithm 3**.

Algorithm 3. Large Scale Segmentation Merging Strategy

Inputs: Segmentation masks of all tiles of the image, $\boldsymbol{M} = \{m_1, m_2 \ \ldots \ m_N\}$,
Outputs: Segmentation result

set the size of sliding window *SlideSize* and checking window *CheckSize*,
set the width of overlapping area *OverlapWidth* .
s.t. *CheckSize*> *SlideSize*> *OverlapWidth*
for (m_i in $\boldsymbol{M}$) :
Read a segmentation result in a sliding window
Remove the objects in the edge. Create an edge frame with the width of 5 px.
if (the objects with more than 30% of the pixel located in the edge frame):
Remove the whole object
else:
Keep the object
Creating a checking window for the sliding window
Update the segmentation results in checking window and place it to the whole image

An example can be seen in **Figure 3-9**. A sliding window (**A**) is defined in a fix size 300x300, the detected objects are labelled in green colors, gray area shows the created edge frame for edge object checking. To update the sliding window, edge objects have been removed (**B**). Then, sliding windows and checking windows are composited of for large scale (arbitrary size) segmentation (**C**). Sliding windows are expressed in yellow solid frame (with overlap-width= 50px); Checking window of the sliding window B is expressed in white dashed frame, the objects to be checked in the checking window are labelled in orange colors.

3.4 Evaluation Measurements

We offer both supervised and un-supervised matrices to evaluate the performance of the segmentation.

3.4.1. Supervised Measurements

Supervised Measurements aim to compare the predicted results to the ground truth.

I. The intersection-over-union (IoU)

The IoU score is commonly used in the evaluation of semantic segmentation,

$$\mathrm{IoU}(\hat{y}, y^*) = \frac{\sum_i |\{\hat{y}_i = 1\} \cap \{y_i^* = 1\}|}{\sum_i |\{\hat{y}_i = 1\} \cup \{y_i^* = 1\}|}. \tag{9}$$

$y_i^* \in \{0,1\}$ is the label of pixel i, $y_i^* = 1$ means pixel i is in the object area; $y_i^* = 0$ means pixel i is in the background. $\hat{y}_i \in \{0,1\}$ is the prediction for pixel i of whether it is part of the object. For bounding box detection, IoU > 0.5 is normally considered a "good" prediction, and object is regarded as TP if, IoU > $Thres$. i.e., **Figure 3-10**.

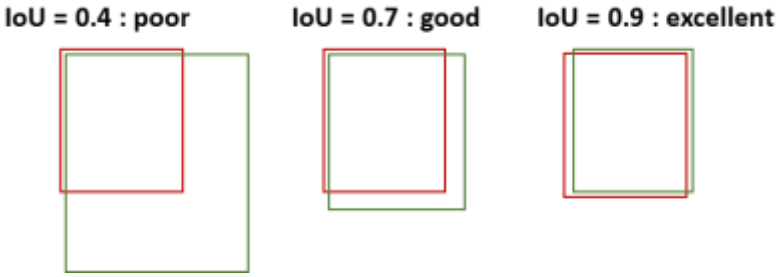


Figure 3-10. Intersection-over-union

II. IoU threshold and matching

For an image with n cells, suppose there are m prediction masks from the network, we can calculate the IoU over the m predictions and n cells to get a matrix M with size $m \times n$. For the prediction mask i and cell j, only if their IoU is the maximum over all the i, j combination, it can be seen as a true match, i.e.,

$$(i, j) = \underset{\substack{i=1,2,\dots n \\ j=1,2\dots m}}{\operatorname{argmax}} IoU\left(Prediction_i, GroundTruth_j\right). \tag{10}$$

Then we can rank all the predictions based on the matched IoU. Given any IoU threshold T, only IoU greater than this threshold will be considered as True positive (TP) example. For calculating the F1score-IoU curve and mean average precision (mAP) value, the threshold set T would be taking from 21 values $T = \{0,0,05.0,01 \dots 0.9,1\}$.

III. Precision Recall and F1 score.

To evaluate the performance of detection accuracy, number of False positive (FP) and the number of False negative (FN) are defined as

$$\begin{aligned} \mathrm{FP} &= m - \mathrm{TP}, \\ \mathrm{FN} &= n - \mathrm{TP}. \end{aligned} \tag{11}$$

Accordingly, Precision, Recall, F_1 score are calculated by

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}},$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}},$$

$$\text{F}_1 = \frac{2\ \text{Precision}\cdot\text{Recall}}{\text{Precision+Recall}}. \quad (12)$$

3.4.2. Unsupervised Measurements

It is impractical to manually detect and delineate the ground truth of all nuclei for a large-scale whole brain image containing over 200,000 cells. So, we used stereology validation by [26], applying an unbiased estimate of the object volume fraction for the whole image and estimate the performance over the whole region. Signal Coverage and split rate target to reflect the false negative cells and unknown class rate targets to reflect the false positive.

I. Signal Coverage

Given that fluorescent nuclei (e.g., DAPI and Histone) dye only stains the informative signals, a prior principle for reliable segmentations is that most of the valid signals are included in the segmentation result.

To find the valid signals from the whole image, it is essential to consider the existence of background noise caused by auto-fluorescence and cell debris. The valid signals are all

the rest of the signals except background. We regard the valid signals and the background noise are distributed in a 2-clustered Gaussian Mixture Model (GMM) with a larger the cluster center the former and a smaller the cluster center the later.

Since the distribution of signal to background noise expresses homogeneity over different regions of the brain, we randomly sample 20 cropped of images with the size of 64x64 pixel from the whole image and convert then to pixel level one-dimensional vectors. Then we train and test a 4-clustered GMM model on those sampled pixels, where the clusters are ordered ascendingly by their centroids' values. The signal-to-background threshold is generating as the average between the maximum of the second cluster and the minimum of the third cluster. Experimentally, the threshold from the 4-clustered model turned out to be more accurate than the 2-clustered one in our study. The valid signals (are the pixels above the generated threshold τ.

For an image X with M pixels, x_m represent the value at the m^{th} pixel. As defined in **Equation (4** , the final binary mask of the all the detected cells is B, then the signal coverage is presented as

$$V = X \cdot B,$$

$$Coverage = \frac{\sum_{m=1}^{M} v_m \cdot (v_m > \tau)}{\sum_{m=1}^{M} x_m \cdot (x_m > \tau)},\ x_m \epsilon X, v_m \epsilon V. \tag{13}$$

The signal coverage always falls in (0,1), it also reflects the sensitivity of the detection model, the higher the coverage is, the less false negatives objects exist for all nuclei.

II. Split Cell Rate

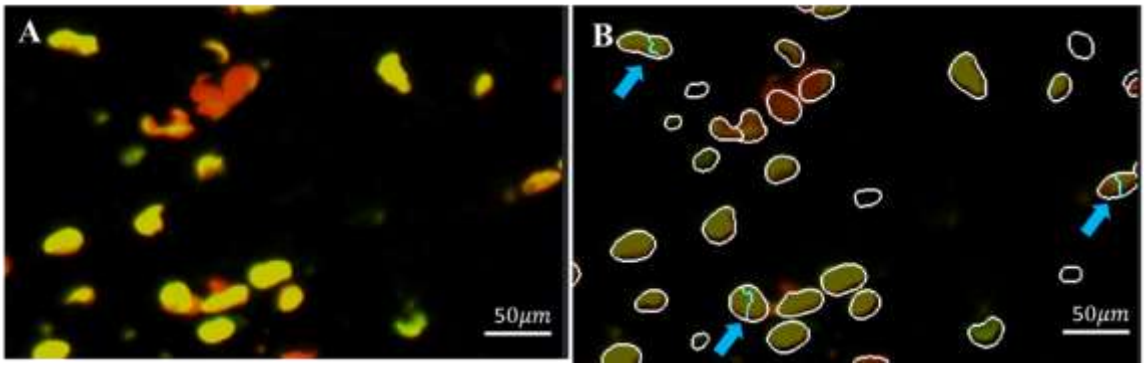


Figure 3-11. Raw Image and Splitting Cells. Splitting cells are indicated in cyan.

From **3.3.3**, we have used sparse decomposition to detect the segmentation objects with more than one cells, such that the number of splitting cells for the whole image can be easily collected. The split cell rate is defined as

$$Split = \frac{Number\ of\ Splitting\ Objects}{Total\ number\ of\ Objects}. \tag{14}$$

One example of split cell recognition is shown in **Figure 3-11.**.

III. Low Confidence Rate

Another output from **3.3.3** is the cell type classification of each objects. We found that there are certain number of objects unable to classify to any cell types, claimed as Low confidence objects,

$$LowConfidence = \frac{Number\ of\ Low\ Confidence\ Objects}{Total\ number\ of\ Objects}. \quad (15)$$

The low confidence objects are either false positive nuclei detected by the model or unknown class objects expect from the five major cell types. Since the provided cell type channel are designed to cover the majority of the cell types in this dataset, this low confidence rate should be able to reflect the false positive object to some extent. One example of low confidence objects is shown in **Figure 3-12**.

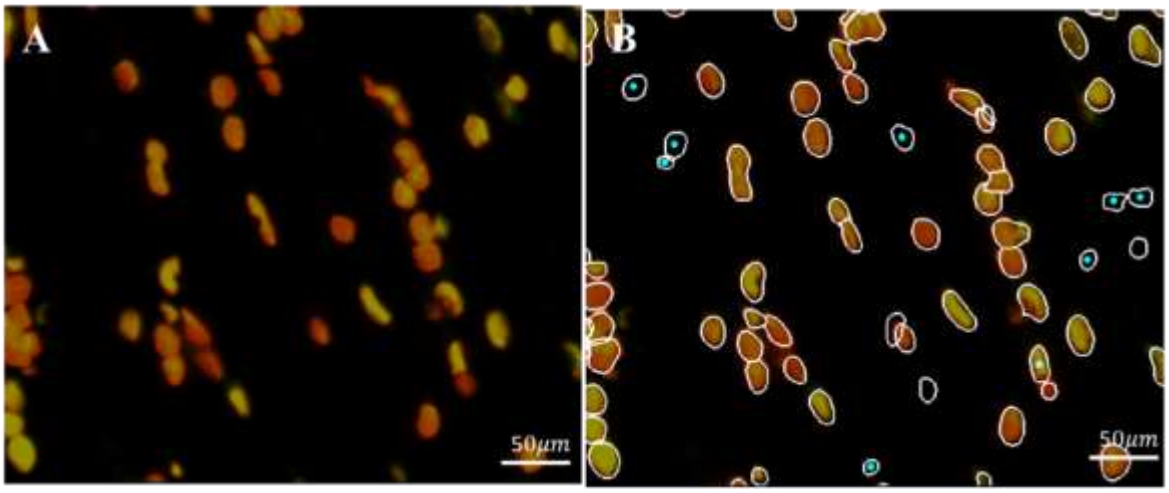


Figure 3-12. Examples of Low Confidence Objects. The seeds are indicated in cyan.

3.5 Performance

For our experiments, the training set consists of 6,000 small images of sizes 512x512 each, cropped from a whole rat brain image, as shown in **Figure 1-4**. One hundred-eighty-one of these randomly cropped images were human annotations for the purpose of validating the results. To be clear, none of the human annotations were used in training. Training MRCNN for one iteration takes five hours using a GPU, while testing on the whole dataset takes three hours, and cell splitting takes thirty minutes. Due to the extensive time it takes to train, we only run two iterations of training, whereas the results show stable improvement after these training sessions.

3.5.1. Dataset Setting and Run Time Summary

We used Tesla V100-SXM2 GPU with 32 GB GPU and 128 CPU with 40 cores. The other detailed settings are shown in **Appendix C.** For the dataset size with the size of 29398×43054 pixels for each channel and eight 8 multiplex channels: three nuclei stain channels for DAPI, Histone, and the composition of DAPI and Histone; five cell type channels for segmentation decomposition only. The whole dataset contains around 200,000 cells, and we set the *Slide Window Size* in **3.3.4** to 512×512. For training, the sliding window's overlap width is 0, so there are 2494 images; for detecting, the overlap width is set to 50 pixels, so there are 6016 images. The training loss can be seen in **Appendix D.**

It takes around five hours to automatically generate the noisy labels, five hours for each round of training, one hour for one step of testing, 1.8 hour for iterative testing, and 20 minutes for segmentation decomposition and one hour for result mering. In total, it only takes around 18 hours to finish the complete pipeline if only run for two training iterations.

3.5.2. Discussions on Background Recovery

Background Recovery helps detect missing objects in the background. We examined 180 validation images of sizes 512x512 and recorded the cell detection results from each step, one example is shown in **Figure 3-13**. For a raw image **(A),** the segmentation results in pseudo colors are shown as follows: **(B)** is the original MRCNN segmentation result as step 0. From step 1,2,3 (image **C, D, E** respectively), the objects that appeared in previous steps are masked in gray. The final segmented result of is showed in **F**. All the testing examples stop before 10 steps, and the average number of cells detected in the background increase with more iterations (see **Figure 3-14 (A)**). It also confirms the convergence of the proposed algorithm. **Figure 3-14 (B)** shows the distribution of the stopping steps over all validation samples. The majority of the samples stop between steps three and five. The image samples stopping at early steps often have sparsely distributed cells, whereas the nuclei in the images that stop at a later step tends to be densely packed. It verified that BgBoost greatly benefits the segmentation of densely packed regions.

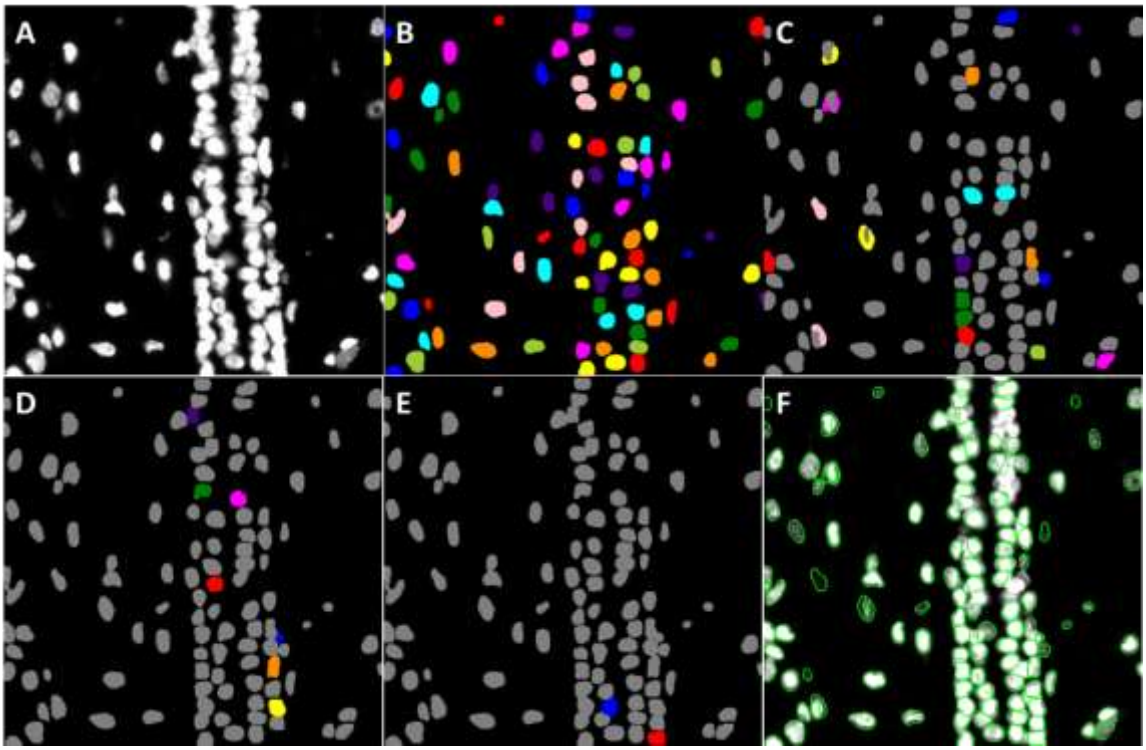


Figure 3-13. An Example of Background Boosting Stopping at Step Three. (A) Raw Image. (B-E) Detection result in iteration 0-3. (F) Final Detection result.

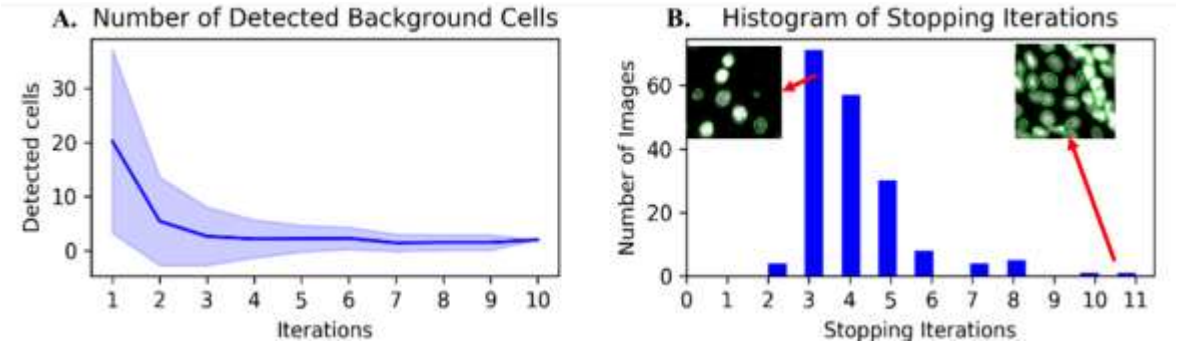


Figure 3-14. Cell Detection Plots as a Function of Background Recovery Iterations. (A) The average number of detected cells of all validation samples in each iteration. The shaded area shows the variations. (B) The histogram of stopping iterations over all validation samples.

3.5.3. Performance on Supervised Measurements

In general, when using the original MRCNN (**Figure 3-15 (2)**), a large proportion of the crowed objects are missed because of the poor annotations (**Figure 3-15 (1)**). Background Recovery and cell splitting are effective refinement techniques as they resolve problems shown above (**Figure 3-15 (3)**). The iterative training produces improved training masks (**Figure 3-15 (4)**). Additional refinement further improves the segmentation masks (**Figure 3-15 (5))**.

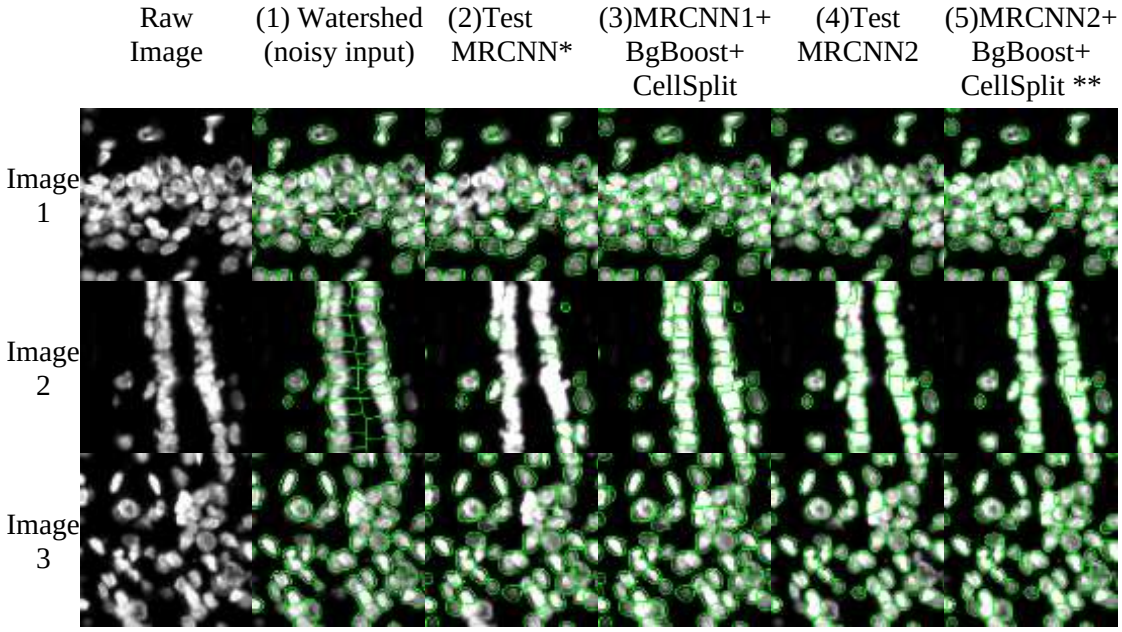


Figure 3-15. Segmentation Result Comparisons. (1) Noisy training sample by watershed segmentation (2) *Original MRCNN testing result (3) Refinements on the original testing result. (4) Testing result after second time training the MRCNN (5) **Final refinement on the testing results.

Table 3-1. Supervised Performance Metrics Comparisons

Method	mIoU	F1@IoU0.5	F1@IoU0.75
Graph-cuts	56.3	46.1	8.4
Watershed (Noisy Label) *	75.7	73.2	55.6
MRCNN	74.2	72.3	48.9
Iterative Training	76.2	74.8	53.5
Iterative Training with Refinement **	**79.1**	**81.1**	**63.9**

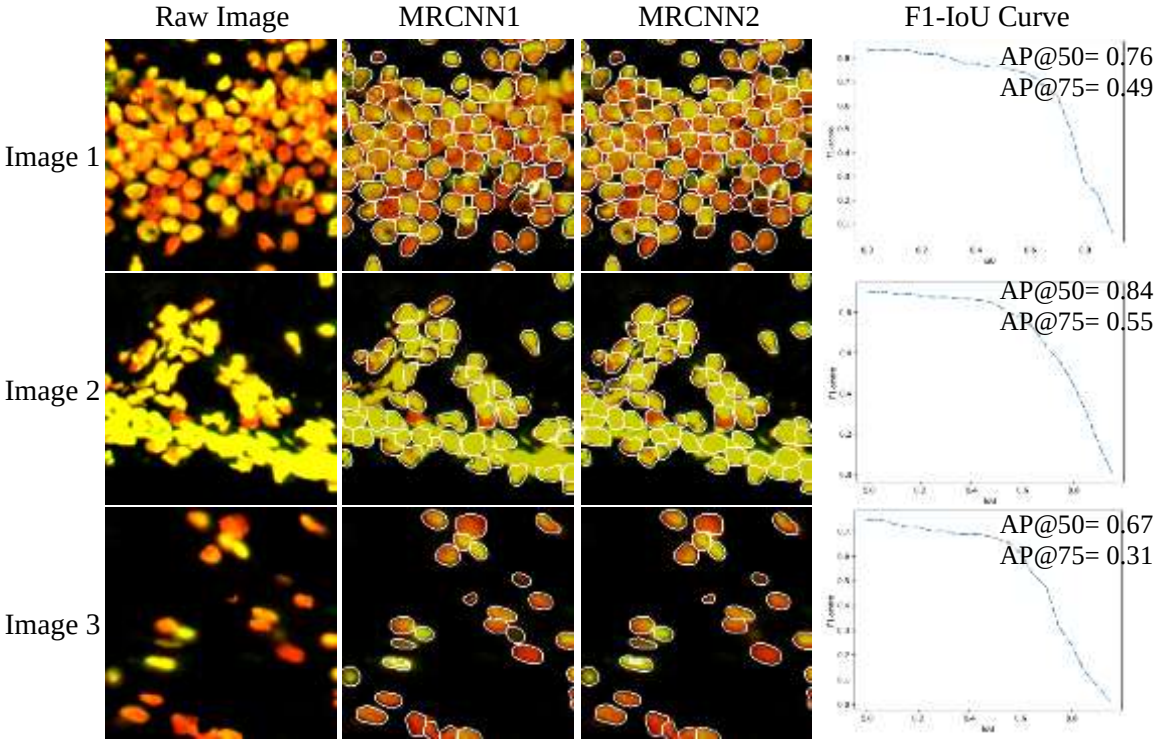


Figure 3-16. Performance on Iterative Training for Three Image Samples.

