INTERACTIVE MINING FOR LARGE-SCALE NEURO-MORPHOLOGICAL DATASETS

by

Zhongyu Li

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2018

Approved by:

Shaoting Zhang

Xinghua Shi

Aidong Lu

Min Shin

ABSTRACT

ZHONGYU LI. Interactive Mining for Large-scale Neuro-Morphological Datasets.
(Under the direction of SHAOTING ZHANG)

In this dissertation, we aim to investigate advanced methods for the computational analytics of large-scale neuro-morphological datasets, which can help neuroscientists interactively explore neurons in real-time. Particularly, we tackle the neuro-morphological analytics into three inter-related components: 1) quantitative descriptions for 3D neuron morphologies, i.e., computing effective features that can differentiate subtle difference among massive neurons; 2) large-scale neuron mining, i.e., efficiently indexing neurons with similar morphologies in large-scale datasets; 3) interactively neuron exploration and visualization, i.e., developing neuron visualization tools and bring human in the loop to explore neurons in an interactive and immersive manner. We propose a series of methods in tackling problems related to the above three components. Regarding the quantitative description, we develop a deep learning framework based on neuron projection, which can transform 3D neurons into 2D images and learn effective neuron features. Regarding the large-scale neuron mining, binary coding methods are introduced, which can transform feature vectors into short binary codes for real-time indexing and mining. Regarding the interactive neuron exploration, we visualize 3D neurons using augmented reality (AR) techniques, where users can provide relevance feedback to further improve the mining performance. The proposed methods are validated on the currently largest neuron database including more than $58,000$ neurons, achieving state-of-the-art performance in comparison with other related methods. More importantly, we demonstrate use cases of our framework in multiple neuron analysis and exploration tasks, showing its potential benefits in facilitating the research of neuroscience.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Prof. Shaoting Zhang for the continuous support of my Ph.D study and related research, for his patience, motivation, and immerse knowledge. He has been motivating and encouraging me to explore various problems in computer vision, machine learning and neuroscience. His guidance helped me in all the time of research and writing of this thesis. None of the work in this thesis would have happened without him.

I would like to thank other committee members: Prof. Xinghua Shi, Prof. Aidong Lu, Prof. Min Shin for their insightful comments, encouragement and valuable suggestions regarding this thesis. It is an honor for me to have each of them serves in my committee.

Thanks to collaborators during my Ph.D study who have contributed many ideas and helps in my research works, especially to Prof. Fumin Shen (University of Electronic Science and Technology of China), Prof. Ruogu Fang (University of Florida), Prof. Henning Müller (University of Applied Sciences Western Switzerland), Erik Butler and Chaowei Fang (The University of Hong Kong). Also thanks to my labmates in the Video and Image Analysis Lab for their kindly helps, and for all the fun we have had in the last three years, especially to Prof. Richard Souvenir, Prof. Emanuela Marasco, Xiaofan Zhang, Bin Kong, Lance Rice, and Junjie Shan.

Special thanks to my friends and colleagues. I benefited a lot from their friendship and help, and they made my years at UNC Charlotte a real pleasure.

DEDICATION

This dissertation is dedicated to my parents: Lixin Li and Congfang Li.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

The brain is the most complex piece of organized matter in the known universe. Understanding the brain function in health and disease is one of the greatest challenges facing modern science. The human brain contains billions of neurons and trillions of synapses. Investigating the morphology of massive neurons is an important topic in brain and neuron science, since morphology plays a major role in determining neurons' connectivity and thus function. Recent frontiers (e.g., BigNeuron [3] and NeuroMorpho [1] projects) have greatly facilitated the research in neuron morphology, resulting in an increasing number of neurons that are reconstructed and added to these public repositories. However, the sheer volume and complexity of these data pose significant challenges for computational analysis, preventing the realization of the full potential of these data, e.g., efficiently finding neurons sharing similar morphologies, identifying neuron types, correlating neuron morphologies with properties, all of which require a deep and exhaustive mining of large-scale neuron datasets.

On the other hand, data mining and knowledge discovery is a long-term research topic in data science and machine learning, which have been widely applied in many academic and industrial scenarios (e.g., computer vision, social media). We advocate that this can also benefit the research in neuronal morphology. Previously, neuron mining only considers certain neuron properties (e.g., brain regions, transmitters) with few reconstructed data sets [4, 5, 6, 7]. In recent years, in response to the above-mentioned pressing need, researchers have started to investigate neuron mining, and achieved preliminary results in neuron retrieval, clustering and comparison [8, 9, 10]. Nonetheless, most of these approaches are not sufficiently efficient or robust when handling large-scale databases, as they are usually simple variants of data mining methods from other domains.

In fact, there are multiple unique challenges in the field of neuromorphological analytics, which may prevent the direct application of existing approaches to neuron mining: 1) The differences among neuron morphologies can be very subtle, i.e., fine-grained differences, as neurons have tree-structured topology, which is hard to represent and differentiate. In addition, some functionally unrelated neurons can be visually similar. 2) Fast-growing neuron databases restrict the computational efficiency. For example, there are more than $80,000$ reconstructed neurons that have been added to the NeuroMorpho database [1]. After the completion of the BigNeuron project [3], a significant number of neurons will be reconstructed. Efficient mining for large-scale and continuously increasing databases is an urgent demand in recent trends. 3) The annotations of neurons are usually not ready for supervised machine learning. Some existing databases only provide coarse annotations (e.g., primary brain regions, cell classes), which are not sufficient to provide comprehensive labels for different types of neurons. It is desired to bring domain experts into the computational loop, such that only a small amount of supervised information will be incorporated to ensure the performance. Additionally, how to visualize neuron morphologies and the analytical results for user exploration are also unsolved challenges.

Considering the above discussed challenges, the goal of this dissertation is to design an effective framework for neuroscientists to interactively explore large-scale neuron databases, in real-time. Different from traditional approaches based on quantitative computation and experimentation [11, 12, 13, 14], this dissertation presents new avenues which assembles the massive neuron morphologies and provides an unified framework to explore and analyze different types of neurons, which will demonstrate its superiority in efficiency, accuracy, scalability and robustness. Particularly, as shown in Fig. 1.1, the neuron mining framework starts with feature representation

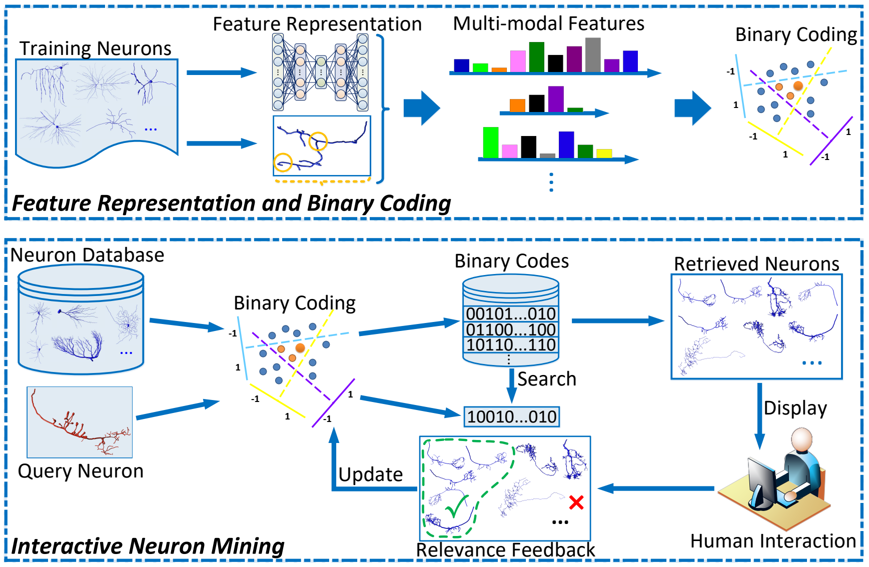


Figure 1.1: Overview of our proposed framework, including neuronal feature representation, binary coding and interactive neuronal mining.