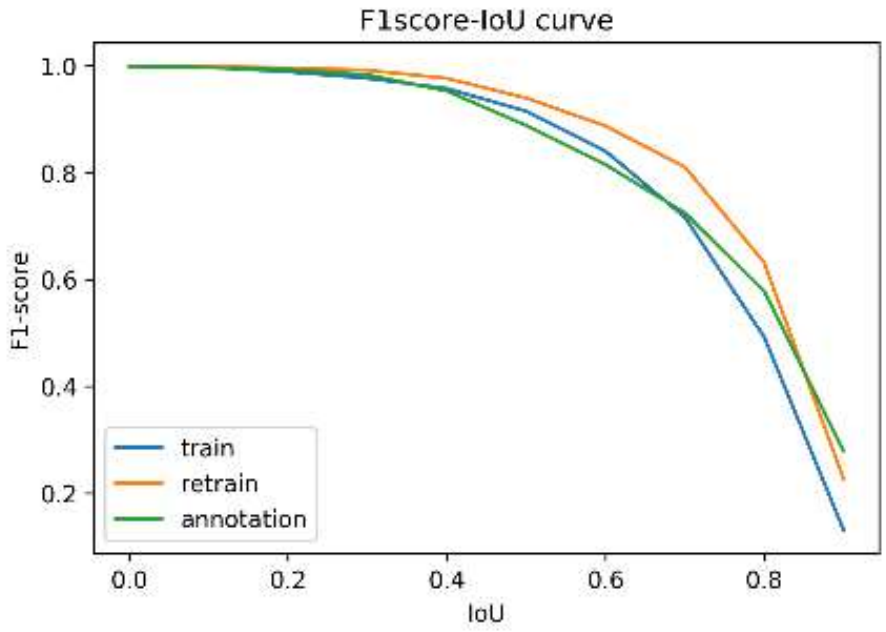


Figure 3-17. Overall F1score-IoU Curve

To evaluate the segmentation performance, we measure the IoU and F1 scores at multiple IoU thresholds, e.g. 0...1. The performance of the parametric methods, including multi-scale LoG with graph-cuts [2], refined by seed-controlled watershed [16], can be seen in the first two rows of **Table 3-1**. The F1-IoU performance over different image samples can be seen in **Figure 3-16**.

The watershed result is used as the noisy labeled annotation input for MRCNN training. As blue line in **Figure 3-17** and the third row in **Table 3-1** shown, directly applying the original MRCNN on the noisy labeled training set sustains a performance drop compared with the training set (green line and the second row). Iterative training and refinement can

primarily help to recover the missing objects and increase the IoU and F_1 performance metrics, as the orange line and the last row in **Figure 3-17** shown.

3.5.4. Performance on Unsupervised Measurements

The segmentation profiling results followed by the matrices in **3.4.2** are shown in **Table 3-2.** Signal Coverage, as the indicator for False Negatives, has improved dramatically by using iteration training pipeline, and further improved by our proposed refinements techniques.

Table 3-2. Unsupervised Performance Metrics Comparisons

Method	Signal Coverage	Split Cell Rate	Low Confidence Rate
Watershed (Noisy Label) *	92.8%	1.6%	0.9%
MRCNN	74.6%	2.8%	1.8%
Iterative Training	87.5%	1.4%	2.5%
Iterative Training with Refinement **	**94.2%**	**1.2%**	**3.5%**

3.5.5. Visualization Results on Large Scale Datasets

The visualization results on whole images are shown in **Figure 3-18** and **Figure 3-19**.

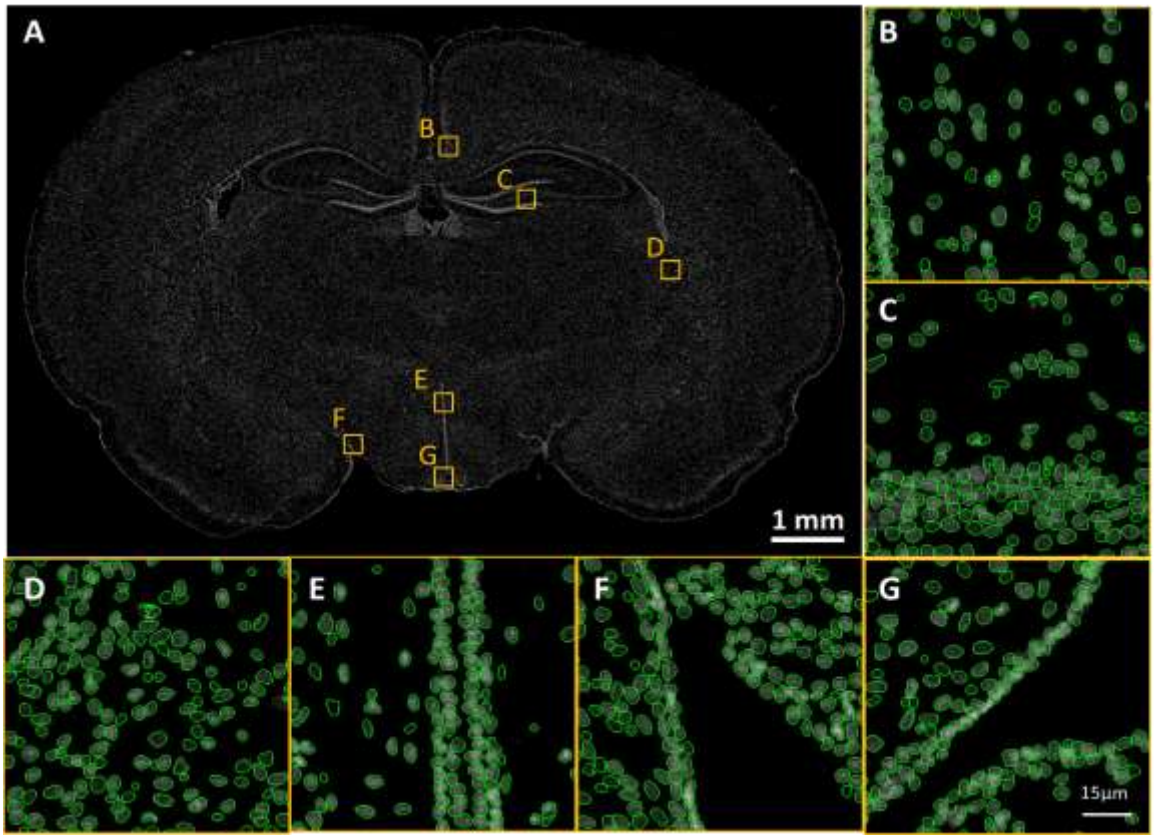


Figure 3-18. Testing Results on Healthy Rat Brain.

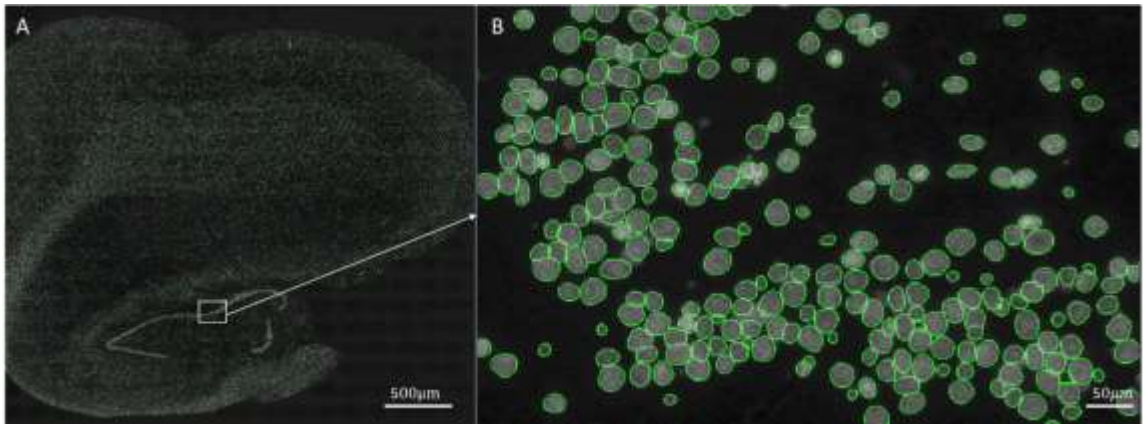


Figure 3-19. Testing Results on Alzheimer Dataset.

3.6 Conclusions

Our pipeline uses whole brain segmented result generated from the conventional computer vision to first-time train the MRCNN on ImageNet pre-trained weights. Then, test it on the same whole brain image to get the first step MRCNN segmentation results. And we further retrain the MRCNN network on the learned weights by using the tested results as the second-time training set. At last, test again on the same images.

Our experiments show that from the first and second time training, the segmentation results have improved apparently. The first training set contains inherent segmentation errors from the conventional parametric method, especially in the regions of hippocampus region where cells are densely packed. Taking advantage of a large amount of correctly segmented cells in sparsely packed regions, MRCNN is able to learn a robust model describing the patterns of nuclei. To analysis the noise patterns the dataset, we outputted the deep feature from MRCNN model intermediate layers. After implemented several state-of-the-art unsupervised clustering and outlier detection methods on the deep features (including K-means, spectral clustering, t-SNE, and hierarchical clustering), there was no obvious cluster showing correlations to segmentation errors. It verified that deep learning is resistance to non-concentrated noise.

We have presented a very practical and effective method for deep training instance segmentation networks for delineating cell nuclei with zero manual annotation data. Even

with the noisy training data, we can demonstrate robust results by modifying the training framework. Our iterative training framework from noisy training data with background Recovery and sparsity-based cell splitting using cues from additional cell-type indicating molecular labels shows a significant improvement, especially in the densely packed regions. Although our experiments have only been performed on MRCNN, a similar technique can be easily applied to other models. Our cell splitting algorithm provides an unsupervised cell splitting verification strategy for an existing segmentation output. It also provides cell classifications as a natural by-product.

3.7 Future Works

Although our fully automatic segmentation pipeline is able to segment the nuclei in the majority of the region, there remains a small number of atypical cell nuclei that are unable to segment correctly. Some concaved “donuts” shaped nuclei cannot be automatically detected and fixed by image filing methods. Some supervised corrections and annotations should solve this problem.

In addition, the cell type mask detection method can also be improved to segment partially overlapped cells. A method like whole cell body reconstruction can be considered.

CHAPTER 4 PHENOTYPE

For the phenotype result generation, we provide both the unsupervised and supervised approaches met the different level of precision requirements and manual effort availability.

4.1 Phenotyping from Segmentation Sparse Decomposition

As described in **3.3.3,** one of the by-products for segmentation decomposition is the estimation of cell type classification. An example of a cropped region of neuron phenotyping results can be seen in **Figure 4-1.** Keep in mind that this result can only be used as estimation since inputs of sparse decomposition are obtained by pixel-level unsupervised clustering, and the clustering result might not be accurate for all cell types.

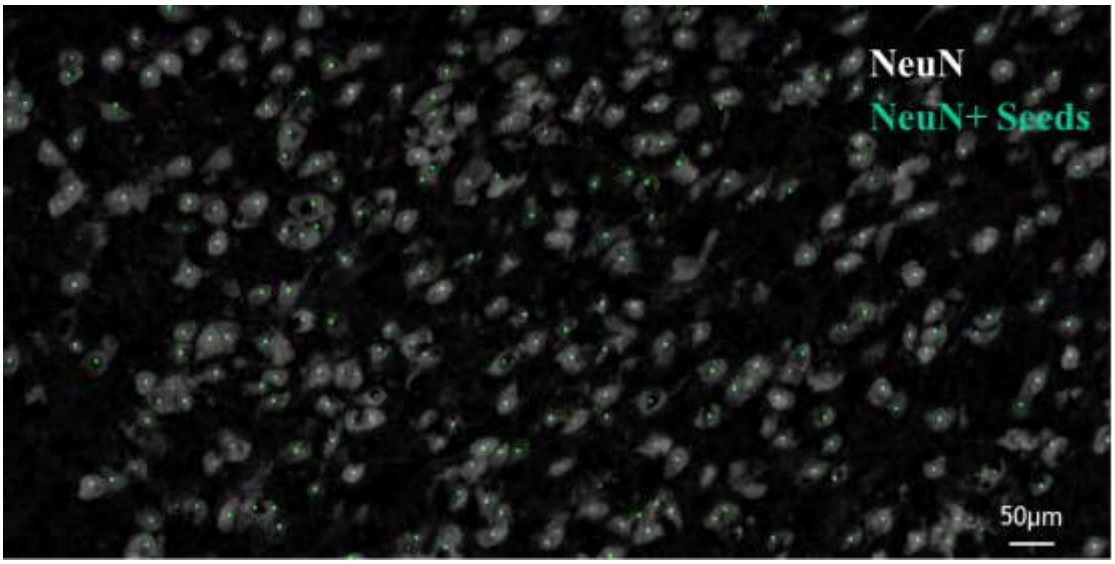


Figure 4-1. Phenotype Result of Neuron Seeds Superimposed on the NeuN Channel.

4.2 Classification with Deep Learning Networks

The output of nuclei detection and segmentation can be used as the input for object-level cell classification. Related works such as cell classification by modified Fast Capsule Net [27] and CNN training with less than 10 samples by meta-learning [28] have led by other members from our lab and collaborators.

4.3 Interactive Gating

To introduce the knowledge of domain experts on the phenotype generating and validation, we convert the segmentation and the feature profiling results into third-party software. Cellular measurements are exported to Flow Cytometry Standard (FCS) and Image Cytometry Experiment (ICE) file formats for visualization and statistical profiling using common commercially available software tools (e.g., FCS Express, De Novo Software; Flow, BD Biosciences, Kaluza, Beckman Coulter, etc.). We provide sample images, staining and imaging protocols, and software in open source form. These data are exported to the cytometry tool FCS Express [29, 30] for visualization.

4.3.1. Flow Cytometry Standard

Flow Cytometry Standard (FCS) is a widely adopted file format in flow cytometry for the reading and writing of data experiments. It can create interactive gating and compatible

with free software, e.g., Kaluza (see **Figure 4-2**). However, FCS is unable to superimpose the cells on raw images and could only achieve gating by histogram of the features.

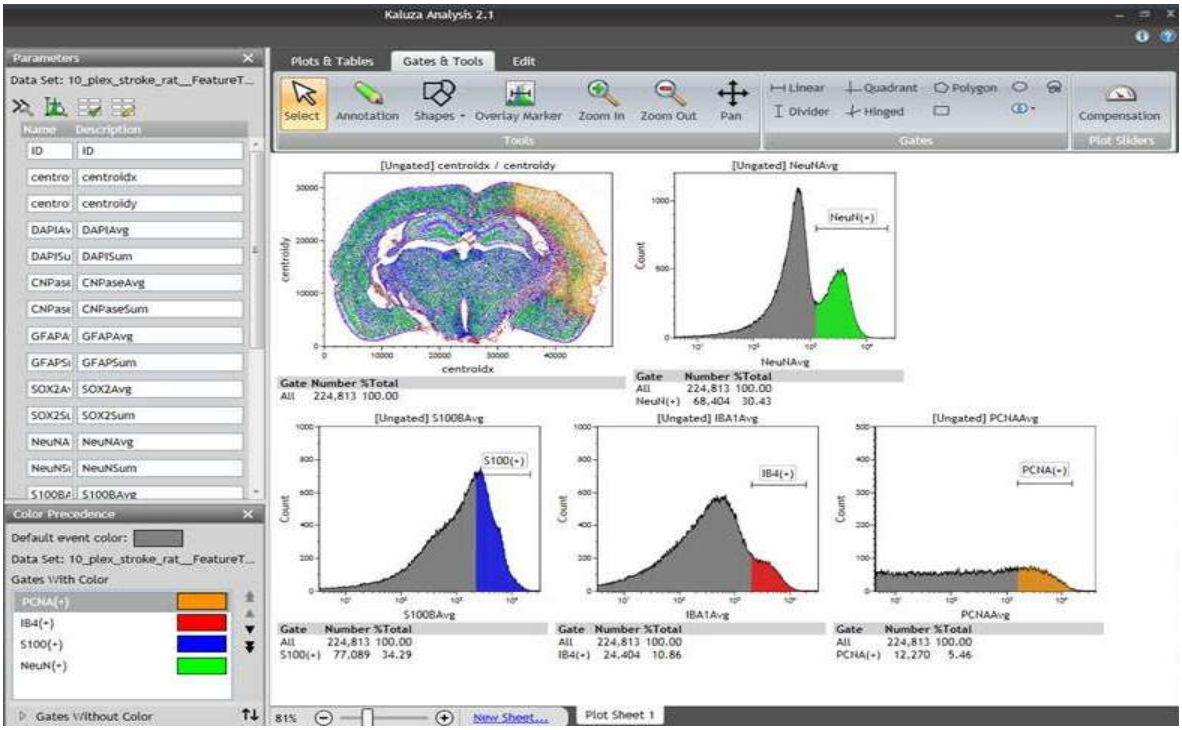


Figure 4-2. Visualize the FCS Outputs on Kaluza Software.

4.3.2. Image Cytometry Experiment

Another standard format is Image Cytometry Experiment (ICE). It can not only achieve interactive gating, but also include all the image and feature information that enabling real-time and interactive validation by associating the original images, cell masks, and intrinsic

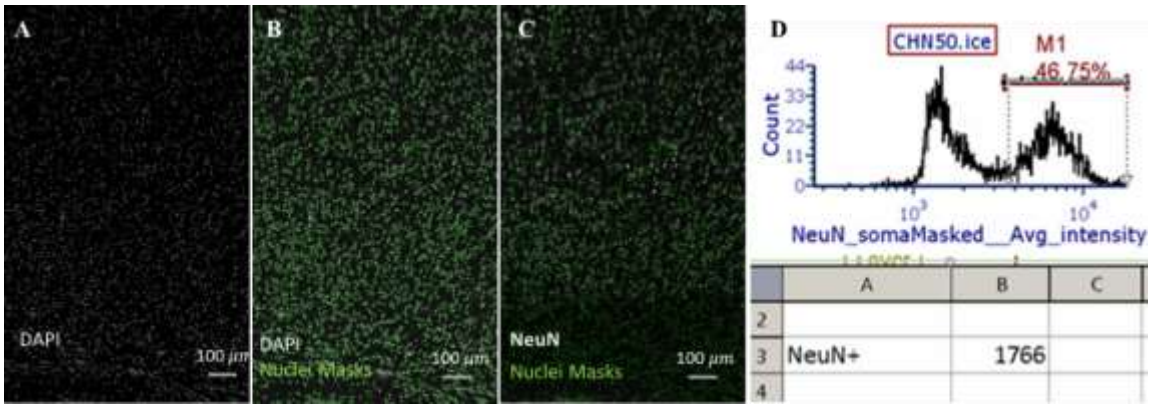


Figure 4-3. Neuron Phenotyping on a Cropped Image by FCS express.

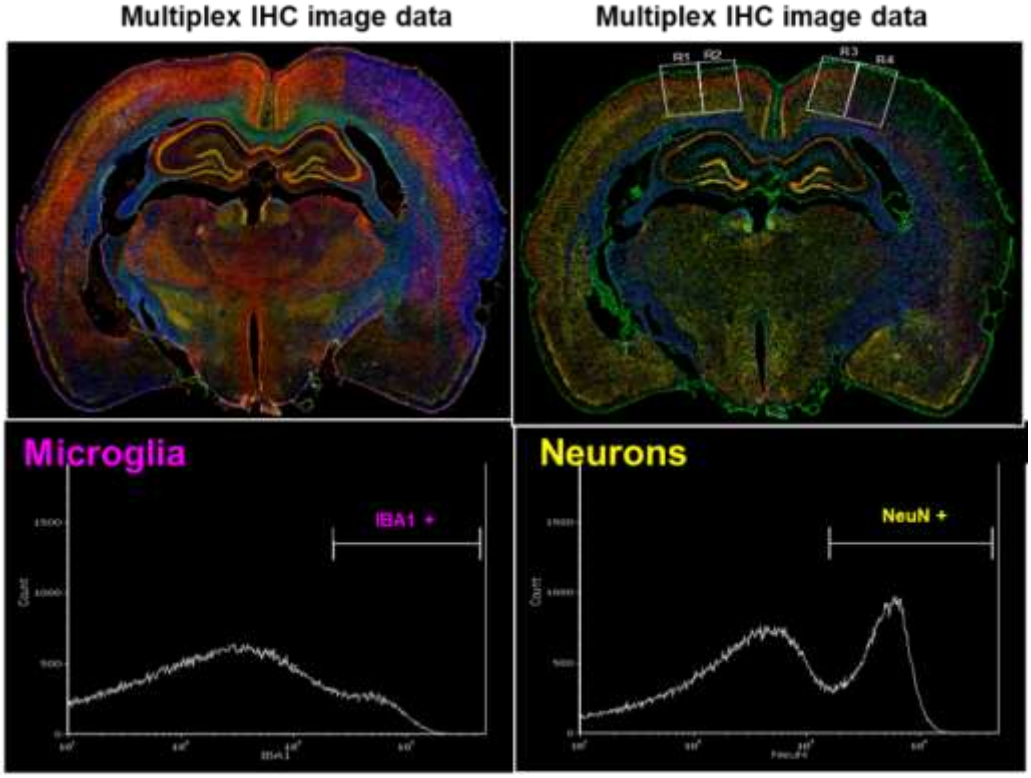


Figure 4-4. Whole Brain Image Interactive Gating.

features. Currently, ICE is only compatible with the commercial software, e.g., FCS express by De Novo Software. An example can be seen in **Figure 4-3.**

The FCS express software is not designed for visualization large scale multiplex images. And it is limited by memory capacity when loading the raw image data. Thus, we implemented image downscaling for the raw images and the spatial information while kept the profiling information as the original precision, as shown in **Figure 4-4.**

CHAPTER 5 CELLULAR NEIGHBORHOOD COLOCATION ANALYSIS

We regard neuron and their neighbour cells as nodes as a form of cell centroid seeds only have spatial positions of and ignore the cell granularity. Based on the hypothesis that cells that are spatially closer to each other in Euclidean distance have higher chance to have functional interactions and influence, we defined our neuron neighbourhood analysis to a fix radius circular region of neurons, in which the radius is represented as “effective range” of neuron. A neighbourhood of a neuron is expected to capture most of the neighbour cells which are functional to the neuron and have less overlap to other neuron neighbourhoods.

We will study the spatial analysis of neuronal neighborhood from two aspects: Colocation analysis in CHAPTER 5 focuses on representation for all neuron-glia cellular combinations. Correlation analysis in CHAPTER 6 focuses on the neighborhood relationship alterations over difference conditions.

Given a collection of spatial features (cell status and centroid coordinates), the co-location pattern discovery process finds the subsets of features frequently located together [31]. In our project, the spatial features are specific types of cell in specific status; the frequent co-location patterns are cellular neighborhood relations in brain. The confidence possibility is the result of neighborhood association rules using the co-location rules algorithm in Reference feature centric model [31], which specifically means that the condition probability that given the existence of neurons in center, how certainty does a

type neighbor cell appear in this neuron neighborhood. One of the Apriori principle algorithms to find out the spatial co-location is implemented in our project based on the Fast mining method [32].

5.1 Introduction

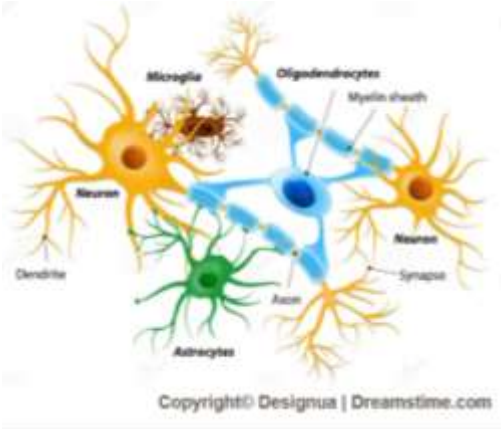


Figure 5-1. Neurons and Neuroglia Cells Neurons and Neuroglia Cells. Glial cells are non-neuronal cells in brain. There are different types of glial cells: oligodendrocyte, microglia, astrocytes and Schwann cells.

The functional network of the central nervous system (CNS) in the brain is primarily constructed by neurons with the support of other cells, specifically glial cells [33]. In vertebrates, microglia, astrocyte, and oligodendrocyte are the three main types of glial cells that have significant functions of supporting and sustaining normal neuronal architecture (**Figure 5-1**). Microglial cells make up the endogenous brain defense and immune system, playing an important role in protecting and supporting neurons [34]. The activation of microglia, following up with immune response molecules producing and dead CNS cells

cleaning, is often regarded as an indicator of a brain insult. Astrocytes express functional receptors for a variety of neurotransmitters and may critically modulate synaptic transmission [35]. They are activated when direct synaptic connections are received from neurons. Oligodendrocyte helps to form myelin and speed up the nerve conduction which is directly connected to the myelin sheath on the axon of neurons. There are also certain non-myelination oligodendrocytes that are related regulation of ionic homeostasis similarly to astrocytes.