for the quantitative description of neuromorphologies, which is the basic step for neuron exploration. Subsequently, instead of directly mining neurons using feature vectors, binary coding techniques are introduced which can enable real-time indexing and clustering by compressing long features into short binary codes. Afterwards, interactive strategy are employed to improve the retrieval precision with a small amount of domain experts' feedback. Based on the fine-tuned retrieval results, multiple application can be developed, such as neuron clustering, comparison, classification and further high-level knowledge discovery. All the analytical results will be organized in an online website for users to mining neurons in real-time. Additionally, advances of augmented reality (AR) techniques with Microsoft HoloLens headsets are employed for 3D neuron visualization and exploration.

In this dissertation, the research work for computational neuron analytics has the following contributions, which will be elaborated in each chapter:

1. A deep learning based framework is proposed for the feature representation of 3D neuron morphology. The framework first presents a neuron projection method to transform each 3D neuron into 2D images with three angles of view.

The projection strategy can preserve the spatial morphologies from the original data and transform 3D neurons in a suitable format for deep feature learning. Then an unsupervised deep neural network is introduced to automatically learn neuron features from the projected 2D images. To the best of our knowledge, this is the first attempt that applies deep learning techniques in the analytics of 3D neuron morphology.

2. Two binary coding methods are introduced in tacking the exploration of large-scale neuron database. The first binary coding method is based on the maximum inner product search (MIPS), which can learn effective coding functions for the linearly inseparable and differentially subtle neuron morphologies. The second binary coding method is based on the online updating schemes, which can efficiently update the coding functions on-the-fly from the continuously expanding neuron databases. In addition, feature hierarchy strategy is proposed to reveal the similarity level of different neuron features during binary coding.

3. To further improve the neuronal mining performance and reduce the semantic gap between neuron morphologies and functional properties, an interactive strategy is presented to bring users in the loop. The interactive strategy can re-consider and re-rank the mining results based on the relevance feedback provided by users. More importantly, the relevance feedback can be also employed to update the similarity measure among neurons in the whole database.

4. Based on the above methods designed for neuron mining, multiple tools are developed in assisting neuron exploration for users, including an online website for real-time neuronal retrieval, and a neuron visualization program using advanced techniques of augmented reality (AR) with Microsoft HoloLens headset. Moreover, this dissertation also demonstrates several use cases for neuron identification, comparison and pattern discovery.

The rest of this dissertation is organized as follows: Chapter 2 reviews recent frontiers of neuron morphology, content-based image retrieval and binary coding. Chapter 3 provides solutions for the quantitative descriptions of 3D neuron morphology. Followed by Chapter 4 introduces the algorithm details of large-scale methods for neuron mining. Chapter 5 presents the algorithm design of interactive neuron exploration, including relevance feedback based mining and neuronal visualization. Afterwards, Chapter 6 demonstrates the developed online system for real-time neuronal retrieval and augmented reality program for neuronal visualization. Finally, Chapter 7 concludes the dissertation and discusses future works.

CHAPTER 2: RELATED WORKS

The works in this dissertation have a strong multidisciplinary component that involves a nexus of ideas from neuroscience, machine learning, and information retrieval. This chapter provides an introduction of related frontiers in the field of 3D neuron morphology, binary coding, and content-based image retrieval.

2.1 Neuron Morphology

In recent years, the research of neuron morphology has a significantly improvement, owing to the advances of neuron tracing and reconstruction techniques. Neuron tracing aims to manually or automatically reconstruct 3D neuron morphologies from fluorescence or electron microscopy images. Compared with 2D neuronal images, 3D morphological data reflect spatial structure of the neuron cell with more comprehensive information [15], e.g., fine-grained details of structures, brunches, bifurcations. From original microscopy images to the 3D neuron morphological data, neuron tracing consists a number of processing step, including image preprocessing (e.g., noise reduction, deconvolution, mosaicking), segmentation (e.g., soma, dendritic trees, spines, axons segmentation), reconstruction and connection [16, 17, 18, 19, 20]. In recent years, there are many tracing and reconstruction software released which make 3D neuron morphological data easier to acquire. Fig. 2.1 illustrates a microscopy image from neuron slices [21] and its corresponding 3D morphological data through *Vaa3D* software [2]. As shown in Fig. 2.1(b), the 3D morphological data provides more precise details for neuron cells which can facilitate the research of quantitative descriptions for further retrieval and analysis.