5.1.1. Neuron Neighborhood Biological Background

The interaction and association between neurons and glial cells bring us the topic of our project: the local environment of glia neighborhoods of neuron. The local neighborhoods are regarded as the near-distance region in the center of neurons where frequent interactions of neuron-glial are observed [36]. This distance is also called an "effective range(r)", which is the length of activated neighboring cells to neurons. In virtual brain tissue, since there is no strict threshold to determine whether a glial cell is activated or not, heuristic designation of the value of effective distance is normally used. The most common value of effective range could be found in [36], i.e., a value between the average radius of neuron territory (soma) and neuronal intercellular distance (around 32.5 μm)

5.1.2. Dataset Description

The same multiplex imaging of fluorescent channels is conducted on the three animals to achieve comprehensive reconstruction of cyto-architecture in rat's brain. As **Figure 5-2** shown. **(A1 – A10)** are the Close-up images of 10 channels from the 1st and 2nd; **(B1 – B10)** are the rounds of immunohistochemistry. **(C, D)** Reconstructed montages showing APC, DAPI, Iba-1, CC3, S100β, PCNA, RECA1, NeuN, and GFAP for the saline-treated and Li+VPA-treated mild mTBI animals. The yellow arrow indicates the location of a fluid percussion injury that was delivered. **(D)** Magnified view of selected boxed regions of interest showing cellular-scale differences.

Among the 20 channels, DAPI is used to generate cell centroid location and cell masks by nuclei segmentation techniques. Then those masks are cast on 10 soma stained biomarker channels for cell type and cell status phenotyping. The phenotype of each channel is determined by the fluorescence intensity with respect to threshold values which are designated by domain experts by using an interactive gating interface. The shape and size of cells are neglected for our analysis, thus only the coordinates of centroid are used to represent cell location. In general, 2-dimensional cell spatial information and cell status-type property construct the basic input in this project.

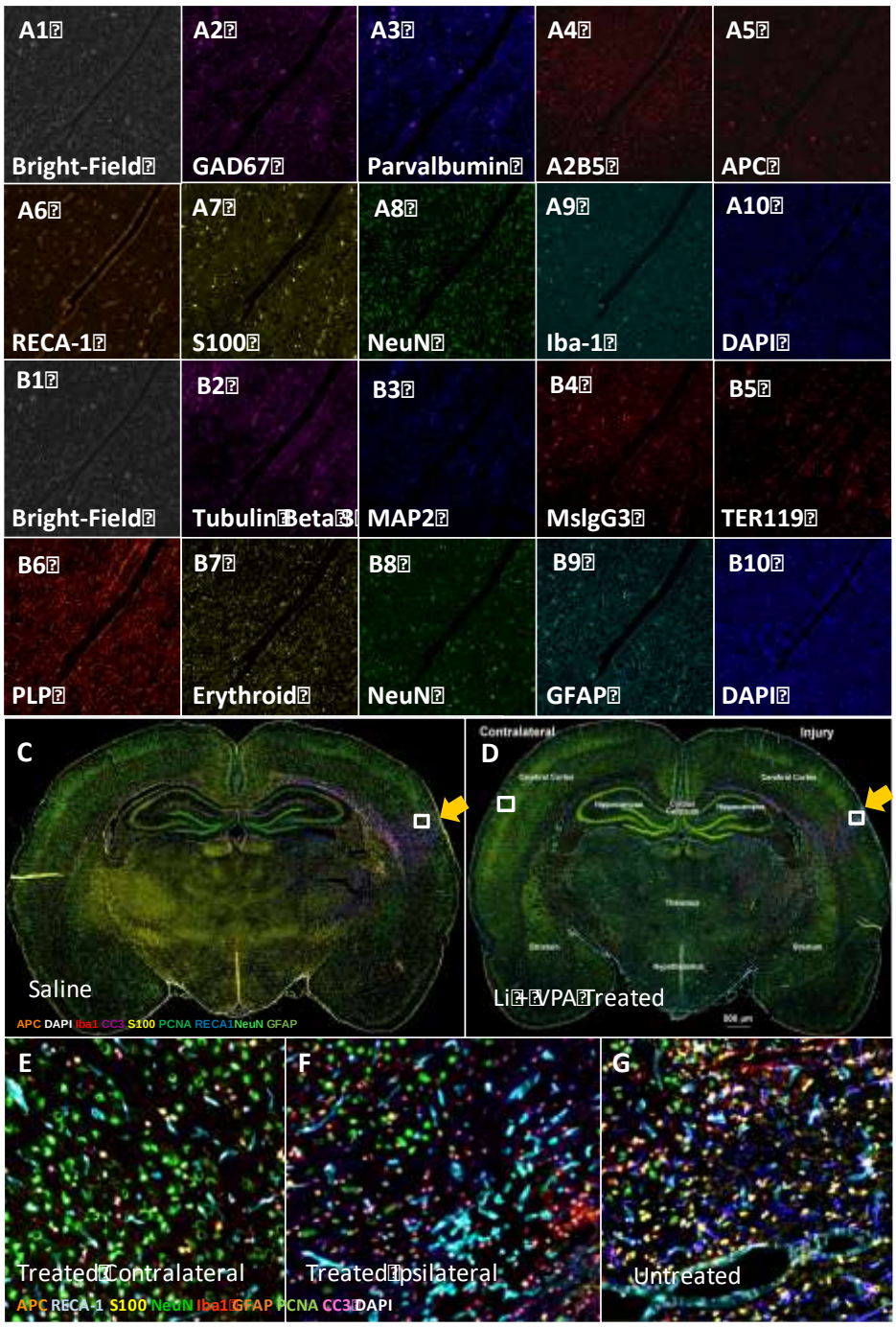


Figure 5-2. Multi-channel Imaging of mFPI-injured Rat Brain.

The item names of **Table 5-1** indicate the types and statuses of neuron and glial cells. One center cell type (neuron) and three neighboring cell types (microglia, astrocyte and oligodendrocyte. Each type of cell has two exclusive cell statues and three common cell statues, five in total. The contents in **Table 5-1** illustrates the Boolean logic of each cell. For example, GABAergic Neurons are the elements from the unit set of (NeuN+) ∩ (GAD67+), and the apoptotic neurons are elements from the unit set of (NeuN+) ∩ (PCNA-)∩ (CC3-).

Table 5-1. Boolean Table of 10 Cell Phenotype Channels

Neurons		Biomarkers	NeuN	GAD67
		GABAergic Neurons	Subset (+)	All (+)
		Non-GABAergic Neurons	All (+)	All (-)
Glial Cell	**Astrocytes**	**Biomarkers**	**S100**	**GFAP**
		Resting astrocytes	All (+)	Subset (low)
		Reactive astrocytes	All (+)	All (high)
	Oligodendrocytes	**Biomarkers**	**S100**	**MBP**
		Myelinating Oligodendrocytes	All (-)	All (+)
		Non-myelinating Oligodendrocytes	All (-)	All (-)
	Microglia	**Biomarkers**	**IBA1**	**Tomato Lectin**
		Resting Microglia	All (+)	All (low)
		Reactive Microglia	All (+)	All (high)
All Cell Types		**Biomarkers**	**PCNA**	**CC3**
		Actively Proliferating Cells	All (+)	All (-)
		Apoptotic Cells	All (-)	All (+)
		Necrotic Cells	All (-)	All (-)

5.1.3. Boolean Spatial Features

Given a collection of Boolean spatial features, the co-location pattern discovery process is devoted to finding the subsets of features frequently located together [31]. In our project, as shown in **Table 5-2** the Boolean spatial features are vectors of specific types of cell in specific status; the frequent co-location patterns are cellular neighborhood relations in brain. Intuitively, the features attaining higher frequency/ importance to appear together are supposed to have more compact neighborhood relations.

Table 5-2. Boolean Spatial Features

ID	Centroid x	Centroid y	Neuron	Microglia	Astrocyte	Oligodendrocyte
1	2	1,263	0	1	0	0
2	2	5,923	0	0	0	0
3	2	810	1	0	0	0
4	2	8,891	0	0	1	0
5	2	3,298	0	0	0	0
…						

5.2 Measure the Attraction of Neuron Neighborhoods

To analysis the neuron neighborhood, we crop a pathological section from the whole rat brain tissue for research of interest as **Figure 5-3** shown. The goal of our project is to find a way to compare the images of three animals (**Figure 5-5. (A1), (B1), (C1)**).

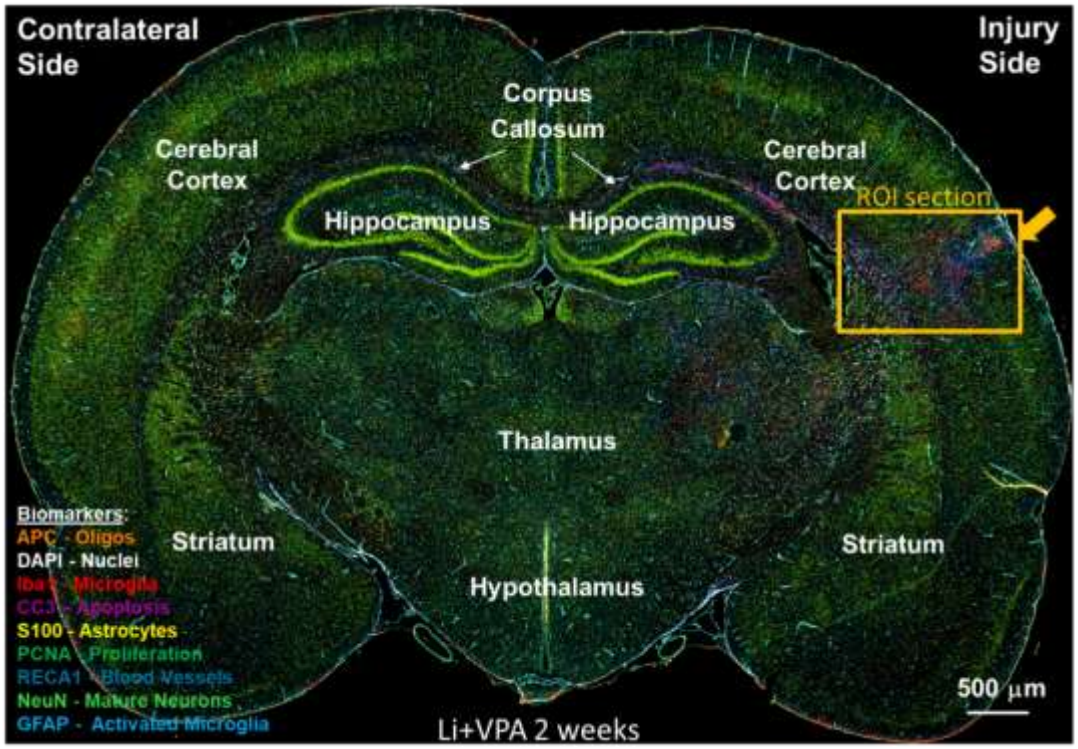


Figure 5-3. Multiplex Fluorescence Immunohistochemistry Illustrates Complex Tissue after Treatment.

5.2.1. Construct Graphical Representation for Local Neighborhoods

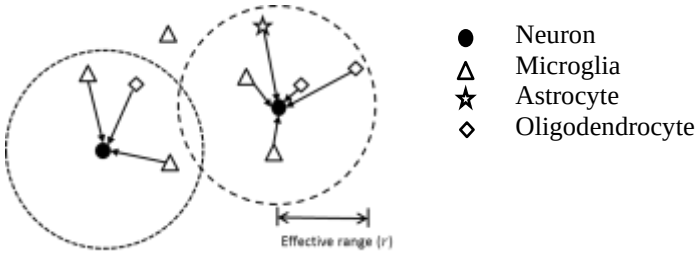


Figure 5-4. The Diagram of Two Neuron Neighborhoods.

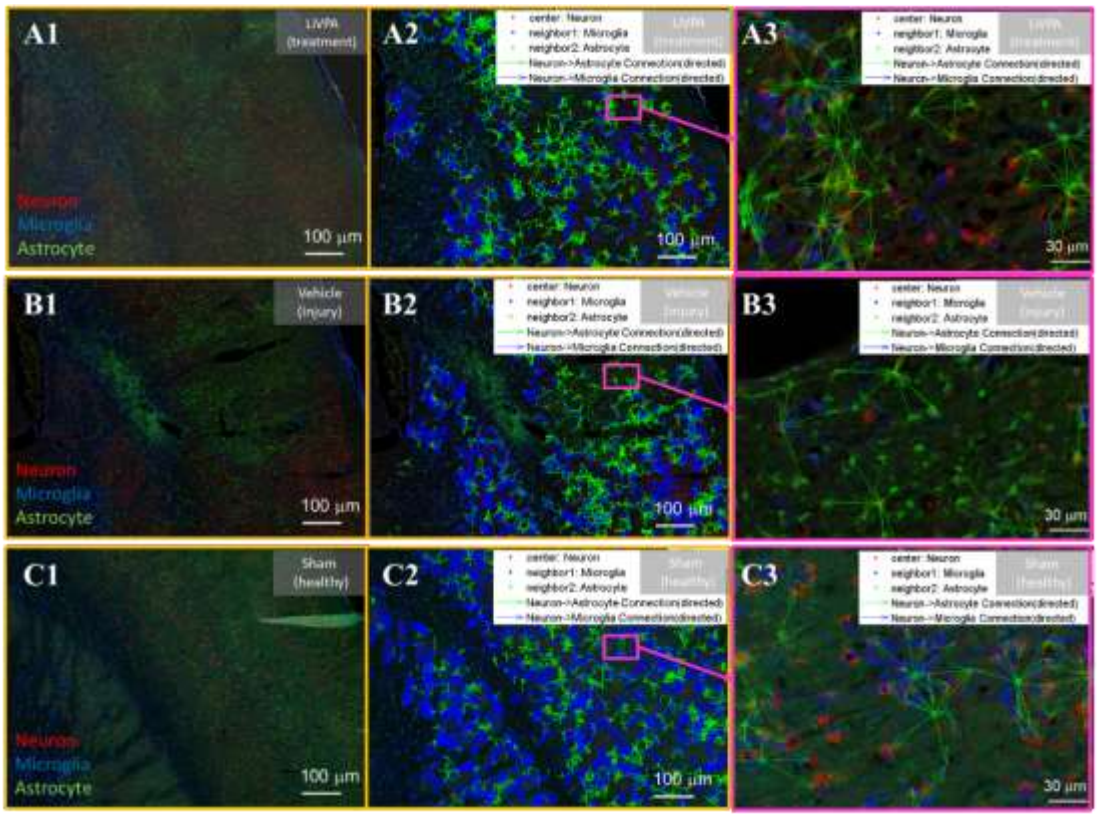


Figure 5-5. Close-ups for the Zoom-in Regions in Figure 5-3.

We use locally connected graphs to represent neuron neighborhoods for the circular region within the "effective range" mentioned in the previous session. A local neighborhood (**Figure 5-4**) is defined as egocentric graph where all the effective neighbor nodes are connected to the center node o. As graphical representations in **Figure 5-5** shown, an effective neighbor point p is connected to a center point o , if and only if their Euclidean distance is less than the designated effective range (r), i.e.,

$$D(p, o) \leq r. \tag{16}$$

5.3 Colocation Rule of Neuronal Neighborhood

5.3.1. Association Rule

The association pattern indicates that an item occurs, another item occurs accordingly. For example, if we observe that customers often buy milk and cereal together in grocery shopping, then we can claim that milk cereal is highly associate with milk.

Association Rule can be used to discover relationship between unrelated data in a data repository, i.e., to find the relationships between the objects which are frequently occur together. Assume there are two types of items $X = \{x_1, x_2 \dots\}$, $Y = \{y_1, y_2 \dots\}$, as shown in **Figure 5-6.**

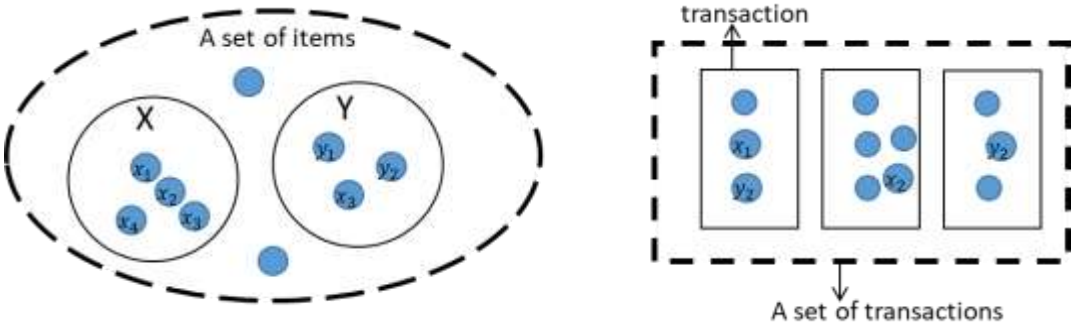


Figure 5-6. Diagrams of Items and Transactions.

A transaction is defined as a set of item types bought together by customers, i.e. the true state of art that elements come out together, as shown in. Prevalence (Pr) is defined as the frequency of all x_i occurs in all transactions:

$$Pr(X) \triangleq \frac{\text{transactions containing } x_i}{\text{number of transactions}}$$

$$\text{support} \triangleq Pr(X \cup Y)$$

$$\text{confidence} \triangleq \frac{Pr(X \cup Y)}{Pr(X)}. \tag{17}$$

Association rule are the pairs of items fit:

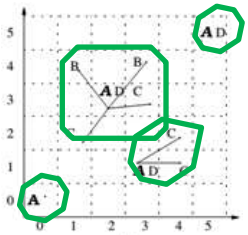
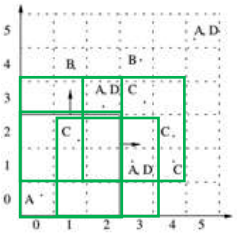
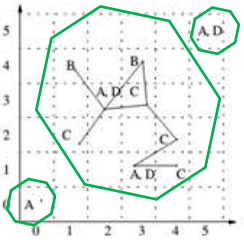
$$\text{Association rule} = \text{items} \begin{cases} \text{support} \geq \text{SUPPORT} \\ \text{confidence} \geq \text{CONFIDENCE} \end{cases}. \tag{18}$$

5.3.2. Colocation Rule

Colocation relationship is the same as association relationship except the transaction is defined by spatial regions. According to [31], there are a few typical models of Co-location, as shown in **Table 5-3**, and the model adopted in our neuronal neighborhood analysis is the "Reference Feature Centric" model where each transactions is a neighborhood centered in a neuron.

Considering that it is time-consuming to generate the transaction and mining co-location rules for such large datasets. We used a fast co-location mining algorithm proposed by Zhang, et al. [32], which enable an efficient way to generate section and mine co-location at the same time, as illustrated in **Algorithm 4** and **Figure 5-7.**

Table 5-3. Illustration of Typical Models of Co-Location

	Local	Reference feature centric	Window centric	Event centric
Co-location Model				
Transactions	partitions of space	instances of reference feature C1 and C2 involved with	possibly infinite set of distinct overlapping windows	neighborhoods of instances of feature types
Interest measures		$Pr\ (C1 \cup C2)$ $Pr\ (C2\|C1)$		
Application		Biology E.g. cancer	Geology E.g. land parcel	Ecology E.g. fire ignition

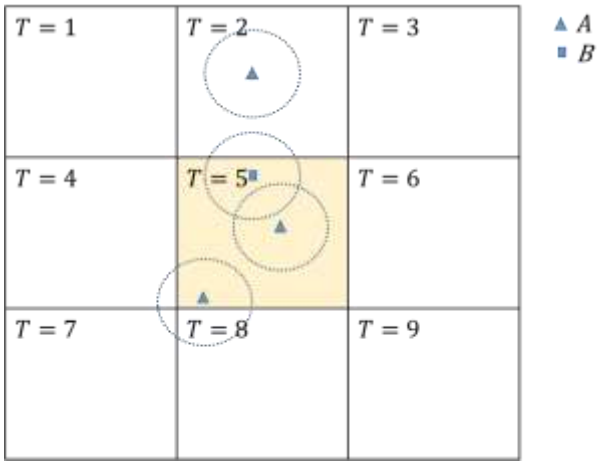


Figure 5-7. Fast Co-location Mining Example

Algorithm 4. Fast Co-location Mining

Input: The complete spatial information of all cell types. The complete map is sectioned in to non over lapped tiles .
Output: Confidence for each types

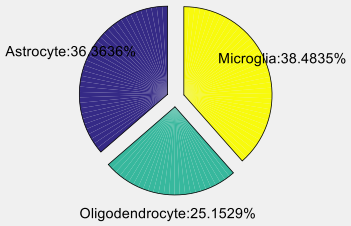
1) ***Spatial-hashing phase***
   for each feature f_i,
      hash the object o into at most 4 partitions.
      Set R_i^T containing objects in R_i hashed into tile T.
      (e.g. for tile 2, $R^2 = \{R_a^2, R_b^2\}$)
2) **Mining phase**
   for each tile T
      for each center feature f_i
         for each neighbor feature f_j
            if an object of f_i and an object of $f_j$are connected
               Confidence index ($f_i \Rightarrow f_j$) $+\mathbf{1}$

5.4 Result Visualization

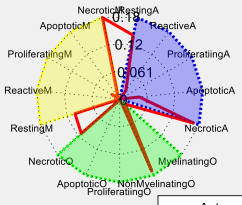
Figure 5-8 depicts neuron neighborhood distribution in the research of interested regions of Vehicle, Sham and Treatment in rat brain, and their radar map with respect to the different states of neuron cells towards neighbor cells. Confidence probability indicates the possibility that given a center cell, a neighbor cell appears at the same time within the neighborhood.

Co-location results in Tissue:vehicle

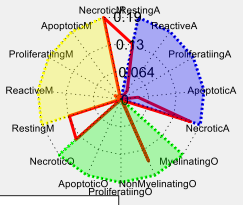
Average neighbor cell distribution in neuron neighborhood (Effective range=32.5)mu m

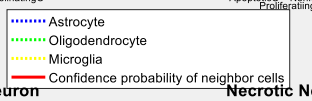


GABAergic Neuron neighborhoods

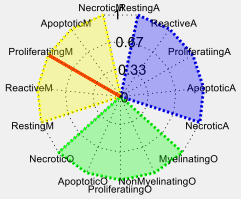


NonGABAergic Neuron neighborhoods

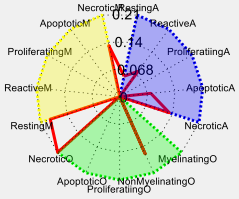




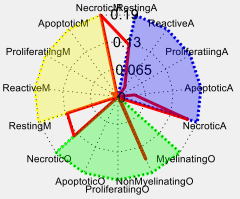
Proliferatiing Neuron neighborhoods



Apoptotic Neuron neighborhoods

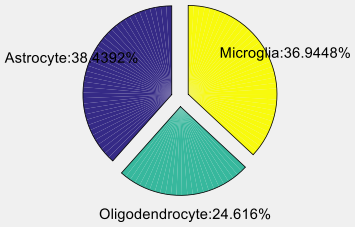


Necrotic Neuron neighborhoods

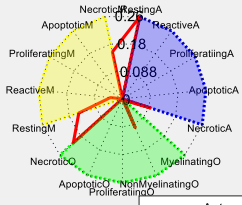


Co-location results in Tissue:sham

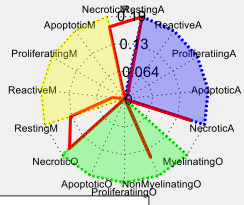
Average neighbor cell distribution in neuron neighborhood (Effective range=32.5)mu m

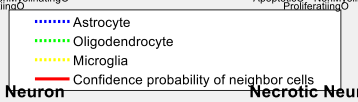


GABAergic Neuron neighborhoods

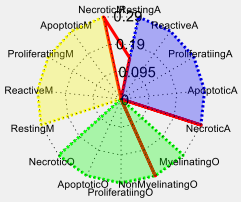


NonGABAergic Neuron neighborhoods

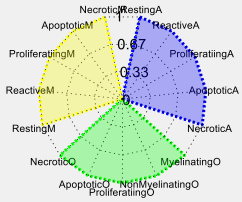




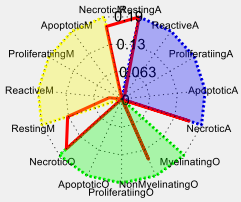
Proliferatiing Neuron neighborhoods



Apoptotic Neuron neighborhoods



Necrotic Neuron neighborhoods



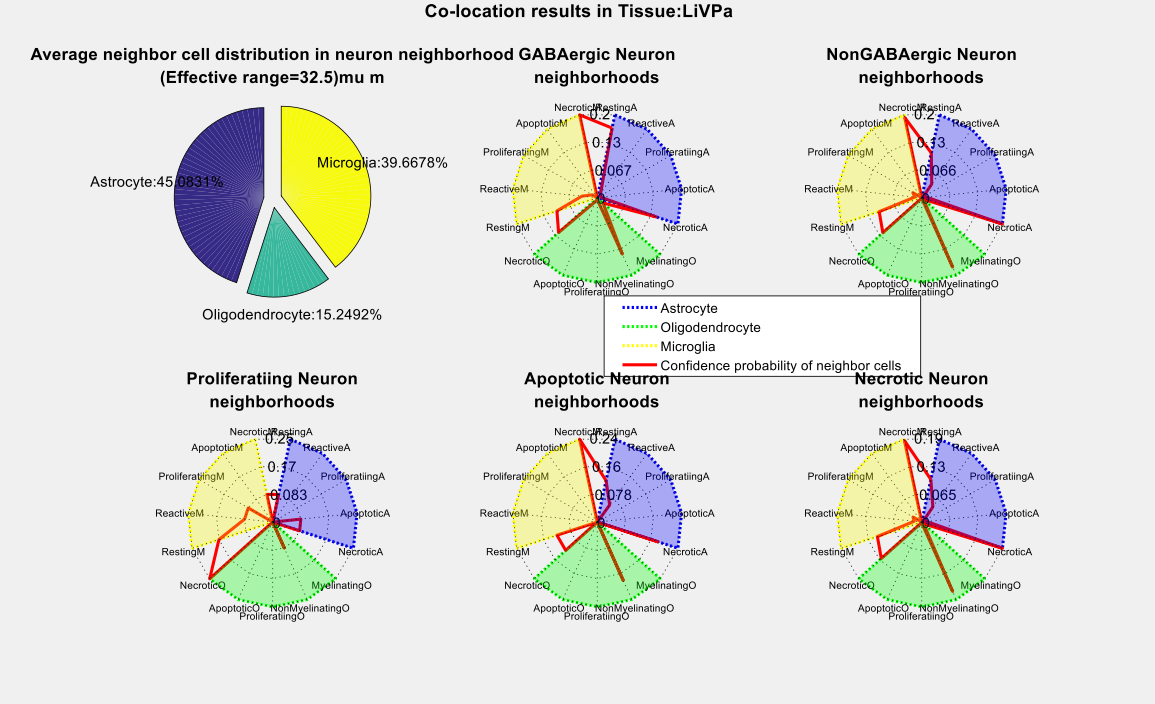


Figure 5-8. The Radar Map Results of Colocation Rules.