Benefited from algorithms and software of neuron tracing, more and more 3D mor-

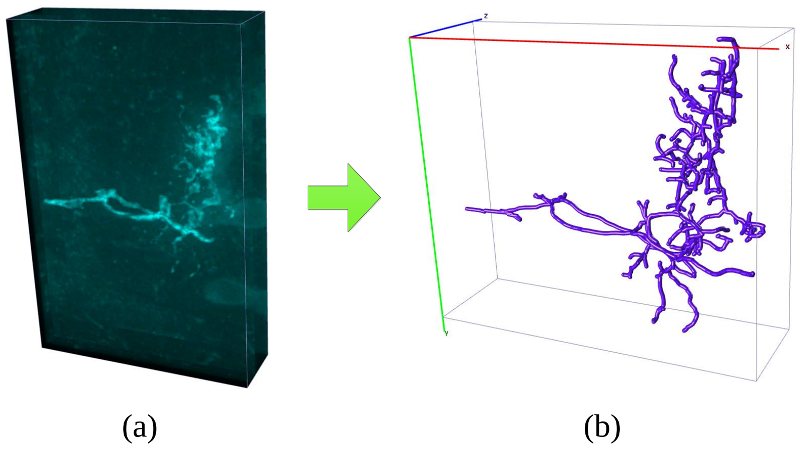


Figure 2.1: From microscopy neuron image to 3D morphological data: (a) original microscopy slices; (b) 3D neuron morphology with quantitative measures.

phological databases are released in recent years. For example, FlyCircuit [22] is one of the early database that collected tens of thousands of reconstructed neurons from the Drosophila's brain. More recently, the successes of BigNeuron project [3] have greatly accelerate the generation of 3D neuron data, where massive neurons are reconstructed and added to the currently largest neuron morphological database, i.e., NeuroMorpho [1]. As shown in Fig. 2.2 (images adapted from NeuroMorpho.Org [1]), the number of reconstructed neurons in NeuroMorpho [1] has increased from $2,000$ to $86,000$ in recent ten years, which are still fast growing. The NeuroMorpho database currently include the reconstructed neurons from 53 species, 278 brain regions and 514 cell types, where the data were provided by 455 archives around the world. This huge neuro-morphological database provide a good opportunity for us to explore and analyze neurons in new ways.

When analyzing the 3D neuron morphology, one critical problem is feature representation, i.e., how to quantitatively describe 3D neurons which can accurately indicate their similarity and difference among original data shape. Unlike 2D images which can extract features with many well-studied algorithms, how to extract effective features from 3D neuron data is still a challenging task. For neuron cells, from axon

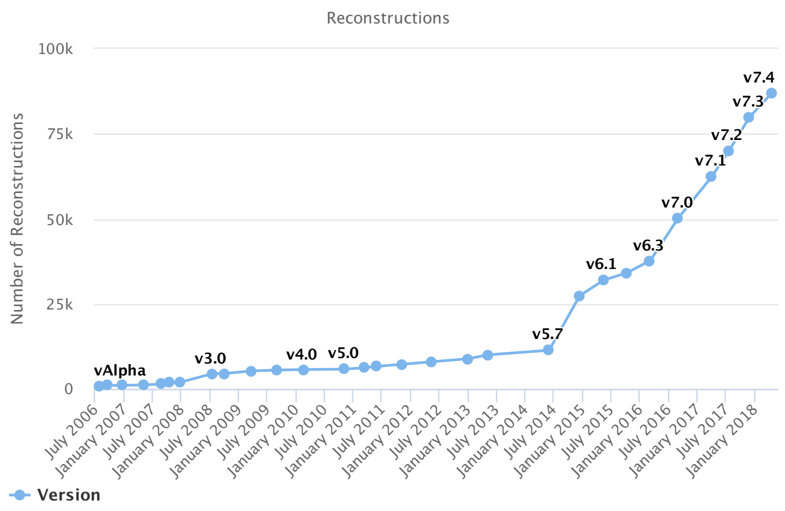


Figure 2.2: Number of neurons increased in NeuroMorpho [1] database, which demonstrated a fast growing in recent years.