5.5 Conclusions

We applied a fast Apriori algorithm to a mining the reference feature centric pattern (star pattern) and co-location rules at the same time. The results reflect the co-appearance relationship of the neuron-glia cell combinations.

CHAPTER 6 CELLULAR NEIGHBORHOOD CORRELATION ANALYSIS

We propose a spatial model to investigate the impact of mild traumatic brain injury (mTBI) and the effectiveness of a potential treatment on glia neighborhoods of neurons. The analysis of neuron spatial neighborhoods is modeled by cumulative influence of all neuron-glial pairs from the same circular surrounding area centered at the neuronal nuclei. The influence of individual neuron-glial cell is quantified by a Gaussian Kernel Density Estimation (KDE) model, which is better than the existing methods in terms of the proximity level from neighbors to centers and robustness of distance parameters. To reveal alterations in neuron neighborhood across different statuses of neuron, we applied sparse group LASSO selection algorithm on three different tissues conditions (injury, normal and treatment) that had been structured by a predefined feature regarding groups of cell statuses. To reveal the impacts of mTBI, we applied a tree- guided group LASSO algorithm which compares the changes of neuron neighborhood across different tissues on the whole dataset organized based on a predefined tree structure regarding cell statuses. Both the two feature selection algorithms can extract individual features and feature groups simultaneously. We have also compared stability and computational cost of the feature selection results over all tissues.

6.1 Introduction

A mild traumatic brain injury (mTBI) is caused by a direct injury to the brain which could be a form a direct percussion, or an explosive blast. mTBI can result in headaches, dizziness, even long-term psychological effect such as depression. The treatment for mTBI is difficult to prove effectively because the changes of pathophysiological brain conditions are often happening concurrently with multiple processes including the primary injury response and secondary alterations. Thus, computational quantification techniques are leveraged to help us have a systematic understanding of the brain condition.

To investigate the impact of mTBI and the effectiveness of a potential treatment on brain tissue, an experiment on a group of rats in the condition of injury without treatment (vehicle), normal (Sham) and injury with treatment (Li+VPa) has been conducted in our project. The injury animal received a fluid percussion injury (FPI) as a type of mTBI ; the Sham-operated animal received all the surgical procedures except FPI; the Li+VPa animal received a combination treatment of lithium and valproate.

6.1.1. Related Works

The methods of cellular neighborhood analysis could generally be divided into two categories: point based analysis and graph-based analysis. In spatial point analysis, cells are regarded as isolated points where the point density and distance between pairwise points are the most important subjects of interest. Arnold [37] uses nearest neighbor

distance and effective radius to measure the cell neighborhoods in order to study the abnormal cyto-architecture in schizophrenia; Morgan [36] extends the single cell type analysis to the interaction of different cell types, yet fails to provide an integrated measurement to model neighborhood pattern. In graph-based analysis, cells are presented as graphical patterns so point based analysis is replaced by graph properties descriptions, proposed by Yener [38]. Yao [39] uses a density weighed distance measurement on the graphical representation of spatial point patterns. Inspired by all these methods, we integrated the density and distance measurements of the neuron neighborhoods by using a single indicator to quantify the neuron -glia influence based on the graphical representation in our project.

Feature selection methods are used to identify the features that have the most significant impact over different classes in a classification problem. Least Absolute Shrinkage and Selection Operator (LASSO) is a common feature selection method for classification problems which allows the coefficients of insignificant features to be zeros [40]. Some derived forms of LASSO such as group LASSO, sparse group LASSO and tree- guided group LASSO [40] can handle hierarchical structures, and the latter two can extract individual and groups features simultaneously. These feature extractors are generally applied on biologically related tasks like brain disease classification using tree LASSO [41], multi-layer projection via hierarchical projection [42], feature selection for clinical disease using tree LASSO [43], and localization of multi-label proteins using group

LASSO [44]. Compared to all the applications of pathological analysis mentioned above, we are the first to implement the hierarchical feature selection on intercellular neighborhood analysis.

6.1.2. Challenges

Currently there is no biological definition that describes the size and range of neuron neighborhoods accurately and quantitatively. Moreover, neurons and glial cells are not distributed homogeneously in different regions of the brain. For these reasons, we will explain why our proposed model is robust to the slight changes in the size of the neighborhood and fits the uneven distribution of cells. Traditional feature selection methods fail to perform well in scenarios where features have hierarchical structure property. Then, we will explain why sparse group LASSO and tree LASSO the most appropriate feature selection methods for our application are.

6.1.3. Contributions

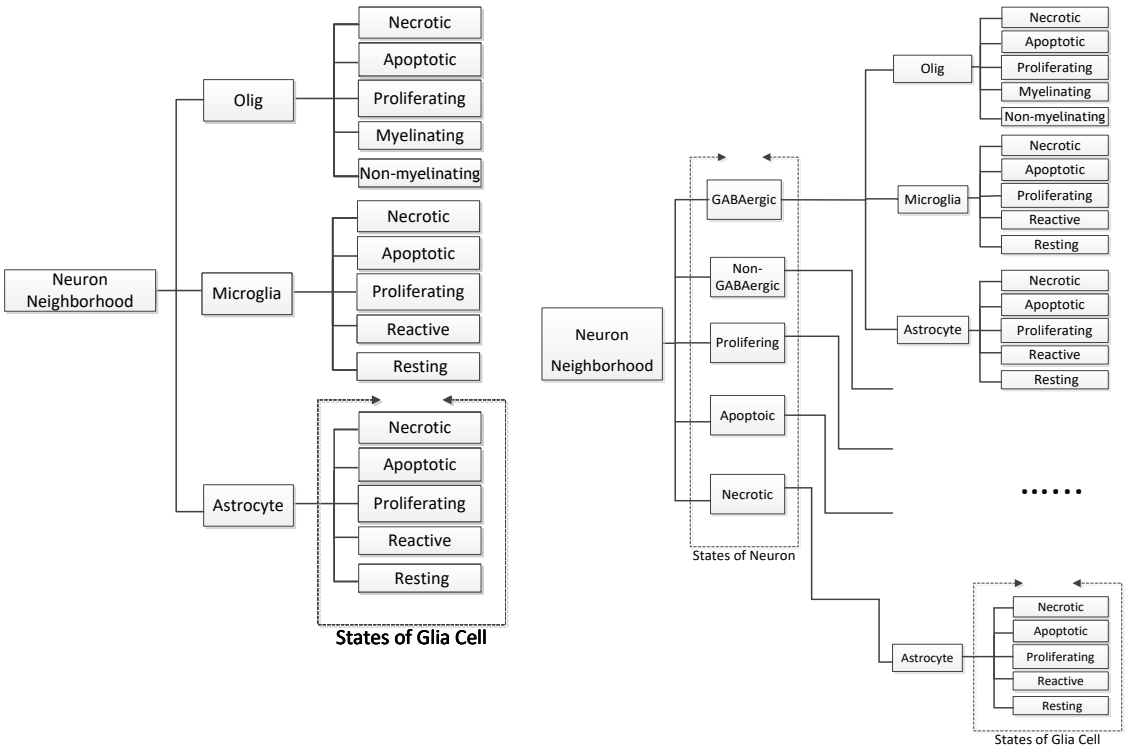
Our task is to develop a method to quantify the impact of mTBI and the effectiveness that a potential treatment has on glial neighborhoods. These impacts are, in particular, the alterations of neuron spatial neighborhoods across different statuses of neurons, and different pathological conditions of tissues (normal, injury and treatment). The neuron spatial neighborhoods are measured by the cumulative influence of all neuron-glial pairs

in the same neuron neighborhood region, in which the influence of individual neuron-glial cell is quantified by a Gaussian Kernel Density Estimation (KDE) model. Then, we applied sparse group LASSO and tree LASSO on the neuron spatial neighborhood to select the intrinsic and hierarchically structured features which in our case defined as the hierarchy of glial cell type and glial cell status. In the end, a stability test is taken to evaluate the consistency of the feature selection across the whole dataset.

6.2 Graphical Representation of Neuron neighborhood

6.2.1. Hierarchical Structure of Neuron Neighborhood

According to the center and neighboring cells definitions and their known states, we construct a cell status hierarchy. As **Figure 6-1** shown, Neurons have five states as the subgroups, and within the neighborhood of neuron, each glial neighboring cell has five states as their subgroup. **(A)** defines the group structure of neuron neighborhood when only the states of glial cells are discriminated. This structure aims to reveal alterations of neuron neighborhood across different statuses of neurons. **(B)** defines a sophisticated form of neuron neighborhood when the states of both neurons and glial cells are considered. By using this structure, we can reveal the impacts of mTBI by comparing neuron neighborhood changes across different tissues.



(A) (B)

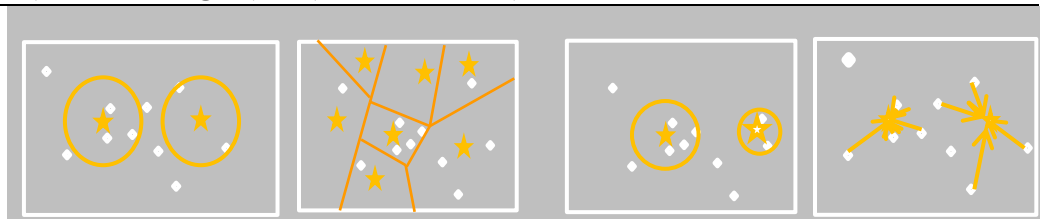
Figure 6-1. Cell Status Hierarchy of Two Kind of Neuronal Neighborhood.

6.2.2. Limitations of Common Feature Representations

There are a couple of research work focusing on the neuron-microglia neighborhoods analysis. Wake, et al. discovered that the states microglia changed in physiological effects [35], whereas didn't consider the distribution alternations of each states. Morgan, et al. quantified the number of neighbors in a fix-radius-neighborhood of each neurons [36], whereas didn't consider the distance between microglia and neurons may change over different area of tissue. Zhou, et al. measured the neighborhood relation by detecting their

clustering coefficient which indicates how compactly neurons are surrounded by microglia [45], however it failed to identify the local pattern when microglia are evenly distributed throughout neurons.

Table 6-1. Common Neighborhood Measurement and their Limitations

	Num of neighbors (Fix radius region)	**Num of neighbors (Voronoi section)**	**Cluster Distance**	**KNN Distance**
Gener-al view				
Aim	Show the density in fixed radius-area of neighborhoods	Separate territory of neighbor cell(15-30 μm wide)	Show how neighbors compactly gather to centers	Mean distance of K-nearest Neighborhood links
Limit-ations	There would be some points that do not belong to any territory; neighborhood territory of centers may overlap	Actual neighborhood territory area of neighbor cell may not vary a lot from each other	Fails to identify the local pattern when neighbor cell is evenly/ randomly distributed throughout neurons	Some points are not connected to any centers; A single neighbor may link to more than one center

To measure the neuron-glial influence in neuron neighborhood, density is the most intuitive way, calculated by the division of the number of neighboring cells to the area of neighborhood. However, it requires the spatial space to have the homogeneous distribution which rarely fits the cell distribution in brain tissue. In addition, density neglects their proximity level to centers, i.e. the direction and distance to centers are not considered. Moreover, it is not robust to the distance parameter. When the distance threshold changes

a little, the neighboring cells located in the margin of neighborhood cause large impact on the result. The other commonly used measurements also have the limitation shown in **Table 6-1**.

6.2.3. Measure the Attraction of Neuron Neighborhood

To overcome those limitations of single measurements, we use a Gaussian Kernel Density Estimation (KDE) model to quantify the attraction measurement based on the fact that the long distance between center cell and neighboring cell, the less possibility that neighboring cell reacts to center cell. Recall a normal Gaussian kernel function, the density estimation of an instance o can be represented as

$$f(o_x) = \frac{1}{r^2}\sum_{i=1}^{t} \exp\left(-\frac{dst(0_x,0_i)^2}{2r}\right). \quad (19)$$

The value of the function gradually decreases from the peak to edge and reaches 0 at the boundary of a circle with the center of o_i and radius of r ($r = 3$ in the figure). Likewise, for an individual neuron-glial pair, a distant weight (DW) from a neighbor point p_j to a center point o_i, where the value of effective range equal to r, is defined as

$$DW(p_j \rightarrow o_i) = \exp\left(-\frac{dis(p_j,o_i)^2}{2r^2}\right). \quad (20)$$

Some properties of the kernel density based distant weight can be seen from an example in **Figure 6-2.** First, DW is robust to distance parameter: on the one hand, the long distance

neuron-glial influence pairs $A_3 \rightarrow B_1$, $A_6 \rightarrow B_1$ are excluded by the distance threshold value r; on the other hand, the minor variance of distant threshold causes the least influence on the result because the DW value near the border is low. Second, the proximity level of neighbor to the center is considered for each effective pair. Last, the overlapping of two neighborhoods are tolerant, i.e. one neighbor instance could affect to more than one center.

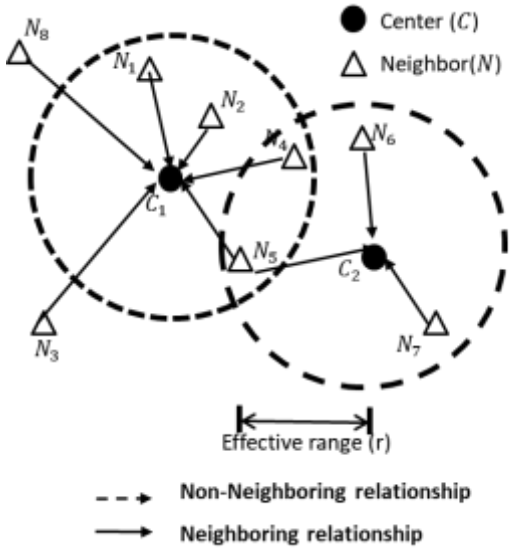


Figure 6-2. Example of Calculating the Distance Weights over all Neuron-glial Pairs.

Since the accumulation of all neuron-glial pairs from the same center forms the neuron neighborhood measurement of the center, the kernelled neighborhood attraction (KNA) of a center o_i is calculated as

$$KNA_i(o) = \frac{\sum_{p \neq o_i} DW(p \rightarrow o)}{N_p}. \quad (21)$$

Continued with same neighborhood graph from **Figure 6-2**, the neighborhood attraction (KNA_i) of two center points B_1 and B_2 is calculated as shown in the third column in **Table 6-2**. Comparing Neighborhood Attraction with other neighborhood measurements, *KNA* is a measurement containing both the information of neighbors' intensity and their distance to centers. Additionally, the result of $KNA_2 > KNA_1$ is positive related to the fact that center B_2 attracts more neighbors than B_1 in their vicinity neighborhoods.

Table 6-2. Comparing the Neighborhood Attraction with Two Other Measurements

Center ID	Number of Neighbors	Average Distance	Kernelled Neighborhood Attraction (KNA_i)
B_1	4	6.85	2.76
B_2	5	6.2	3.67

Therefore, neighborhood attraction is an appropriate measure to quantify neuron neighborhoods.

6.3 Supervised Feature Selection with Respect to Different Classes

In a classification problem, significant features are a subset of features that change most across different classes. Selecting the significant features helps understand the

classification model and trim the irrelevant and redundant features. For example, it is useful to find the pathogenic indicators from a set of common physical indicator for efficient disease diagnosis. In this capture, we will demonstrate two supervised feature selection techniques for features with hierarchical structure: sparse group LASSO and tree LASSO.

6.3.1. Using LASSO to Select Features

Least Absolute Shrinkage and Selection Operator (LASSO) is a common feature selection method for classification problems which enables the feature selection results to have most sparsity [40]. Given a learning model to classify objects from a set of features:

$$y = Xw + \varepsilon \, . \tag{22}$$

X is the dataset contacting N samples and M dimensions with the matrix size of $M \times N$, y is the label vector with size of $N \times 1$ and w is the regression coefficient vector assigned to the respective features. The feature selection works to discard the insignificant features with their coefficients close to zero.

In order to solve w as a classification weights for features, we use the least square optimization as the most popular method with the combination to the L_1 term of penalty,

$$\mathrm{w} = \underset{w}{\operatorname{argmin}}\left(\frac{1}{2}\,\|y - Xw\|_2^2 + \lambda\|w\|_1\right) \ , \lambda > 0. \tag{23}$$

λ is defined as a regularization parameter controlling the sparsity of the features. By comparing the penalty form of Ridge regression in L_2 norm and LASSO in L_1 norm as shown in **Figure 6-3**, LASSO is a better solution for this optimization function because it leads to the smallest subset of a non-zero coefficient for a convex optimization problem.

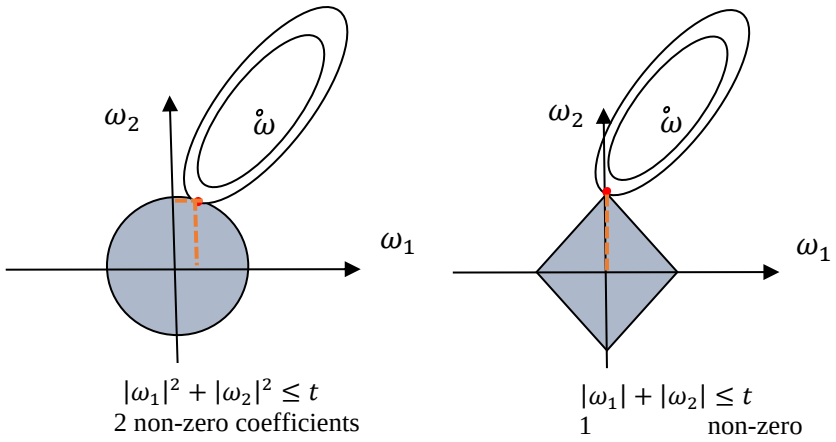


Figure 6-3. Ridge Regression in L_2 Norm and LASSO in L_1 Norm Penalty

The solution of the **Equation**(**23** does not have a closed form, so it is generally solved by iteration of coordinate-wise minimization [46].

6.3.2. Using Sparse Group-LASSO to Select Features

The general form of LASSO is unable to fit the natural group structure (**Figure 6-1**) of the features in our project. So we introduce group LASSO derived from LASSO to solve

this problem, which forces all the features in the group selected or unselected as the same time as can be seen in **Figure 6-4**.

Figure 6-4 . Comparing of LASSO, Group LASSO and Sparse Group LASSO. Each cell denotes a feature and light color represents the corresponding cell with coefficient zero.

Consider g disjointed groups of covariates form the structure of the features, where each group $i = 1 \dots g$ weighted by h_i. The group LASSO solves a convex problem

$$w = \underset{w}{\operatorname{argmin}} \left(\frac{1}{2} \|y - Xw\|_2^2 + \lambda \sum_{i=1}^{g} h_i \left\|w_{G_i}\right\|_p \right), \lambda > 0. \quad (24)$$

Compared to ordinary LASSO, the group LASSO simply replaces the L_1 norm of w with the L_p norm of the weighted coefficients in each group. It enables either the entire vector w_{G_i} to be zeroes or all to be nonzero.

Once a group of features has selected, one also needs to select the individual feature within the group simultaneously. By taking advantages of both LASSO and group LASSO, sparse group LASSO uses a sparsity parameter to control the between-group and within-group sparsity,

$$w = \underset{w}{\operatorname{argmin}} \left(\frac{1}{2} \|y - Xw\|_2^2 + \lambda \left[(1-\alpha)\|w\|_1 + \quad \alpha \sum_{i=1}^{g} h_i \left\| w_{G_i} \right\|_p \right] \right),$$
$$\lambda > 0, \;\; 0 < \alpha < 1 \,, \tag{25}$$

where $(1-\alpha)$ controls the feature level selection and α controls the group level selection. Group LASSO would be regarded as a special form of sparse group LASSO when $\alpha = 1$, and LASSO would be regarded as a special form of sparse group LASSO when $\alpha = 0$.

6.3.3. Using Tree-LASSO to Select Features

Except for group structure, there are also some cases when features have hierarchy structure as required in our project. Tree-guides group LASSO [40] is proposed for features denoted by an index tree. For an index tree G with the depth of 3, G_j^i represent the index of the j^{th} node in the i^{th} depth, where each depth of the tree has n_i nodes.

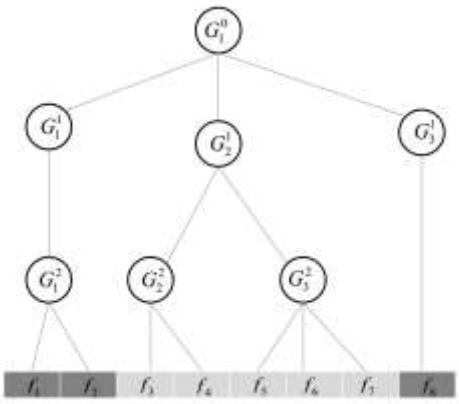


Figure 6-5. Illustration of Tree Guided Group LASSO. The index tree has depth of 3 with 8 features [42].

Take an example of **Figure 6-5**, the G_j^i s are

$$G_1^0 = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\},$$

$$G_1^1 = \{f_1, f_2\}, G_2^1 = \{f_3, f_4, f_5, f_6, f_7\}, G_3^1 = \{f_8\},$$

$$G_1^2 = \{f_1, f_2\}, G_2^2 = \{f_3, f_4\}, G_3^2 = \{f_5, f_6, f_7\}, G_3^1 = \{f_8\}. \quad (26)$$

An index tree should fit the requirements: the root node G_1^0 should contain all the bottom nodes; the index of each nodes should not overlap to each other.

After the index tree has defined, the tree guided group LASSO regularization is

$$\min_w \frac{1}{2} \|y - Xw\|_2^2 + \lambda \sum_{i=0}^{d} \sum_{j=1}^{n_i} h_j^i \left\| w_{G_j^i} \right\|_q. \quad (27)$$

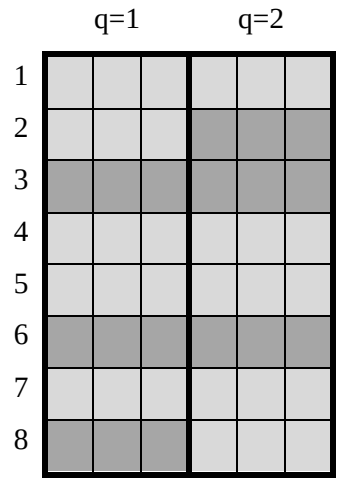
Group LASSO regard as a special case of overlapping group LASSO, which enabled the feature under specific nodes to be selected comparing to the group selection in general group LASSO. Note that tree LASSO could degenerate to group LASSO if $d = 0$, and could further degenerate to LASSO if $d = 0$ and $n_i = 0$.