to soma and then to dendrite, they usually express tree-topological structures. In previous articles [23, 24, 25, 8, 26, 27, 28, 29], researchers defined multiple quantitative measurements based on this tree-topological structure, such as neuron's total height, number of branches, the Euclidean distance from compartments to somas, the angle between two terminal branches etc. Then these quantitative measurements are combined together as the feature representation of each 3D neuron data. However, these hand-crafted features are designed for previously neuron analytical tasks with small sized datasets (e.g., including hundreds to thousands of neurons). When tackling much larger neuron datasets (e.g., including tens of thousands or more neurons), the representational performance cannot be guaranteed. Therefore, new methods should be developed for the feature representation of 3D neuron morphology.

2.2 Binary Coding and Hashing

In recent years, binary coding and hashing have been widely used to solve large-scale problems in machine learning, computer vision and other related areas. After feature representation of 3D neuron morphology, how to efficiently tackle the large-scale database is also a critical problem. For the binary coding and hashing methods, by compressing long feature vectors into short binary codes, the storage and similarity search will be much more efficient in binary Hamming space compared with high dimensional feature space. Therefore, binary coding and hashing techniques can be also employed to compress neuronal features and solve analytical problems of large-scale neuron databases. However, the key question is how to obtain binary coding or hashing functions which can not only split feature vectors via binary codes, but also keep similarities among original data. Recently, multiple articles [30, 31] reviewed recent advances of binary coding and hashing, with various types of methods, including data-independent and data-dependent, supervised and unsupervised, linear and nonlinear hashing algorithms. This section briefly review the taxonomy and methods of binary coding and hashing, and discuss their applicability for the large-scale problems of neuronal datasets.

Data-independent methods usually design generalized binary coding or hashing functions to compact any given datasets. Locality-Sensitive Hashing (LSH) and its variants [32, 33, 34] are one of the most representative data-independent methods. This type of methods ensures the data similarity with long binary hash bits and multiple hash tables. However, these methods may not suit neuron retrieval problem because of the specificity of the datasets, i.e., the tree structure of neurons and their non-separability in Euclidean space make them different with other datasets. Another category is the data-dependent methods, whose binary coding functions are obtained through learning from given datasets for particular problems. In recent years, a large number of learning-based methods are proposed, including but not limited to Iterative Quantization (ITQ) [35], Isotropic Hashing (IsoHash) [36], Minimal Loss Hashing (MLH) [37, 38], FastHash [39], etc. Some of them are supervised methods, i.e., including classification labels, which have already achieved excellent performance in large-scale retrieval. But current neuron database such as NeuroMorpho [1] lack enough normative annotations for every neuron. Therefore, unsupervised binary coding should be a better choice for neuron retrieval. Another taxonomy of binary coding methods is based on the category of coding function, i.e., whether the coding function can transform features into linear or non-linear space. As mentioned before, neuron morphological features are measured based on the tree-like structure, while the tree-like structures of unrelated neurons sometimes appear similar which makes them hard to distinguish in linear space. Compared with linear binary coding/hashing algorithms, nonlinear algorithms usually generate more sensitive binary codes to divide data in nonlinear space. Representative methods such as Kernel-Based Supervised Hashing (KSH) [40], Spectral Hashing [41], Anchor Graph Hashing (AGH) [42], Inductive Manifold Hashing (IMH) [42], etc., they construct coding functions based on nonlinear kernel matrix or manifold structure. However, one disadvantage of the above mentioned nonlinear methods is that they fail to consider the diversity among different features when learning binary codes.