6.4 Feature Selection Measurements

To measure the result of feature selection, we raise an aspect of the robustness of selected features often neglected by people. The stability of a feature selection method is to quantify the selection result changing over all distribution of dataset. It could be used to either compare the method robustness over different methods or the consistency of selected features in one dataset. To take the latter one for our research, "stability test" is defined as measuring the feature selection consistency across the feature sets produced with slightly differing training set, drawn from the same distribution [47].

Take an example in **Table 6-3**, an eight-dimensional dataset is divided into two subsets $q = 1\ and\ q = 2$. The same LASSO regularization is applied on the subset where their LASSO coefficients have slightly different values as the left table illustrated. Two feature selections are implemented by picking the $K = 3$ features with the top K value of LASSO coefficients. There are two ways to denote an order of feature coefficient: the index (p) of the original features and the rank(r) of the coefficient value (from the highest to the lowest), followed by two measurements to assess the feature selection algorithms.

Table 6-3. An Example of Indexes and Ranks of LASSO Coefficients Result



	Subset q=1							
LASSO coefficient	0.1	2.0	6.0	0.2	0.6	7.0	1.0	3.0
Index(p)	1	2	3	4	5	6	7	8
Rank(r)	8	4	2	7	6	1	5	3
	Subset q=2							
LASSO coefficient	0.2	5.0	7.8	1.0	1.2	4.4	1.1	2.0
Index(p)	1	2	3	4	5	6	7	8
Rank(r)	8	2	1	7	5	3	6	4

6.4.1. Jaccard Similarity Measure

Jaccard Similarity Measurement (JSM) is a typical measurement to measure the feature selection stability by observing the feature index changes **Table 6-3**. Assuming the index of Q sub-samples of training data is given by $q = 1, \dots, Q$, then a subset ta features could be noted as $S = (S_1, S_2, \dots S_q), S_i \in \{0,1\}$, where 1 stands for select features and 0 stands for unselected features. A Jaccard index J is a metric to measure the similarity between 2 subsets of features (S_q and $S_{q'}$)

$$J(S_q, S_{q'}) = \frac{|S_q \cap S_{q'}|}{|S_q \cup S_{q'}|}. \tag{28}$$

For example, in **Table 6-3**, the index 3,6,8 in $subeset\ S_1$ and the index 2,3,6 in $subeset\ S_2$ are selected , so the Jaccard matrix of $J(S_1, S_2)$ =2/4 = 1.5.

The Jaccard similarity measurement based on Jaccard Index is defined as the average of Jaccard indices over each pair of feature sets

$$JSM = \frac{2}{Q(Q-1)} \sum_{q=1}^{Q-1} \sum_{q'=q+1}^{Q} J(S_q, S_{q'}). \quad (29)$$

Noted that the value of JSM is between 0 and 1. The more the value is close to 1, the more stable the feature selection is over all pairs of Q sub-samples in the data set in terms of the feature index.

6.4.2. Spearman's Rank Correlation Coefficient

Spearman's Rank Correlation Coefficient (SRCC) is set to measure the feature selection stability by observing the feature rank changes [46] . The vector of ranking r over all features by sorting the feature selection coefficients is denoted as $r = r_1, r_2 \dots, r_p$, where p is the total number of features. Spearman's rank correlation coefficient is a rank-based similarity between 2 ranking r and $'$, i.e.,

$$SRCC(r, r') = 1 - 6 \sum_{j=1}^{p} \frac{r_j - r_j'}{p\,(p^2-1)}. \quad (30)$$

For example in **Table 6-3**, the rank of the first subset is r = {8,4,2,7,6,1,5,3}, the second subset r' = {8,2,1,7,5,3,6,4}.

The possible range of values for SRCC is between -1 and 1, where 1 shows that two rankings are identical, 0 shows that there is no correlation between two rankings and -1 shows that rankings are in a reverse order.

Similar to JSM, the SRCC is also applied on all the possible pairs in Q sub-samples, then the final form of SRCC is

$$SRCC = \frac{2}{Q(Q-1)} \sum_{q=1}^{Q-1} \sum_{q'=q+1}^{Q} SRCC(r_q, r_{q'}) . \quad (31)$$

The more the SRCC is closer to 1, the more the whole data set is stable in terms of ranking.

6.5 Results Evaluations

To figure out the alterations of neuron neighborhood across different statuses of neuron and different tissues, we constructed the neuron neighborhood graphical representation and selected the features from the locally connected graphs (**Figure 5-3. (B2), (C2), (D2)**). Note that the number of samples is denoted as N, and the dimension is denoted as D.

6.5.1. Discussions on the KNA Parameters

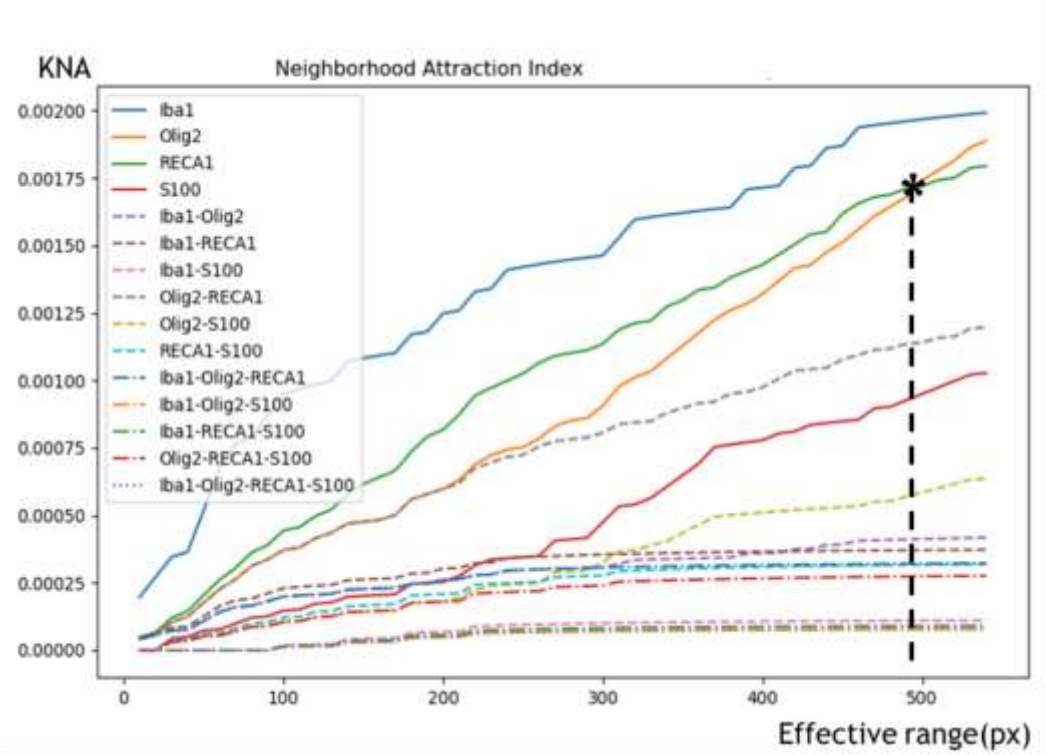


Figure 6-6. Kernelled Neuronal Neighborhood Attraction over the Whole Brain.

To profile the neighborhood representation by our proposed measurement, we calculate the KNA values over all neighboring cell types and effective ranges. The run time for calculation on ~200,000 cells only needs the 20s. Also, compared to other methods, we can monitor one center type's neighborhood pattern to multiple neighbor types.

As shown in **Figure 6-6**, over all the four neighbor cell types represented by IBA1, Olig2, RECA1 and S100, the following subpopulation groups have the most impact in vicinity neuronal neighborhoods, i.e., the effective range less than 200px:

1) In one-neighbor neighborhood scenarios, microglia have the largest KNA value;
2) In two-neighbor neighborhood scenarios, Oligodendrocyte-endothelial cell combination has the largest KNA value;
3) In three-neighbor neighborhood scenarios, the Microglia-oligodendrocyte-endothelial cell combination has the largest KNA value.

Another interesting finding is that, in one-neighbor type, Oligodendrocyte's KNA value starts to surpass the endothelial cells around 480 px, which reflects the oligodendrocyte shows greater significance on the neuronal neighborhood in the far distance.

6.5.2. EXPERIMENT 1: Neuron Status Comparison

To find which type and status of glial cells is mostly associated with the nearby neuron statuses changes, we use the simple definition of neuron neighborhood as the group structure. As shown in **Figure 6-1 (A)**, neuron neighborhood features are partitioned into three glial cell types: oligodendrocyte, microglia and astrocyte, and each glial cell has five states. The five neuron statuses are regarded as the class labels.

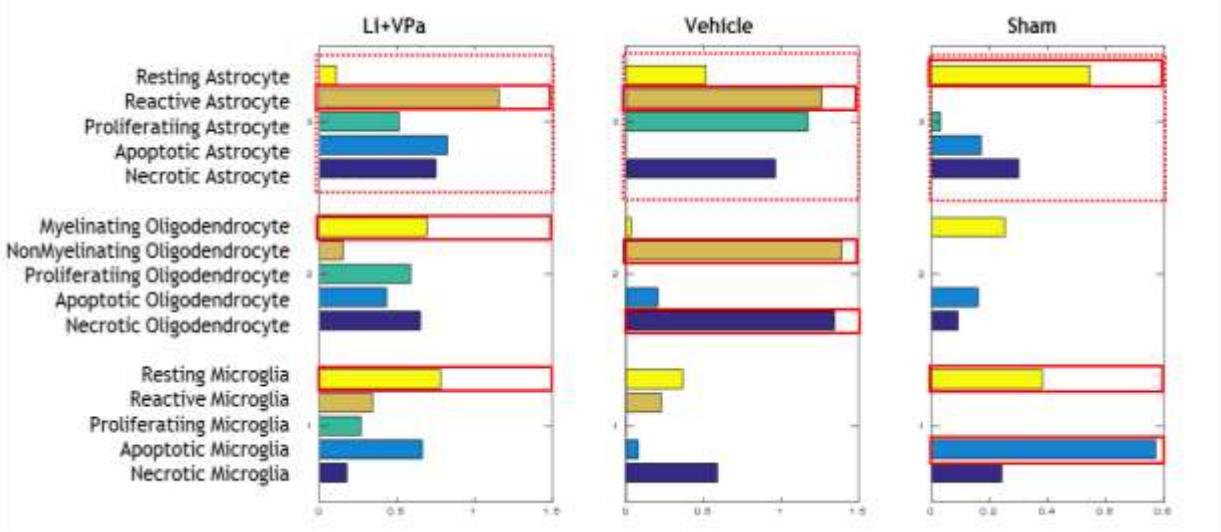


Figure 6-7. The Feature Selection Result of Neuron Status Comparisons by Using Sparse Group LASSO Algorithm.

Sparse group LASSO has been applied on three datasets independently for three tissues Li+VPa (N × D= 4230 ×15), Vehicle ($N \times D$ = 3473×15), and Sham (N × D = 7038 × 15).The parameter α is set to 0.5 in this experiment. Feature selection result in **Figure 6-7** indicates that the neighboring cell status of astrocyte contributes the most to neuron status changes in all tissues. The absolute value of sparse group LASSO coefficients are normalized by feature scaling. The solid frames indicate the top three significant features in feature level selection, dashed frames indicate the top one significant glial cell type in group level selection. The numbers of non-zero features of Li+VPa, and Vehicle and Sham are 15,12 and 10, respectively. In Vehicle dataset, Sparse group LASSO selects 12 out of 15 non-zero coefficients, which means apoptotic astrocyte, proliferating oligodendrocyte

and proliferating microglia barely make changes in neuron neighborhood in injury condition. In Sham dataset, Sparse group LASSO selects 10 out of 15 non-zero coefficients, which means reactive astrocyte, non-myelinating oligodendrocyte, proliferating oligodendrocyte, reactive microglia and proliferating microglia barely make changes in neuron neighborhood in health condition.

Generally, the neighboring cell status of astrocyte contributes the most to neuron status changes of all the individual condition of tissues, which verifies the role of astrocytes on neuron transition modulation. When the brain is in normal condition, two additional types of neuron-glial interactions are present in injured rats than healthy rats, especially neuron-astrocyte and neuron- oligodendrocyte interactions; three additional types of neuron-glial interactions are present in treated rats than injured rats (five additional types in treated rats than healthy rats) .

Table 6-4. Evaluations of the Feature Selection Results.

Measurements		**Li+VPa**	**Vehicle**	**Sham**
Stability	**JSM (index)**	0.22	0.44	**0.57**
	SRCC (rank)	-0.06	0.14	**0.25**
Computational	**Time (s)**	0.12	0.10	**0.03**
Cost	**Number of Iteration**	578	559	**119**

Stability tests had implemented on the feature selection results. Each dataset has partition into $q = 10$ sub-samples, and the top $k = 3$ features have selected. From **Table**

6-4 we can see that Sham tissue dataset is the most stable both in terms of index and rank. In addition, Sham dataset also has the least computational cost.

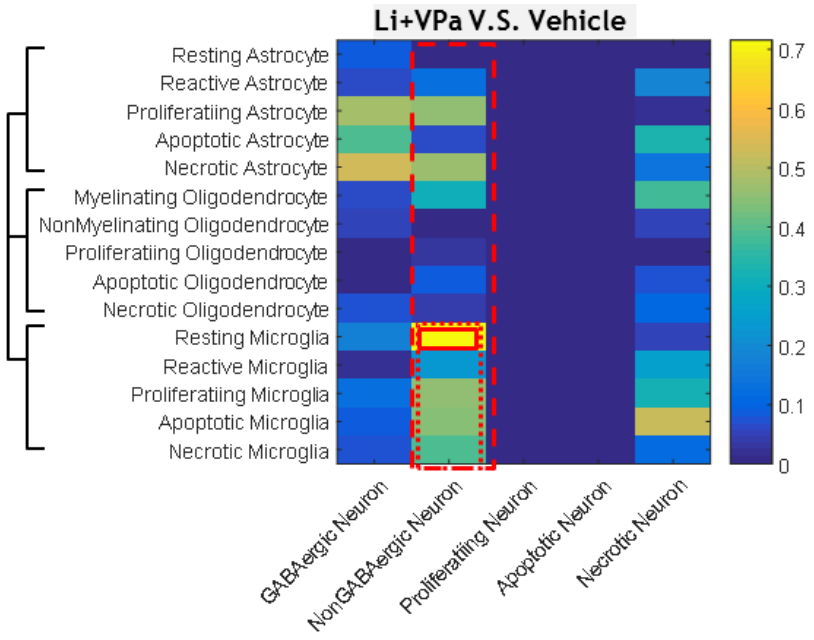
6.5.3. EXPERIMENT 2: Tissue Type Comparison

In this experiment, we take the tree structure of neuron neighborhood definition as **Figure 6-1 (B)** shown. The states of neurons are no longer the class labels as experiment 1 but the parent nodes definitions of the feature structure tree with 75 nodes. The number of nodes in level 1 is 5, number of nodes in level 2 is 15, and the number of nodes in level 3 is 75. To find out which type and status of neuron neighborhood associate the most to the changes of tissue types, Samples from Li+VPa, Sham and Vehicle are merged into one dataset ($N \times D = 14741 \times 75$) and distinguished by their corresponding class labels. Except for the three classification problem, we also apply three pair-wise comparisons of two-class classification problems: Li+VPA v.s. Sham ($N \times D = 11268 \times 75$), Vehicle vs Sham($N \times D = 10511 \times 75$), Li+VPA vs Vehicle($N \times D = 14741 \times 75$).

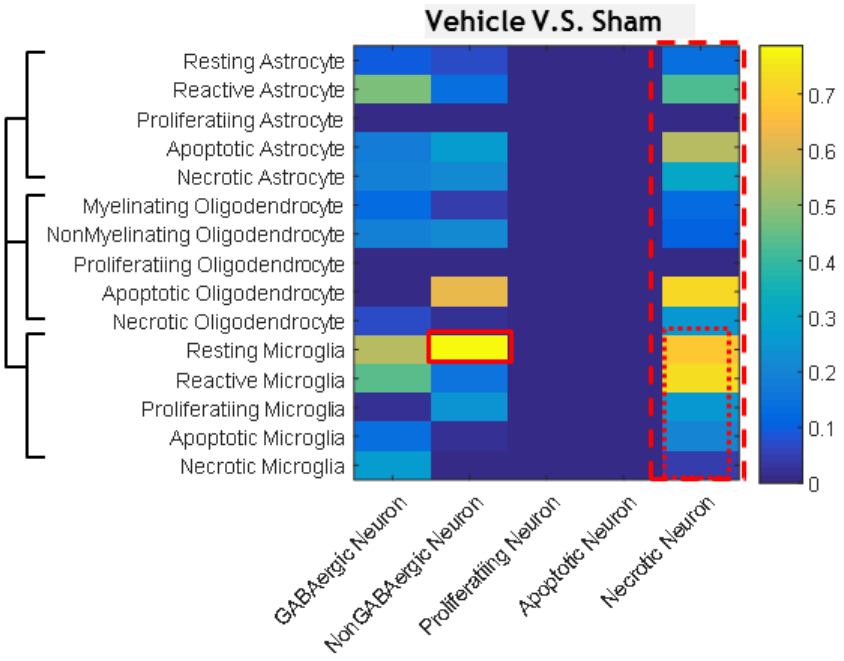
The tree–guides group LASSO has applied on three two-class datasets and a three-class dataset. The color maps of the coefficients of the four problems are shown in **Figure 6-8**. **(A-C)** show the results of pair-wise comparisons and (**D)** shows the result of a whole class comparison. **(A)** and **(B)** indicate the effect of Li+VPa treatment and the effect of injury, respectively. The solid frames indicate the feature selection; dotted frames indicate the level 1 group selection for glial cell types; dashed frames indicate the level 2 group

selection for neuron states. This indicates that both mTBI and treatment cause the highest impact on the neuron-microglia neighborhoods when the datasets are compared pair-wisely; this verifies the function of the microglia, which is an efficient response to injury and recovery. By comparing Vehicle and Sham dataset, mTBI causes the highest impact on the neuron-microglia neighborhoods. By comparing Li+VPA and Vehicle dataset, the treatment of Li+VPA causes the highest impact on the neuron-microglia neighborhoods. The spatial neighborhoods of non-GABA-ergic neurons show the greatest differences with respect to proliferating oligodendrocytes (meriting a further investigation).

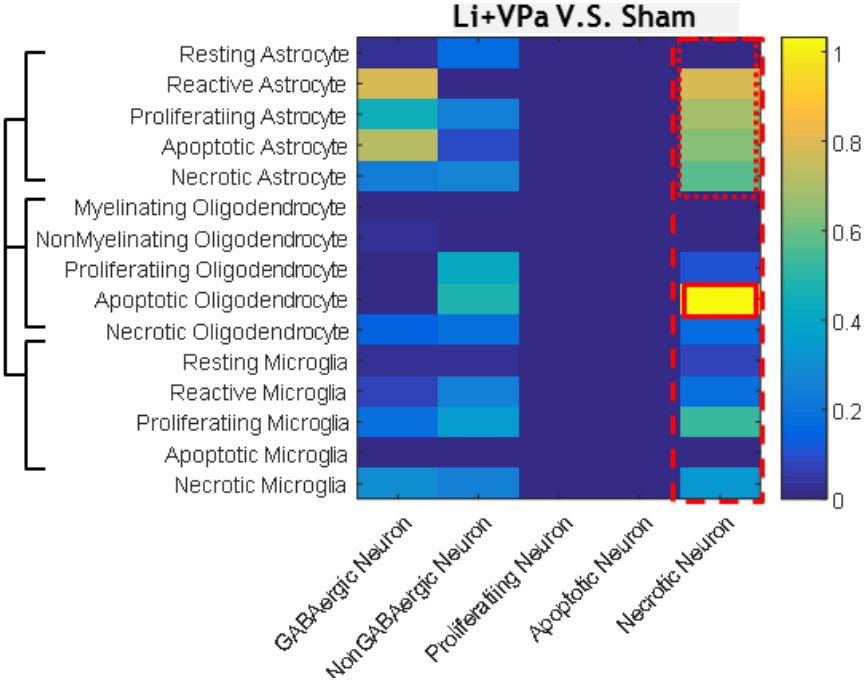
A



B


Vehicle V.S. Sham
Resting Astrocyte
Reactive Astrocyte
Proliferatiing Astrocyte
Apoptotic Astrocyte
Necrotic Astrocyte
Myelinating Oligodendrocyte
NonMyelinating Oligodendrocyte
Proliferatiing Oligodendrocyte
Apoptotic Oligodendrocyte
Necrotic Oligodendrocyte
Resting Microglia
Reactive Microglia
Proliferatiing Microglia
Apoptotic Microglia
Necrotic Microglia
GABAergic Neuron
NonGABAergic Neuron
Proliferatiing Neuron
Apoptotic Neuron
Necrotic Neuron
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

C


Li+VPa V.S. Sham
Resting Astrocyte
Reactive Astrocyte
Proliferatiing Astrocyte
Apoptotic Astrocyte
Necrotic Astrocyte
Myelinating Oligodendrocyte
NonMyelinating Oligodendrocyte
Proliferatiing Oligodendrocyte
Apoptotic Oligodendrocyte
Necrotic Oligodendrocyte
Resting Microglia
Reactive Microglia
Proliferatiing Microglia
Apoptotic Microglia
Necrotic Microglia
GABAergic Neuron
NonGABAergic Neuron
Proliferatiing Neuron
Apoptotic Neuron
Necrotic Neuron
1
0.8
0.6
0.4
0.2
0

D

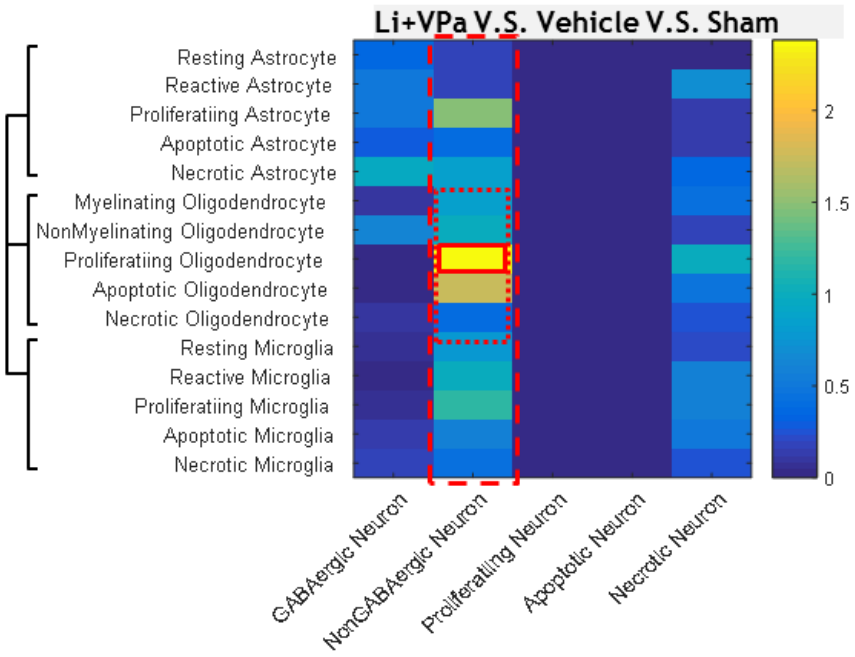


Figure 6-8. Color-Coded Heatmap Display of LASSO Coefficients.

Table 6-5. Stability and Computational Cost of 10 Top Feature Selection Results

		Two Classes			Three Classes
Classifier		**Li+VPa V.S. Sham**	**Li+VPa V.S. Vehicle**	**Vehicle V.S. Sham**	**Li+VPa V.S Vehicle V.S Sham**
Stability	**JSM (index)**	**0.17**	0.11	0.08	0.09
	SRCC (rank)	0.30	**0.31**	0.20	0.25
Computational Cost	**Time (s)**	0.70	**0.46**	0.68	2.11
	Number of Iterations	1550	**1210**	1790	4440

Stability tests had implemented on the feature selection results. Each dataset has partition into $q = 10$ sub-samples, and the top $k = 10$ features have selected. From **Table 6-5** we can see that feature selection of "Li+VPa Vs Sham" problem is the most stable in terms of index, "Li+VPa Vs Vehicle" is the most stable in terms of rank. Feature selection of "Li+VPa Vs Vehicle" has the least computation cost.

6.5.1. EXPERIMENT 3: Brain Side Comparison

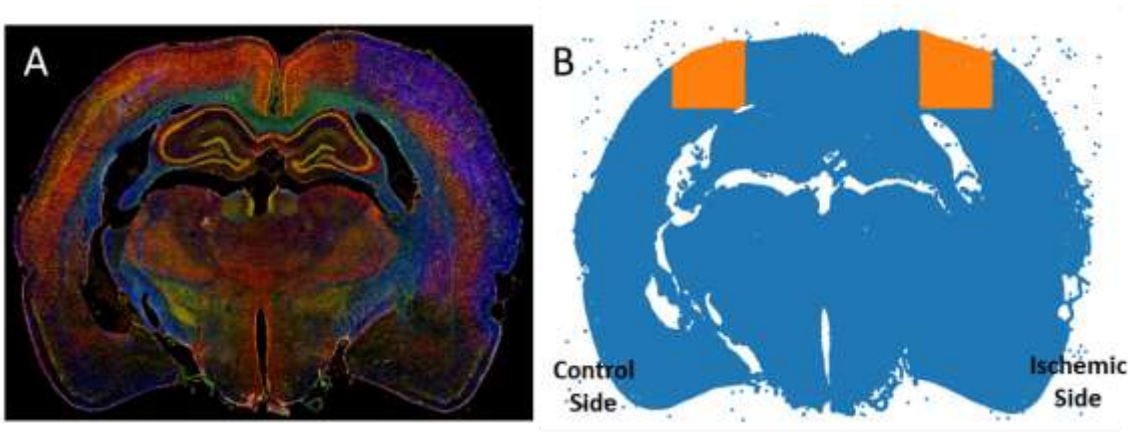


Figure 6-9. (A) Multiple IHC Image (B) Nuclei Seed Data and Analysis of ROIs.

Other than the TBI dataset, we have also analyzed a stroke rat brain at 72 hours post-Middle Cerebral Artery Occlusion (MCAO) injury, as shown in **Figure 6-9**. We use the Sparse Group-LASSO to compare the neuronal neighborhood in control (left) and ischemic(right) sides of the brain, as shown in the orange region in **Figure 6-9 (B)**.

To visualize how LASSO coefficients perform on the alteration of neighborhood relationships. We plot the graphical representation of the neuron-glia types that has the largest coefficient (0.45), as shown in **Figure 6-10 (A)** , and the smallest coefficient (0.06) as shown in **Figure 6-10 (B).** It is obvious that the graphical plots in the former express a large alteration between the left and right sides, and the latter expresses a little alteration.

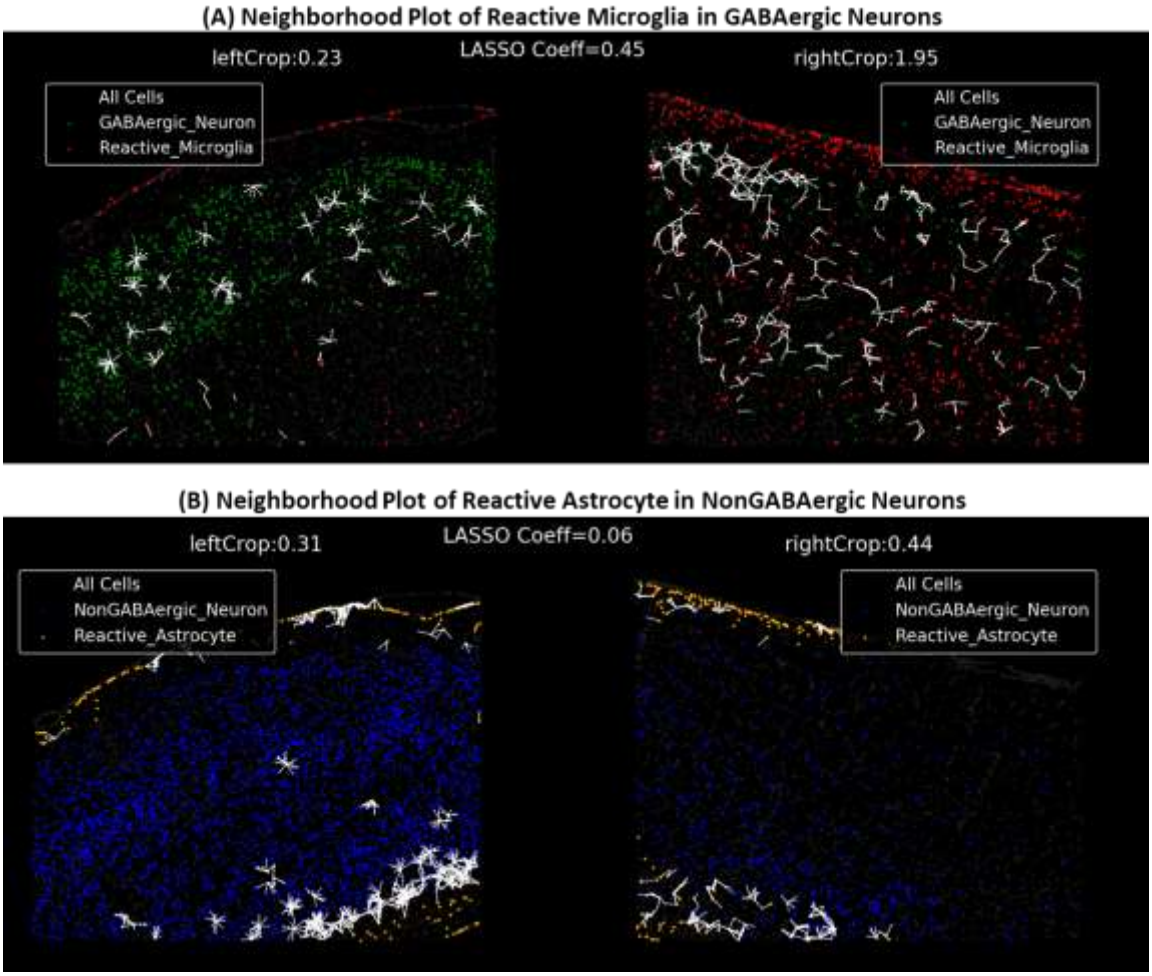


Figure 6-10. Sparse Group LASSO Coefficients in Stroke Dataset.

6.6 Conclusions

We developed a robust method to detect and profile injury caused alterations to brain tissue at the multi-cellular scale. To analyze the interaction of multiple cell types, we used a single measurement to measure the local neuron neighborhoods by combining the density and distance of neuron-glial influence; these measurements previously could only be used independently. Our model represented biological cyto-architecture better than existing methods in terms of the proximity level from neighbors to centers, fitting the uneven distribution of cells, and performing robust to distance parameters

Then, sparse group LASSO and tree LASSO are applied to extract the features that are pre-defined by their intrinsic hierarchical structures. Both the two features selection algorithm can extract individual features and feature groups simultaneously. We have also compared the stability and computational cost of the feature selection overall tissues. The experiments show that our method produced stable and efficient feature selection results.

By taking the feature construction, feature selection, and feature stability test, our method generates the following biological findings: generally, the neighboring cell status of astrocyte contributes the most to neuron status changes in all condition of tissues. When the brain is in normal condition, only 10 out of 15 types of neuron-glial interactions cause neuron status changes; when the brain getting injured, two more types of neuron-glial interactions (especially neuron-astrocyte and neuron-oligodendrocyte interactions) are

excited compared to normal condition; when the injured brain received treatment, all 15 types of neuron-glial interactions are excited. Within all three condition of tissue dataset, Sham (healthy) is the most stable in terms of both index and rank, also has the least computational cost.

Both mTBI and treatment cause the highest impact on the neuron-microglia neighborhoods when the datasets are compared pair-wisely; this verifies the function of the microglia, which is an efficient response to injury and recovery. The spatial neighborhoods of non-GABAergic neurons show the greatest differences with respect to proliferating oligodendrocytes (meriting a further investigation). The feature selection results "Li+VPa Vs Sham" problem is the most stable in terms of index, while "Li+VPa Vs Vehicle" is the most stable in terms of rank.

Our method could be used as an effective tool to profile the alterations of cellular microenvironment in single and multiple pathological condition. In addition, the neighborhood attraction could be used to measure the interaction and association between any two types of spatial instances not limited to cellular microenvironment with neuron and glial cells.

6.7 Future Work

Since we are currently using 2-dimensional image to process, the model of neuron neighborhood is defined in 2D spaces. In the future, we could implement the analysis of

neuron neighborhood changes to three-dimensional space so as to approximate the real neuron neighborhood. We could also extend the application of our algorithm to the whole brain when the phenotyping results of the cell in the whole brain are available. Other work like exploring other hierarchical feature selection methods and comparing the performance against tree-LASSO could also be considered.

CHAPTER 7 LARGE SCALE MULTI-PLEX IMAGE REGISTRATION

Registration is an important step for image alignment of multiple rounds. When the number of channels goes beyond the limitation of single round staining, even the most experienced staining technician cannot guarantee the tissues are accommodated to the same location in the microscope before and after a few times of staining. Thus, the changes in different rounds, such as tissue distortion and misalignment, are inevitable problems in highly multiplex image analysis. The registration algorithm could detect those changes.

To solve the registration problem, the following assumptions are made: In multiplex multi-round imaging, images within rounds are completely aligned while could be stained in different biomarkers, whereas images between rounds are from the same tissue and share one channel with the same staining biomarker (e.g., DAPI to stain all nuclei). The shared channels are the targets to be registered with subtle misalignment (within 50 pixels) performed in affine transform.

Using a large set of biomarkers helps us better understand the underlying pathology and better insight into the biological process. But due to the limitation of the number of fluorescent markers in each round, we must reuse the common dyes in several rounds by stripping and re-staining the tissue. The stripping and re-staining cause misalignment between rounds. We use a robust affine deformable registration algorithm based on DAPI (nuclear) channel images to align the montages from one staining round to the next. First,

we extract key-points and corresponding descriptors using Oriented FAST and Rotated BRIEF (ORB) method [48]. We find the matching keypoints based on the similarity of descriptors using the Brute-force matching method with the Hamming distance metric. The outliers of matched keypoints are removed using RANdom SAmple Consensus (RANSAC) method [49]. Finally, the most robust matched keypoints are used to estimate the transformation parameters.

7.1 Previous Methods

The available registration algorithms are mainly two categories: Intensity-based and feature-based methods. The feature-based method overcame the problems in higher repeatability and robustness to different illumination situations [50]. The three main steps of feature-based image registration methods are: keypoint extraction, keypoint matching and transform estimation.

For transform estimation, RANSAC is a normal and effective way that people generally use to select the robust affine transform vectors [51]. While for keypoint extraction, researchers have made great effort to seek for reliable solutions with higher accuracy. For example, Scale-Invariant Feature Transforms (SIFT) [52] and Speeded Up Robust Features (SURF) [53] are predominated algorithms for years. However, the former demands high computational costs, and the latter shows sensitivity to image noise. In our research, we adopt the current advanced descriptor named Oriented FAST and Rotated BRIEF (ORB)

[48] as a combination of Features from Accelerated Segment Test (FAST) [54] key point descriptor and the BRIEF [55] which claimed to outperform the mentioned algorithms with higher speed and higher accuracy.

It is also worth mentioning that there are powerful CNN-based image registration methods such as [56, 57], which can efficiently register the deformable image. However, most of them are not fit for large images, and the simple run on crop images can cause edge effect issues.

7.2 Basic ORB Based Image Registration Pipeline

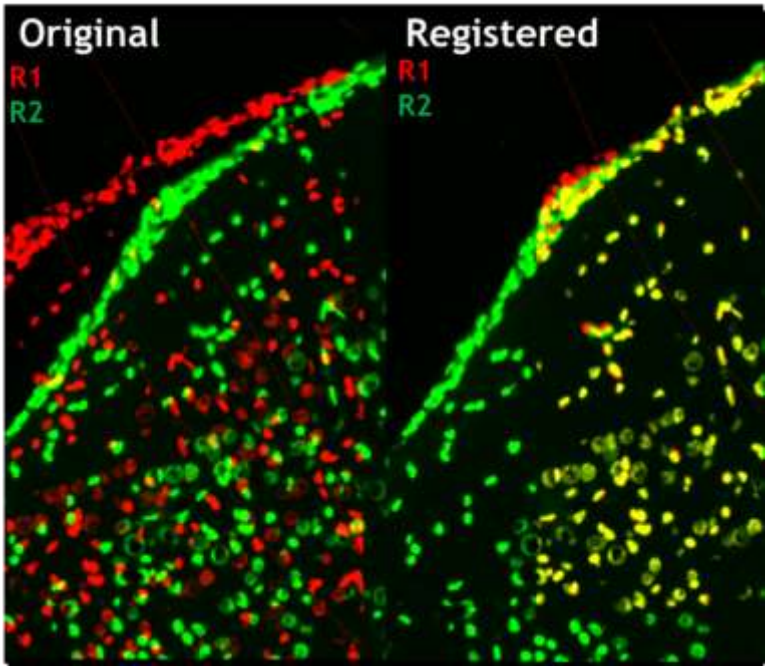


Figure 7-1. Register the Source Image to Target Image at Small-scale.

Scikit-image has provided a panorama pipeline for image registration and stitching [58]. We first briefly illustrate this pipeline as our baseline model. It is illustrated as a prototype of small-scale brain image registration from a source image I_s to a target image I_t. As shown in **Figure 7-1** , the target image from round 2 is shown in green, source image from round 1 is stained in red. The left image shows the original miss-aligned images; the right image shows the expected registered results. ORB algorithm is used as an advanced binary feature descriptor extractor for both images. The basic procedures of image registration are:

1) Extract and match the keypoints in the target (reference image) and source (to be registered to the target image) channels;
2) Select the robust matched pairs and their transform vector;
3) Wrap two images according to their offsets to the common portion and their transform.

7.2.1. Feature Detection and Extraction

The goal of feature detection and matching is to estimate a transformation from I_s to a target image I_t with the target number of features at most *N*. Each ORB feature consists of a pair of keypoint and descriptor: keypoints tell the spatial location of the feature, and descriptors tell the vector representation of the feature.

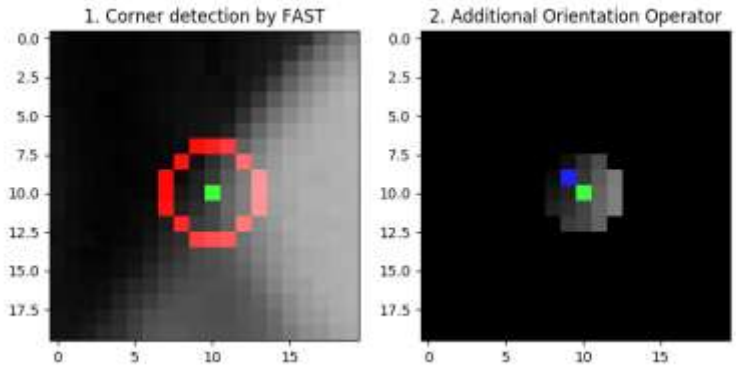


Figure 7-2. ORB Feature Keypoint Detection of an Example Patch.

The keypoint detection of ORB uses FAST method with an additional orientation operator component. According to the original FAST algorithm, a point is claimed as a corner keypoint O (the green points in **Figure 7-2**) only when there are more than F_n continuous pixels over all the checking pixels are "valid" pixels. 16 checking pixels are the ones in the circle centered in the interest point with the radius to 3, as shown in the pixels inside the red circle in **Figure 7-2 (1)**. A pixel is defined as "valid" when its pixel intensity is higher than the keypoint O intensity plus F_τ. In our experiment, we set the $F_n = 9$, $F_\tau = 0.08$. Then, the orientation of the corner keypoint is calculated as a supplement for the original FAST. The intensity centroid C is the gravity/ mass center of the intensity patch, including all the pixels inside the circle. The orientation can be represented as the arctan of the vector $\overrightarrow{OC}$. This angle θ will be used for the upcoming feature extraction to achieve rotation invariance. An example can be seen in **Figure 7-2**: **(1)** shows FAST Method to

detect corners, the point of interest O is colored in green, and the checking points are in red; **(2)** shows the Orientation Operator Calculation, and the intensity centroid C is in blue.

The descriptor generation of ORB uses the BRIEF method with an additional rotation awareness. First, for each keypoint O, using ORB to extract an image patch with the size of 31×31 centered at O and rotating the patch by θ to a canonical direction. Then sooth the image by calculating the integral image of the patch. A binary test τ runs as following: randomly select two pixels in the patch, assign the result to 1 if the former is smaller than the latter, otherwise to 0. For the ORB_n digits ($ORB_n = 256$ in our experiment) vector of a binary descriptor, the values are obtained from running the binary test τ for ORB_n times. Last, to achieve rotation invariance, a rotated BRIEF/ steer BRIEF is used by only keeping the consistent digit values over the discretized patch orientation θ.

In general, for each feature of interest over the total number of n features, there is a 2-dimensional coordinate representing the location of the feature, and a 256- dimensional binary descriptor representing the vector representation of the feature correspondingly.

7.2.2. Feature Matching

First, a brute-force way of feature matching is implemented for each descriptor. Since the descriptor vector is binary, Hamming distance [59] is used to measure the pair of descriptor's differences. For n_S descriptors in source image I_s and n_t descriptors in target

image I_t , a metric $n_S \times n_t$ is calculated ; each descriptor in I_s is one-to-one paired by its closest match to I_t . If the closest distance of a pair of descriptors is larger than a user-defined threshold, this pair will be deprecated. Thus, it will result in m matched descriptors, where $m \leq \min(n_s, n_t)$. Finally, m keypoints with distinct spatial coordinates from I_s and I_t will be selected correspondingly.

7.2.3. Transformation Estimation

According to the setting of multi-round images, misaligned images between rounds could be represented in affine transformations, combining translation, scaling, similarity, reflection, rotation, etc. A 2D Affine transformation can be represented as an augmented matrix composited of a linear map $\boldsymbol{A}$ and a projection matrix $\boldsymbol{b}$, denoted as

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}. \tag{32}$$

We know that for linear transformation, extracting two points can result in a linear function. If more than two points are given, the additional points falling in the linear estimate are inliers, and those falling outside the linear estimate are outliers. In other words, the distribution of inliers data can be explained by some model parameters, while outliers data cannot be explained by the parameters of the model. Thus, the optimal regression line could be obtained by maximizing the number of inliers when changing the selection of two points. That is the core idea for RANSAC transformation estimation. The only difference

is that the data points are expressed in 2D space, so we need to extract 3 points representing a projection matrix. RANSAC for affine transform estimation works as follows: randomly selecting m matched keypoints, estimate an affine transform and fit it to the rest of keypoints, calculate the inlier and outliers, and redo the whole process again. The optimal transformation is the one that has the maximum number of inliers.

In the end, the final registered image for source image I_s is calculated by the inverse transform of the affined transformation model to I_t.

7.3 Improvements on Large Scale Texture Featured Image Registration

The challenges of large-scale multi-round image registration lie in handling the keypoint extraction of large textured images. On the one hand, the scale of the images restricts the keypoint extraction feasibility. E.g., in our case, one channel of an image with a size of 40,000 times 30,000 pixels has over 10,000 key points detected in total. Simply implementing the pipeline often encounters the problem of exceeding the storage capacity of large-scale image and low efficiency. On the other hand, the special texture property of brain image makes it impractical to rely on the detected key points from the whole image simply: those are densely distributed in characteristic regions such as hippocampus and hypothalamuses while sparsely distributed are located in a large proportion of cortex regions. Instead, evenly distributed keypoints methods are preferable because all regions in the whole brains need to be aligned.

To solve the problems mentioned, we specifically applied extraction by image tiles, which generates even and fully covered keypoints over all the image regions and provides the feasibility of multiprocessing acceleration. Images are cropped into L number of small tiles and keypoint extraction is implemented for each tile in parallel with the same setting. Moreover, to avoid the redundant usage of pairwise registration between rounds, we reuse the descriptors from the target round and register all source rounds to the universal target round individually. We will elaborate the steps for our proposed improvements.

7.3.1. Generate Tile Ranges

To achieve accurate registration on different local regions, we design a strategy to split the whole image into different small crops, as illustrated in **Algorithm 5**. Considering the misaligned pixel shifting, an overlap width for each tile is necessary, as long as there are sufficient match keypoints in each pair of target and source tile images. As shown in **Figure 7-3, (A)** Target and Source images with the size of 1000×1500 px is split into 20 tile images **(B)** with the size of 350×350 px and overlap width 50 px. The cropped tiles are shown in pink dashed lines, an example of a tile image is shown in pink solid line.

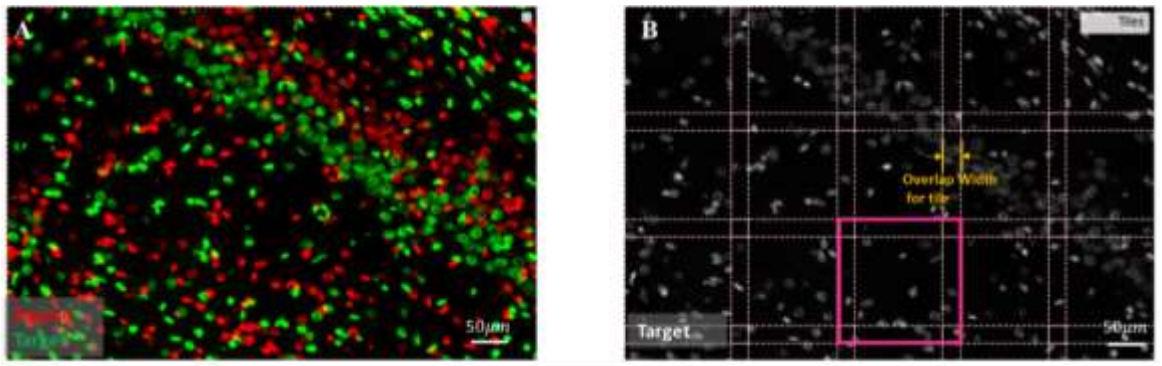


Figure 7-3. Tile Range Generation.

Algorithm 5. Generate Tile Ranges

Parameters: Tile shape($tile_w, tile_h$), overlap width (o)
Outputs: Tile Ranges $\vec{L}$ = [$l^1 \ldots l^L$]

Objective image width W = *min (* $width(I_s)$, $width(I_t)$ *)*
Objective image height H = *min (* $height\ (I_s)$, $hight\ (I_t)$ *)*
$i = 1$
for w *in range(*$1 : W : tile_h$*- o):*
    for h *in range(*$1 : W : tile_h$*- o):*
        l^i = [$w, h, w + tile_w, h + tile_h$]
        $i = i + 1$

7.3.2. Tile-based Feature Detection and Extraction

We enforce a uniform distribution for the keypoint detection over all *L* image tiles, as claimed in **Algorithm 6**. Compared to the original method in **7.2.1**, although the best *N* number of keypoints is set the same, the summation of the number of keypoints detected in each image tiles is not necessarily equal to the number of keypoint detected in the whole image. Instead, the detected tiled-based summation is generally large than the whole image one.

Algorithm 6. Feature Extraction by Tiles

Parameters: total number of best keypoints *N*

Outputs: n_t keypoints $[\overrightarrow{P_t^1}, \dots \overrightarrow{P_t^L}]$ and descriptors $[\overrightarrow{D_t^1}, \dots \overrightarrow{D_t^L}]$ from target image

n_{ts} keypoints $[\overrightarrow{P_s^1}, \dots \overrightarrow{P_s^L}]$and descriptors $[\overrightarrow{D_s^1}, \dots \overrightarrow{D_s^L}]$ from source image

$n_s, n_t = 0$

for l *in* $\vec{L}$*:*

get the tile image i_t^l, i_s^l *by extracting the crop image by the tile range* l

According to 7.2.1, set number of best keypoint to N/L*,*

Detect n_t^l *of local feature keypoint* $\overrightarrow{P_t^l}$ *and descriptors* $\overrightarrow{D_t^l}$ *from* i_t^l

Convert to the global feature keypoint $\overrightarrow{P_t^l} = \overrightarrow{P_t^l} + l$ [0,1]

$n_t = n_t + n_t^l$

Detect N_s^l *of local feature keypoint* $\overrightarrow{P_s^l}$ *and descriptors* $\overrightarrow{D_s^l}$ *from* i_s^l

Convert to the global feature keypoint $\overrightarrow{P_s^l} = \overrightarrow{P_s^l} + l$ [0,1]

$n_s = n_s + n_s^l$

The illustration of feature extraction can be seen in the **Figure 7-4**. Cyan dots in **(A)** show the keypoint extraction results from the original method, Cyan and yellow dots in **(B)** show the ones from tile-based improvement. The overlapped tiles are shown in pink grids. The additional keypoints extracted from the tile-based improvements compared to the original ones are shown in yellow dots. Tiled-based keypoint extraction enforced a uniformed sampling compared to the original method.

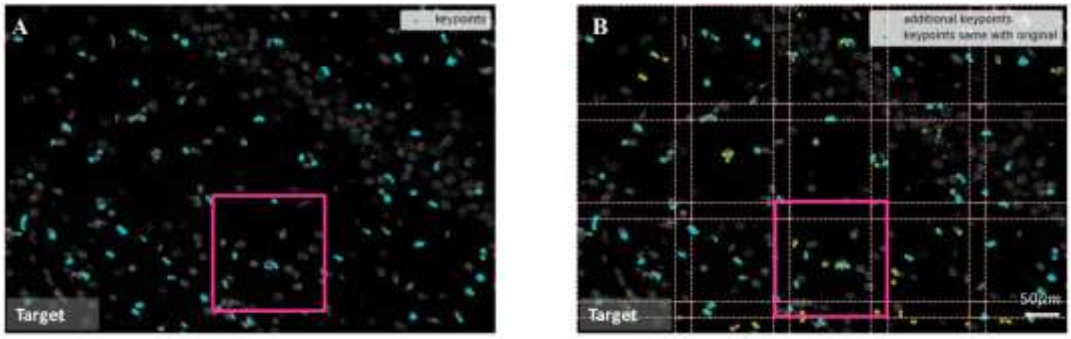


Figure 7-4. The Comparison of Keypoint Extraction of the Original and Tiled-based ORB Method.

7.3.3. Tile-based Feature Matching

Feature matching, likewise, can be implemented by each tile of image. However, depending on the different tile size and overlap width settings, we cannot guarantee that most of the matched keypoint of the source image tile lies in the same target image tile. Thus, we design a “checking window” dilated to image tile to increase the matched pairs' searching range. The dilation pixel is set to checking dilation R according to **Algorithm 7,** and each tile image of the source image will receive m^l matching pairs. It is worth to

mention that if not implement it by tile, the calculation of the Hamming distance matrix of $n_S \times n_t$ from all the target and source image keypoints are often excess the computation cache memory, resulting in that the original ORB pipeline is infeasible on large images (e.g. a general setting of n_S *or* n_t can be close to 100,000). Instead, matching by tiles only need the space for $n_S^l \times n_t^l$.

Algorithm 7. Feature Matching by Tiles

Parameters: n_t descriptors from target, n_t descriptors from source, checking dilation *R*
Output: matched descriptors $\vec{M}$

$\vec{M} = \emptyset$
for l *in* $\vec{L}$*:*
get the descriptors $\overrightarrow{D_s^l}$ *from source image located in l*
checking range c = the rectangular defined by l with a dilation by R
get the descriptors $\overrightarrow{D_t^c}$ *from target image lies in c*
find matched keypoints $\vec{M}^l$ *by brute force method According to 7.2.12.*
$\vec{M} = \vec{M} \cup \vec{M}^l$

As illustrated in **Figure 7-5**, **(A)** shows the results from the original method, **(B)** shows the tiled-based improvement, the matchings in the same tiles are using the same color. The overlapped tiles are shown in grids.

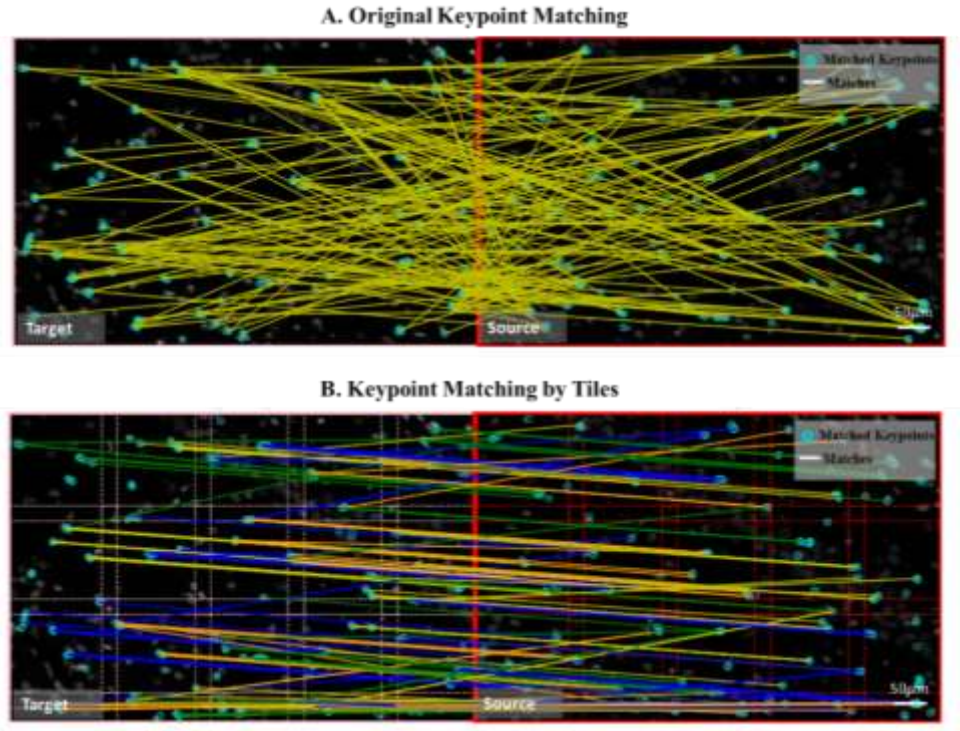


Figure 7-5. The Comparison of Keypoint Matching by the Original and Tiled-based Method.

7.3.4. Tile Sampled Hierarchical RANSAC

To estimate the misaligned image's transformation, a difficult edge effect will occur if we simply run by tiles as we did for feature selection and matching. We cannot guarantee the transformation in different tiles is identical to each other. Thus, it still needs to run the estimation it globally.

The basic procedure of RANSAC, as discussed in **7.2.3**, randomly selects matched keypoints over the whole image. However, we need to avoid the local alignment when a

few matched keypoints in the same tile are selected together at this step. Thus, we propose a tilled-based hierarchical RANSAC that first randomly select min_$samples$ of tiles, then randomly select one keypoints in each selected tile, as claimed in **Algorithm 8**. In this way, we can guarantee that the random keypoint to estimate RANSAC's transformation model never belongs to the same tile.

Algorithm 8. RANSAC by Tiles

Parameters: number of matched keypoints $|\vec{M}|_0$, total numbers of iterations K, residual threshold θ, minimum number of samples S ($s \leq |\vec{L}|_0$)

Output: *optimal affine transformation* F'

minimum inlier rate I_{min} = 100%
for i in 1…K:
  sampled keypoints $\vec{M}_s = \emptyset$
  while $|\vec{M}_s|_0 < S$:
    randomly select one tile $l_s \in \vec{L}$
    get the keypoints $\vec{M}_l$ *located inside* l_s
    if $|\vec{M}_l|_0 > 0$:
      randomly select one keypoint $m \in \vec{M}_l$
      $\vec{M}_s = \vec{M}_s \cup \{m\}$
  estimate the affined transform from keypoints $\vec{M}_s$, *denoted as* F_i
  for l in $\vec{L}$:
    get the keypoints $\vec{M}_l$ *located inside l*
    number of inliers $n^l = 0$
    if $|\vec{M}_l|_0 > 0$:
      for m in $\vec{M}_l$:
        calculate the model residual $r = F_i(m)$
        if $|r| < \theta$:
          $n^l = n^l + 1$

Algorithm 8. **(Continued)**

Current inlier rate $I_i = \left(\sum_{j=1}^{L} \frac{n^j}{|\vec{M}_j|_0}\right) / \left|\vec{L}\right|_0$

if $I_i < I_{min}$:

Update $F' = F_i$

The illustration of Hierarchical RANSAC can be seen in the **Figure 7-6**. The yellow stars in **(A)** show the random sampled three matching keypoints; the red stars in **(B)** show the ones from hierarchical random sampling. The overlapped tiles are shown in pink grids. It can also be seen that compared with the original method, the key point extraction based on tile realizes unified sampling.

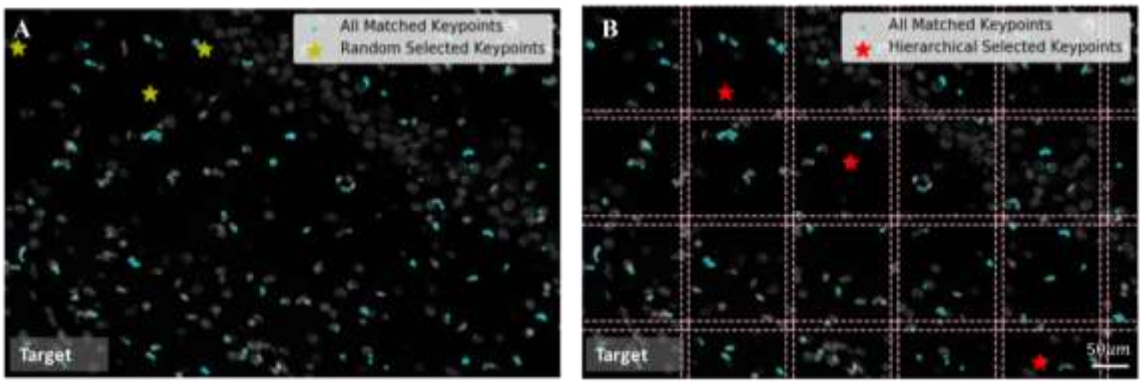


Figure 7-6. The Comparison of the Original RANSAC and Hierarchical RANSAC.

7.3.5. Bootstrapped Regional Refinement

Artificial errors such as tissue folding occasionally happen to the acquired images. From an example in **Figure 7-7 (A)**, the top-right corner of the image expresses a local region of tissue folding, which cannot be corrected by the aforementioned global image

registration strategy. To recover those local misaligned regions, we proposed a bootstrapped regional refinement method.

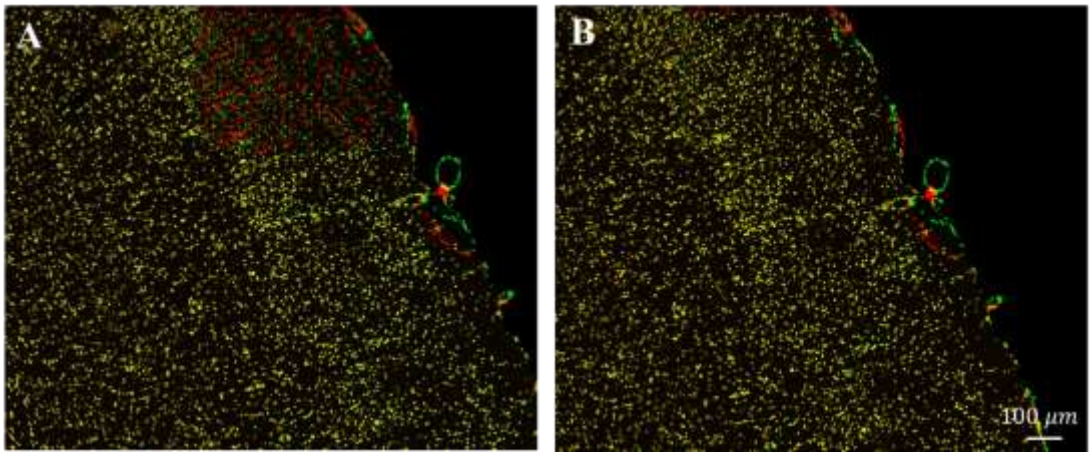


Figure 7-7. An Example of Fixing a Folded Area by Regional Refinement. (A) Registered source image superimposed on target image. (B) The result after regional refinement.

Given that the source image and the target image have similar intensity distribution, the simplest way to compare the registered source image with the target image is to compare their binarized masks. We use the Otsu thresholding method [60, 61] to binarize the registered and the target image and then calculate the proportion of the two binary masks' absolute mask to the target image's binary mask. Here we defined an error rate as

$$ErrorRate = \frac{\sum_{m=1}^{M} \| |I_s(m)>\tau_s| - |I_t(m)>\tau_t| \|_0}{\sum_{m=1}^{M} \| I_t(m)>\tau \|_0},\ I_s(m)\epsilon X,\ I_t(m)\epsilon V. \quad (33)$$

If the proportion is larger than a threshold (e.g.70%), we claim this region is a highly distorted region that needs regional refinement. An example of an acceptable registered image with $ErrorRate = 65.1\%$ is shown in **Figure 7-8**. The left is the registered source image (green) superimposed on the target image (red); the right is the binarized difference map of the registration result. In the end, registration from **Algorithm 5** to **Algorithm 8** is implemented in all the local misaligned regions.

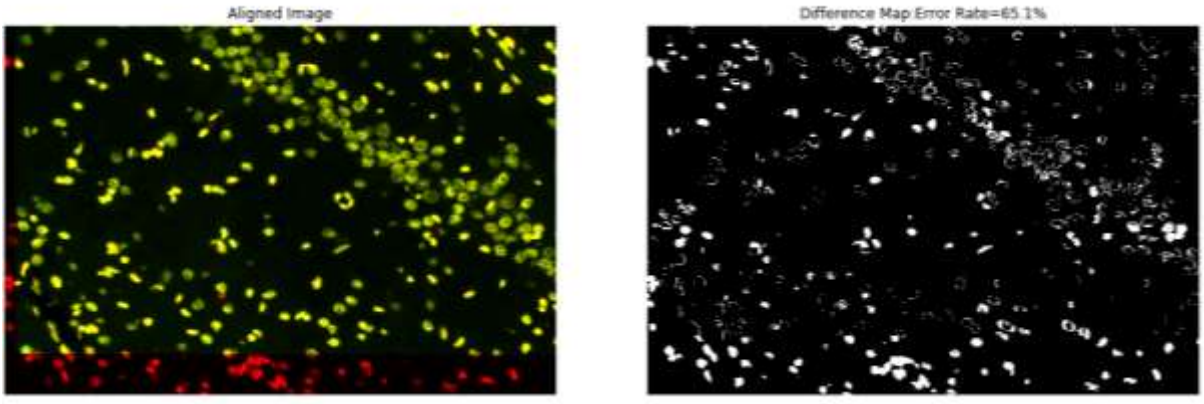


Figure 7-8. An Example Difference Map for Registered Image.

7.4 Experiment Discussion

7.4.1. Multiprocessing Acceleration

Since the procedure of feature detection and extraction (**7.3.2**) and feature matching **(7.3.3**) are implemented independently for each tile, we can apply multi-thread parallel programming to accelerate the pipeline speed. Given that the matrix calculation of keypoint

matching requires adequate memory, we only implement parallel programming on the CPU. The general idea is to evenly distribute all the tiles into each CPU core such that each core runs a subset of tiles simultaneously. The exact run time improvement for multiprocessing depends on the number of tiles in each thread, and the number of tiles depends on the size of the tile specified by the user. From the summary shown in **Figure 7-9**, we can see that multiprocessing is on average 4 times faster than a single thread.

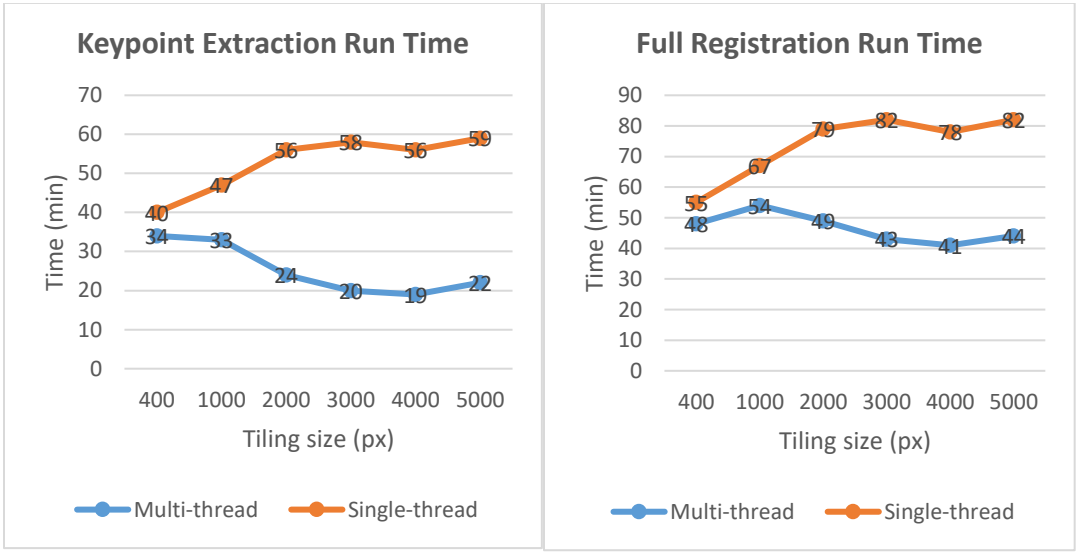


Figure 7-9. Run Time Summary of Registration. Example of registering a pair of images with the size of 40,000 px by 30,000 px.

7.4.1. Other Parameter Settings

Although **Figure 7-9** shows a faster run time for small image tiles, the registration accuracy is not necessarily negatively related to tile shape. Thus, to take a tradeoff of speed

and accuracy performances, we provide the experimental based parameter suggestions as shown in **Table 7-1**.

Table 7-1. Multiplex Image Registration Parameters

Section	Parameters
Tile Generation	target_shape = (4000,8000) tile_shape = (1000, 1000) check window dilation = 100 crop_overlap = 10
Tile-based Feature Detection and Extraction	n_keypoints = 300000 fast_threshold = 0.08 harris_k = 0.1
Tile-based Hierarchical RANSAC	min_samples = 5 residual_threshold = 5 max_trials = 600 random_state = 42
Regional Refinement	err_thres = 1.5 bootstrap = True imadjust = True
Others	multiprocess = False

7.5 Conclusions

Our framework of the tile-based registration pipeline shows significant improvement in large-scale image alignment. Our main contributions are as follows:

We propose a large-scale texture image registration algorithm based on tile, which can uniformly sample the key point detection in all regions of the image and is more feasible for texture-based images. Keypoint extraction and keypoint matching are implemented

independently and locally for each tile, and the transformation estimation by RANSAC has also been modified to tile properties resoundingly.

We not only break the limitation on image size, whereas the original ORB based image registration algorithm failed to run on large-scale images, but also applied a parallel programming acceleration for the descriptors and key points extraction and matching (speed up by 4-5 times).

Lastly, the algorithm can automatically detect and correct the misaligned sub-regions from artificial error to fully maintain raw image information.

REFERENCES

[1] T. Lindeberg, "Scale selection properties of generalized scale-space interest point detectors," *Journal of Mathematical Imaging and Vision,* vol. 46, no. 2, pp. 177-210, 2013.

[2] Y. Al-Kofahi, W. Lassoued, W. Lee and B. Roysam, "Improved Automatic Detection and Segmentation of Cell Nuclei in Histopathology Images," *IEEE Transactions on Biomedical Engineering,* vol. 57, no. 4, pp. 841-852, 2010.

[3] J. Cousty, G. Bertrand, L. Najman and M. Couprie, "Watershed Cuts: Thinnings, Shortest Path Forests, and Topological Watersheds," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 32, no. 5, pp. 925-939, 2010.

[4] P. Neubert and P. Protzel, "Compact Watershed and Preemptive SLIC: On Improving Trade-offs of Superpixel Segmentation Algorithms," in *ICPR '14 Proceedings of the 2014 22nd International Conference on Pattern Recognition*, 2014.

[5] N. Bougen-Zhukov, S. Y. Loh, H. K. Lee and L.-H. Loo, "Large-scale image-based screening and profiling of cellular phenotypes.," *Cytometry Part A,* vol. 91, no. 2, pp. 115-125, 2017.

[6] F. Xing, Y. Xie and L. Yang, "An automatic learning-based framework for robust nucleus segmentation.," *IEEE transactions on medical imaging,* vol. 35(2):, pp. p. 550-566., 2015.

[7] O. Ronneberger, P. Fischer and T. Brox., "U-net: convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, Springer, 2015.

[8] K. He, G. Gkioxari, P. Dollar and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference On Computer Vision*, 2017.

[9] J. C. Caicedo, A. Goodman, K. W. Karhohs, B. A. Cimini, J. Ackerman, M. Haghighi, C. Heng, T. Becker, M. Doan, C. McQuin, M. Rohban, S. Singh and A. E. Carpenter, "Nucleus segmentation across imaging experiments: the 2018 Data Science Bowl.," *Nature Methods,* vol. 16, no. 12, pp. 1247-1253, 2019.

[10] P. Yuan, A. Rezvan, X. Li, N. Varadarajan and H. V. Nguyen, "Phasetime: Deep Learning Approach to Detect Nuclei in Time Lapse Phase Images.," *Journal of Clinical Medicine*, vol. 8, no. 8, p. 1159, 2019.

[11] G. Papandreou, L.-C. Chen, K. P. Murphy and A. L. Yuille, "Weakly-and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[12] A. Khoreva, R. Benenson, J. Hosang, M. Hein and B. Schiele, "Simple Does It: Weakly Supervised Instance and Semantic Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] Z. Shi, Y. Yang, T. M. Hospedales and T. Xiang, "Weakly-supervised image annotation and segmentation with objects and attributes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. p. 2525-2538., 2016.

[14] M. Li, M. Soltanolkotabi and S. Oymak, "Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2020.

[15] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," arXiv preprint arXiv:1412.6596, 2014.

[16] X. Zhao, S. Liang and Y. Wei, "Pseudo Mask Augmented Object Detection," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[17] B. Biggio, B. Nelson and P. Laskov, "Support Vector Machines Under Adversarial Label Noise," in *Asian Conference on Machine Learning*, 2011.

[18] B. Frenay and M. Verleysen, "Classification in the Presence of Label Noise: A Survey," *IEEE Transactions on Neural Networks*, vol. 25, no. 5, pp. 845-869, 2014.

[19] D. Rolnick, A. Veit, S. Belongie and N. Shavit, "Deep Learning is Robust to Massive Label Noise," *arXiv preprint arXiv:1705.10694*, 2018.

[20] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev and R. Fergus, "Training Convolutional Networks with Noisy Labels," in *3rd International Conference on Learning Representations, ICLR 2015*, 2014.

[21] C. Pelletier, S. Valero, J. Inglada, N. Champion, C. M. Sicre and G. Dedieu, "Effect of Training Class Label Noise on Classification Performances for Land

Cover Mapping with Satellite Image Time Series," *Remote Sensing,* vol. 9, no. 2, p. 173, 2017.

[22] A. Drory, S. Avidan and R. Giryes, "On the Resistance of Neural Nets to Label Noise.," *arXiv preprint arXiv:1803.11410,* 2018.

[23] W. Abdulla, "Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow," Github, 2017. [Online]. Available: https://github.com/matterport/Mask_RCNN.

[24] R. Rubinstein, M. Zibulevsky and M. Elad, "Efficient implementation of the K-SVD algorithm and the Batch-OMP method.," Department of Computer Science, Israel, 2008.

[25] J. Nunez-Iglesias, R. Kennedy, S. M. Plaza, A. Chakraborty and W. T. Katz, "Graph-based active learning of agglomeration (GALA): a Python library to segment 2D and 3D neuroimages," *Frontiers in Neuroinformatics,* vol. 8, pp. 34-34, 2014.

[26] P. R. Mouton, H. A. Phoulady, D. Goldgof, L. O. Hall, M. Gordon and D. Morgan, "Unbiased estimation of cell number using the automatic optical fractionator," *Journal of Chemical Neuroanatomy,* vol. 80, 2017.

[27] A. Mobiny and H. V. Nguyen, "Fast CapsNet for lung cancer screening," *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 741-749, 2018.

[28] P. Yuan, A. Mobiny, J. Jahanipour, X. Li, P. A. Cicalese, B. Roysam, V. M. Patel, M. Dragan and H. V. Nguyen, "Few Is Enough: Task-Augmented Active Meta-Learning for Brain Cell Classification.," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2020.

[29] L. C. Seamer, C. B. Bagwell, L. Barden, D. Redelman, G. C. Salzman, J. C. S. Wood and R. F. Murphy, "Proposed new data file standard for flow cytometry, version FCS 3.0.," *Cytometry*, vol. 28, no. 2, pp. 118-122, 1997.

[30] J. Spidlen, W. Moore, D. Parks, M. Goldberg, C. Bray, P. Bierre, P. Gorombey, B. Hyun, M. Hubbard, S. Lange, R. Lefebvre, R. Leif, D. Novo, L. Ostruszka, A. Treister, J. Wood, R. F. Murphy, M. Roederer, D. Sudar, R. Zigon and R. R. Brinkman, "Data File Standard for Flow Cytometry, version FCS 3.1.," *Cytometry Part A*, vol. 77, no. 1, pp. 97-100, 2009.

[31] S. Shekhar and Y. Huang, "Discovering Spatial Co-location Patterns: A Summary of Results," in *SSTD '01 Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, 2001.

[32] X. Zhang, N. Mamoulis, D. W. Cheung and Y. Shou, "Fast mining of spatial collocations," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

[33] H. Kettenmann and A. Verkhratsky, "Glial cells: neuroglia," 2016.

[34] W. J. Streit, "Microglia as neuroprotective, immunocompetent cells of the CNS," *Glia*, vol. 40, no. 2, pp. 133-139, 2002.

[35] H. Wake, A. J. Moorhouse, A. Miyamoto and J. Nabekura, "Microglia: actively surveying and shaping neuronal circuit structure and function," *Trends in Neurosciences*, vol. 36, no. 4, pp. 209-217, 2013.

[36] J. T. Morgan, G. Chana, I. Abramson, K. Semendeferi, E. Courchesne and I. P. Everall, "Abnormal microglial–neuronal spatial organization in the dorsolateral prefrontal cortex in autism," *Brain Research*, vol. 1456, pp. 72-81, 2012.

[37] S. E. Arnold, D. D. Ruscheinsky and L.-Y. Han, "Further evidence of abnormal cytoarchitecture of the entorhinal cortex in schizophrenia using spatial point pattern analyses," *Biological Psychiatry,* vol. 42, no. 8, pp. 639-647, 1997.

[38] B. Yener, "Cell-graphs: image-driven modeling of structure-function relationship," *Communications of The ACM,* vol. 60, no. 1, pp. 74-84, 2016.

[39] X. Yao, L. Chen, L. Peng and T. Chi, "A co-location pattern-mining algorithm with a density-weighted distance thresholding consideration," *Information Sciences,* vol. 396, pp. 144-161, 2017.

[40] S. Kim and E. P. Xing, "Tree-guided group lasso for multi-task regression with structured sparsity,," in *Proceedings of the 27th International Conference on Machine Learning,* 2010.

[41] T. T. Wu and K. Lange, "Coordinate descent algorithms for lasso penalized regression," *The Annals of Applied Statistics,* vol. 2, no. 1, pp. 224-244, 2008.

[42] J. Tang, S. Alelyani and H. Liu, "Feature selection for classification: A review," 2014, pp. 37-64.

[43] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of The Royal Statistical Society Series B-statistical Methodology*, vol. 68, no. 1, pp. 49-67, 2006.

[44] J. Friedman, T. Hastie and R. Tibshirani, "A note on the group lasso and a sparse group lasso," *arXiv preprint arXiv:1001.0736*, 2010.

[45] J. Zhou, E. D. Gennatas, J. H. Kramer, B. L. Miller and W. W. Seeley, "Predicting regional neurodegeneration from the healthy brain functional connectome.," *Neuron*, vol. 73, no. 6, pp. 1216-1227, 2012.

[46] W. Pirie, "Spearman rank correlation coefficient," *Encyclopedia of Statistical Sciences*, 2006.

[47] I. Kamkar, S. K. Gupta, D. Phung and S. Venkatesh, "Stable feature selection for clinical prediction," *Journal of Biomedical Informatics*, vol. 53, pp. 277-290, 2015.

[48] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, 2011.

[49] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of The ACM,* vol. 24, no. 6, pp. 381-395, 1981.

[50] H. Narasimha-Iyer, A. Can, B. Roysam, V. Stewart, H. Tanenbaum, A. Majerovics and H. Singh, "Robust detection and classification of longitudinal changes in color retinal fundus images for monitoring diabetic retinopathy," *IEEE Transactions on Biomedical Engineering,* vol. 53, no. 6, pp. 1084-1098, 2006.

[51] R. A. Maronna and V. J. Yohai, "Robust regression with both continuous and categorical predictors," *Journal of Statistical Planning and Inference,* vol. 89, no. 1, pp. 197-214, 2000.

[52] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[53] H. Bay, T. Tuytelaars and L. V. Gool, "SURF: speeded up robust features," *European Conference on Computer Vision,* vol. 3951, pp. 404-417, 2006.

[54] E. Rosten, R. Porter and T. Drummond, "Faster and Better: A machine learning approach to corner detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 32, no. 1, pp. 105-119, 2010.

[55] M. Calonder, V. Lepetit, C. Strecha and P. Fua, "BRIEF: binary robust independent elementary features," in *ECCV'10 Proceedings of the 11th European conference on Computer vision: Part IV*, 2010.

[56] G. Balakrishnan, A. Zhao, M. R. Sabuncu, J. Guttag and A. V. Dalca, "VoxelMorph: A learning framework for deformable medical image registration," *IEEE Transactions on Medical Imaging*, vol. 38, no. 8, pp. 1788-1800, 2019.

[57] Y. Zhu, Z. Zhou, G. Liao and K. Yuan, "New loss functions for medical image registration based on VoxelMorph," in *Medical Imaging 2020: Image Processing*, 2020.

[58] S. V. D. Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart and T. Yu, "Scikit-image: image processing in Python," *PeerJ*, vol. 2, no. 1, 2014.

[59] D. J. S. Robinson, An introduction to abstract algebra, 2003.

[60] N. Otsu, "A threshold selection method from gray level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62-66, 1979.

[61] D. Liu and J. Yu, "Otsu method and k-means," in *Ninth International Conference on Hybrid Intelligent Systems*, 2009.

[62] H. J. G. Gundersen, "Notes on the estimation of the numerical density of arbitrary profiles: the edge effect," *Journal of Microscopy,* vol. 111, no. 2, pp. 219-223, 1977.

[63] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, "How transferable are features in deep neural networks," in *Advances in Neural Information Processing Systems 27*, 2014.

[64] A. L. Tenza, "Scattering transform for breast cancer detection," , 2015.

[65] S. Shekhar, M. R. Evans, J. M. Kang and P. Mohan, "Identifying patterns in spatial information: a survey of methods," *Wiley Interdisciplinary Reviews-Data Mining and Knowledge Discovery,* vol. 1, no. 3, pp. 193-214, 2011.

[66] M. Niepert, M. Ahmed and K. Kutzkov, "Learning convolutional neural networks for graphs," in *ICML'16 Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, 2016.

[67] H. Kettenmann, F. Kirchhoff and A. Verkhratsky, "Microglia: new roles for the synaptic stripper," *Neuron,* vol. 77, no. 1, pp. 10-18, 2013.

[68] A. Getis and J. Franklin, "Second-Order Neighborhood Analysis of Mapped Point Patterns," *Ecology,* vol. 68, no. 3, pp. 473-477, 1987.

[69] P. J. Diggle and A. G. Chetwynd, "Second-order analysis of spatial clustering for inhomogeneous populations.," *Biometrics,* vol. 47, no. 3, pp. 1155-1163, 1991.

[70] S. Boillée, C. V. Velde and D. W. Cleveland, "ALS: a disease of motor neurons and their nonneuronal neighbors.," *Neuron,* vol. 52, no. 1, pp. 39-59, 2006.

[71] S. Wan, M. W. Mak and S.-Y. Kung, "Sparse regressions for predicting and interpreting subcellular localization of multi-label proteins," *BMC Bioinformatics,* vol. 17, no. 1, pp. 97-97, 2016.

[72] H. Kettenmann and A. Verkhratsky, "Neuroglia: the 150 years after," *Trends in Neurosciences,* vol. 31, no. 12, pp. 653-659, 2008.

[73] J. Wang and J. Ye, "Multi-layer feature reduction for tree structured group lasso via hierarchical projection," in *Proceedings of the 28th International Conference on Neural Information Processing Systems,Vol.1,* 2015.

[74] R. Tibshirani, "Regression shrinkage and selection via the LASSO," *Journal of the Royal Statistical Society Series B-methodological,* vol. 58, no. 1, pp. 267-288, 1996.

[75] R. Real and J. M. Vargas, "The probabilistic basis of Jaccard's index of similarity," *Systematic Biology,* vol. 45, no. 3, pp. 380-385, 1996.

[76] M. Liu, D. Zhang, P. T. Yap and D. Shen, "Tree-guided sparse coding for brain disease classification," *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI),* vol. 15, pp. 239-247, 2012.

[77] L. V. Iterson and A. S. Kaufman, "SEIN Stichting Epilepsie Instellingen Nederland & School De Waterlelie, Cruquius," 2009.

[78] T. Hastie, R. Tibshirani and M. Wainwright, Statistical learning with sparsity: the lasso and generalizations, CRC press, 2015.

[79] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012.

[80] I. Goodfellow, Y. Bengio and A. Courville, "Deep learning," Cambridge: MIT press, 2016, pp. Vol. 1, No. 2..

[81] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature pyramid networks for object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[82] D. Liu, D. Zhang, Y. Song, C. Zhang, F. Zhang, L. O'Donnell and W. Cai, "Nuclei segmentation via a deep panoptic model with semantic feature fusion.," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

[83] J. B. Maitin-Shepard, V. Jain, M. Januszewski, P. H. Li and P. Abbeel, "Combinatorial energy learning for image segmentation," in *Advances in Neural Information Processing Systems*, 2016.

[84] D. Zhang, Y. Song, D. Liu, H. Jia, S. Liu, Y. Xia, H. Huang and W. Cai, "Panoptic segmentation with an end-to-end cell R-cnn for pathology image analysis," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2018.

[85] H. Yu, K. P. Lim, S. Xiong, L. P. Tan and W. Shim, "Functional morphometric analysis in cellular behaviors: shape and size matter," *Advanced Healthcare Materials*, vol. 2, no. 9, pp. 1188-1197, 2013.

[86] B. D. Ripley, "The second-order analysis of stationary point processes," *Journal of Applied Probability*, vol. 13, no. 2, pp. 255-266, 1976.

[87] K. Ohsawa and S. Kohsaka, "Dynamic motility of microglia: purinergic modulation of microglial movement in the normal and pathological brain," *Glia*, vol. 59, no. 12, pp. 1793-1799, 2011.

[88] M. L. Narro, F. Yang, R. Kraft, C. Wenk, A. Efrat and L. L. Restifo, "NeuronMetrics: software for semi-automated processing of cultured neuron images," *Brain Research,* vol. 1138, no. 1, pp. 57-75, 2007.

[89] L. McKeen-Polizzotti, K. M. Henderson, B. Oztan, C. C. Bilgin, B. Yener and G. E. Plopper, "Quantitative metric profiles capture three-dimensional temporospatial architecture to discriminate cellular functional states," *BMC Medical Imaging,* vol. 11, no. 1, pp. 11-11, 2011.

[90] S. Mallat, "Group invariant scattering," *Communications on Pure and Applied Mathematics,* vol. 65, no. 10, pp. 1331-1398, 2012.

[91] M. López-Hidalgo, W. B. Hoover and J. Schummers, "Spatial organization of astrocytes in ferret visual cortex," *The Journal of Comparative Neurology*, vol. 524, no. 17, pp. 3561-3576, 2016.

[92] A. S. Fotheringham and P. Rogerson, Spatial Analysis and GIS, vol. 39, CRC Press, 2013, p. 238.

[93] P. M. Dixon, "R ipley's K Function," *Wiley StatsRef: Statistics Reference Online*, 2014.

[94] G.-J. d. Haan, R. Thijs, C. Deckers, I. Wildenberg and C. V. Laerhoven, "Stichting Epilepsie Instellingen Nederland [SEIN] , Heemstede and Zwolle, The Netherlands," *Epilepsy & Behavior*, vol. 76, 2017.

APPENDIX

A. Loss Functions of MRCNN

Loss Function : $\boldsymbol{L = L_{cls} + L_{box} + L_{mask}}$

$\boldsymbol{L_{cls}}$: Cross Entropy Loss with Softmax

$$\boldsymbol{p_i = softmax(y_i)}$$

$$\boldsymbol{L_{cls} = -\frac{1}{N}\sum_i y_i\, log(p_i) + (1 - y_i) log(1 - p_i)}$$

$\boldsymbol{L_{box}}$: Soft L1 Norm

$\boldsymbol{t(k)}$: a scale-invariant translation and

log-space function

$$\boldsymbol{L_{box}(u, v) = \sum L_1^{smooth}\,(t_u, t_v)}$$

$\boldsymbol{L_{mask}}$: Cross Entropy Loss ($\boldsymbol{Km^2}$)

$$\mathbf{L_{mask} = -\frac{1}{m^2}\sum_{i} y_{ij}\, log(\widehat{y_{ij}}^{k}) + (1 - y_{ij}) log(1 - \widehat{y_{ij}}^{k})}$$

Decouple the mask and class prediction

→ the classification does not depend on mask prediction

B. Figures for MRCNN Modules

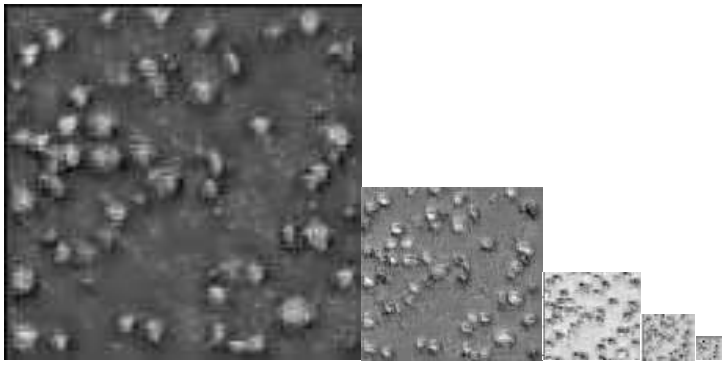


Fig 1. Square Feature Maps of after FPN (with the weights of 512,256,128,64,32)

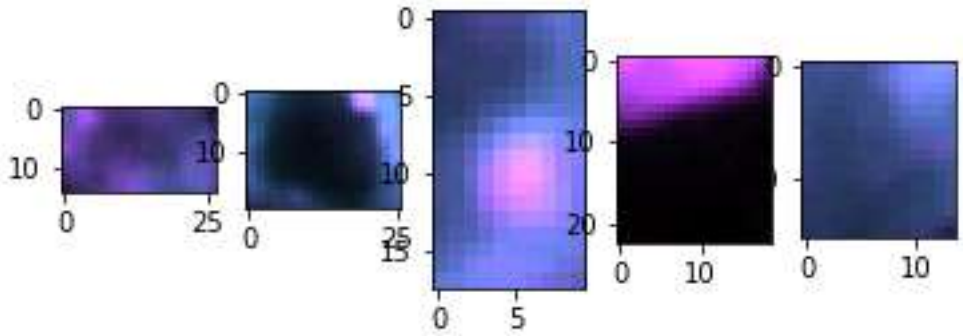


Fig 2. ROIs of 5 patches

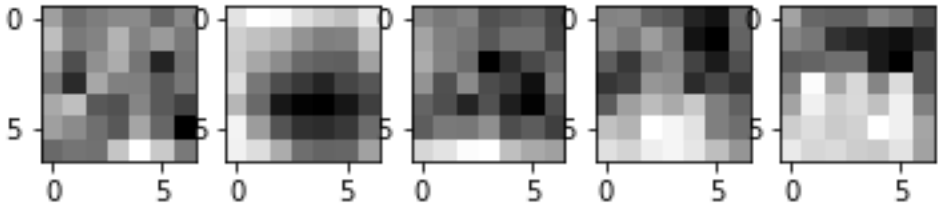


Fig 3. Fixed-size feature maps of 5 patches (1 of 255)

C. Recommended System Requirements

[1] Multi-thread CPU (e.g. INTEL Core i7)

[2] NVIDIA CUDA-enabled GPU: (e.g. Tesla K80 in our experiment);

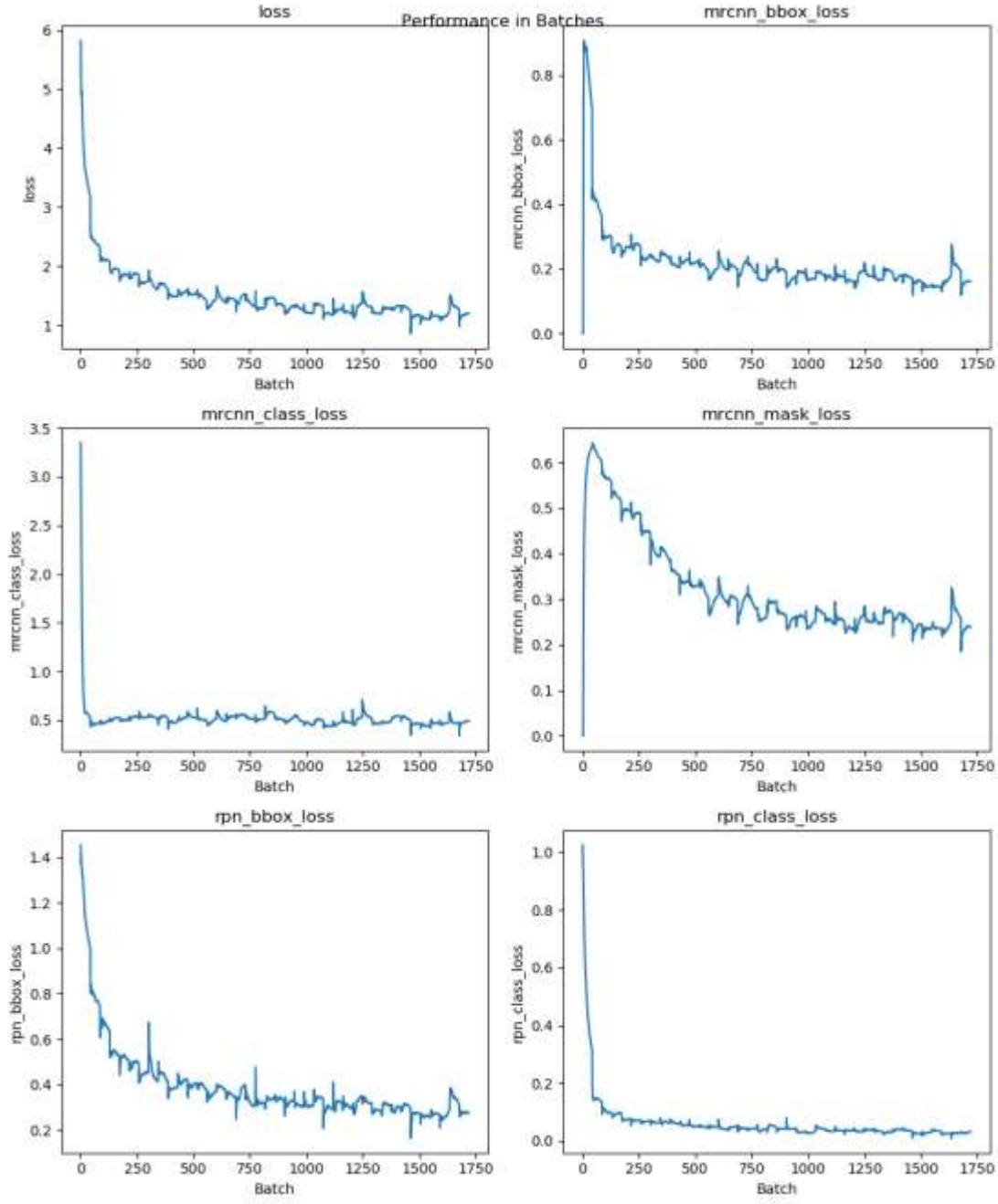
[3] Hard-drive: 1TB solid state hard drive;

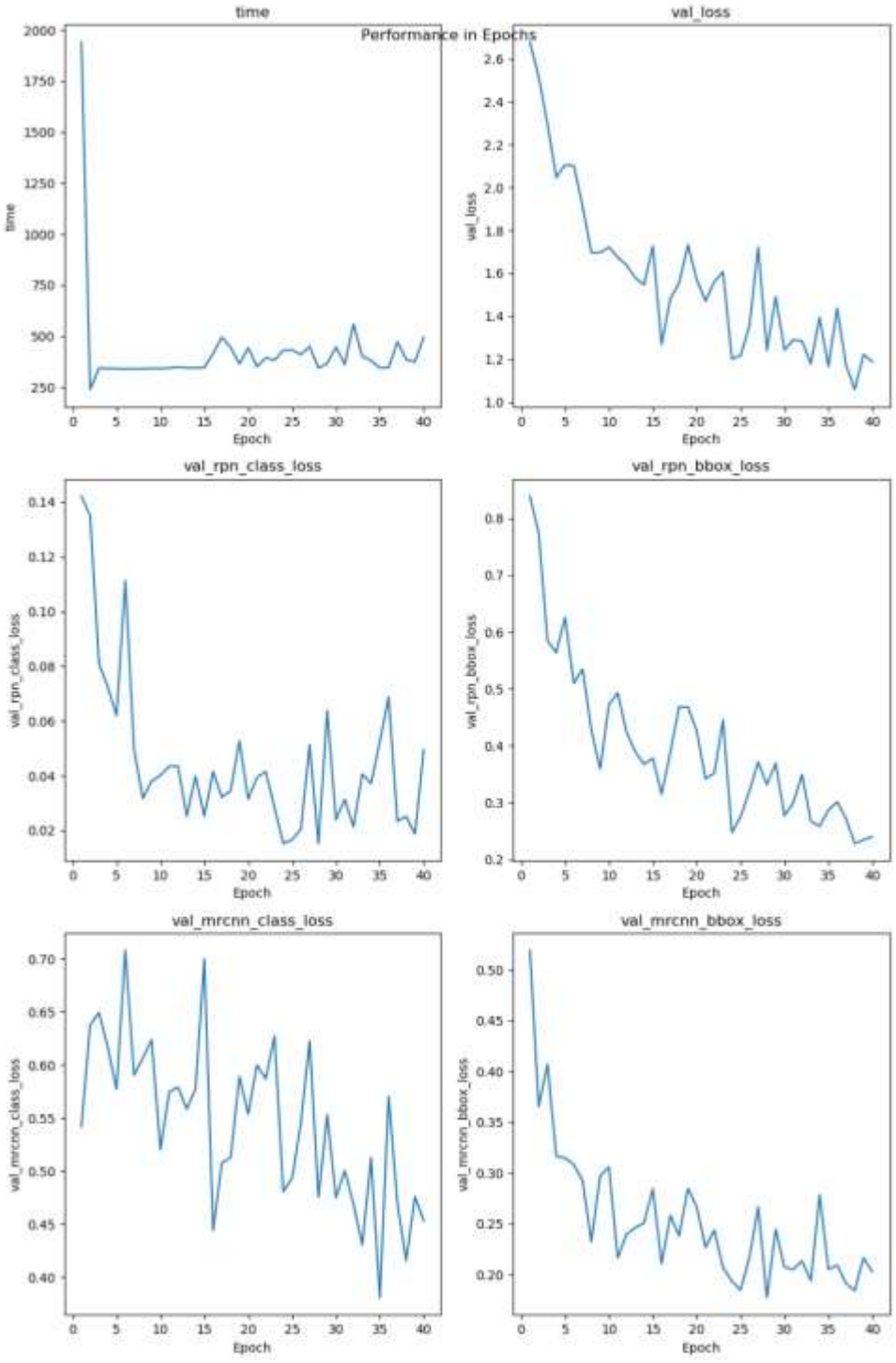
[4] Operating system: Linux (Redhat or Ubuntu);

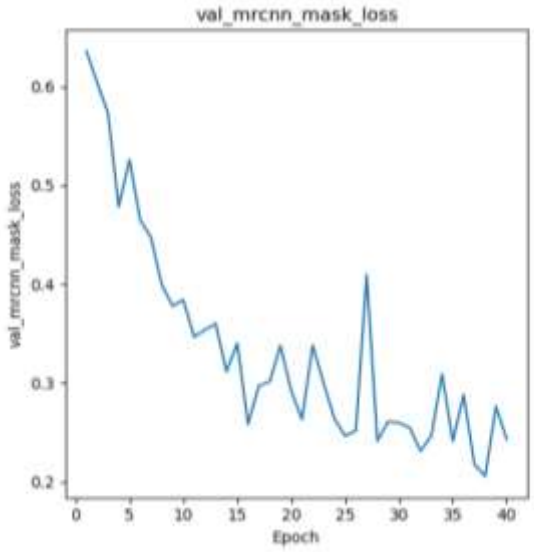
[5] CUDA/10.0.130

[6] Tensorflow v2.1 and object detection APIs;

D. Loss Performance of MRCNN




Performance in Epochs
time
Epoch
val_loss
val_rpn_class_loss
val_rpn_bbox_loss
val_mrcnn_class_loss
val_mrcnn_bbox_loss


val_mrcnn_mask_loss
val_mrcnn_mask_loss
0.6
0.5
0.4
0.3
0.2
0
5
10
15
20
25
30
35
40
Epoch