2.3 Content-based Image Retrieval

Over the past 25 years, Content-based Image Retrieval (CBIR) has been one of the most vivid research topics in the field of computer vision. Many CBIR methods are developed for accurate and efficient image retrieval. Especially in recent years, with the ever-increasing number of digital images (e.g., ImageNet [43], COCO [44], PASCAL VOC [45], etc), CBIR has moved towards the era of big data. Massive amounts of images can provide rich information for comparison and analysis, and thus facilitate the generation of new algorithms and techniques that can tackle image retrieval in large databases. As a comprehensive application in computer vision and machine learning, CBIR has developed many branches for different concerns and targets. In general, content-based image retrieval can be divided into two stages, i.e., feature extraction to represent images and feature indexing to find relevant samples. Deep learning [46] is one of the most popular methods for feature representation that is particularly suitable for large image databases, where massive amounts of data can boost the retrieval performance by training deep and complex neural networks with millions of parameters [47, 48]. For the feature indexing, the key problem is computational efficiency, i.e., similarity searching in millions of images with thousand dimensional features vectors. Methods such as vocabulary trees [49] and hashing [30] can efficiently tackle this problem, either through changing the indexing structure or compressing the original features.

In addition to the feature extraction and indexing, when the retrieved databases do not have enough annotations, simply analyzing massive image data may not results in accurate retrieval results, due to the existed gap between images and their semantic information. In order to improve retrieval performance and reduce the semantic gap, some CBIR systems introduce domain experts/users in the loop, which can interactively provide relevance feedback for the previous retrieval results. Generally, for an image query and its coarse retrieval results, there are three kinds models to give relevance feedback: (1) positive feedback, where the users only need to select relevant images; (2) positive-negative feedback, where the users need to select both relevant and irrelevant images; (3) positive-neutral-negative feedback, where the users need to specify the degree of relevance for all the retrieved images. A comprehensive review of the early work on relevance feedback for CBIR is presented in [50]. Most of the early approaches use the marked images as individual queries and combine the retrieval results to refine the similarity weights of relevant images [51, 52]. In recent years, many algorithms in the machine learning field have been used for the interactive CBIR problem, e.g., random forests [53], graph-cut [54], random walk [55], manifold learning [56]. All these methods can improve the retrieval performance with several rounds of interactive feedback. However, it is still a challenging issue when applying user interaction in large-scale neuron databases.

CHAPTER 3: QUANTITATIVE DESCRIPTIONS FOR 3D NEURON MORPHOLOGY

3.1 Motivation

Understanding neuron morphology is a fundamental task to explore neuronal circuits, functional and physiological properties. Recent frontiers in neuron tracing and reconstruction (e.g., BigNeuron [57], NeuroMorpho [58]) have greatly facilitate the research of neuron morphology. Increasing number of neurons are digitally reconstructed and added to the public repositories with tens of thousands of neurons [1, 3]. For each reconstructed neuron, their morphologies are recorded in a SWC format file including a set of neuron nodes with segment types, locations, radius, and connections [59]. Accordingly, these huge amount of morphological data bring new opportunities for neuron mining and knowledge discovery.

In recent years, neuron morphology has been investigated based on computational models and machine learning techniques. Scorcioni et al. [25] first developed L-measure tool for the quantitative measurement of neuron morphology, which can compute neuroanatomical parameters from 3D reconstructed neuron data. Costa et al. [24] proposed the concept of neuromorphological space and identified the most important geometrical features in neuron cell, including neuron's total length, branch numbers, etc. With these measurement based features, multiple tools and methods have been proposed for the analytics of neuron morphology. For example, Wan et al. [8] developed *BlastNeuron* for the comparison, retrieval and clustering of 3D neuron morphology. In *BlastNeuron*, they employed L-measure tool and moment invariants as morphological features for similarity search. Costa et al. [26] presented *NBLAST* to measure pairwise neuronal similarity by considering both position and

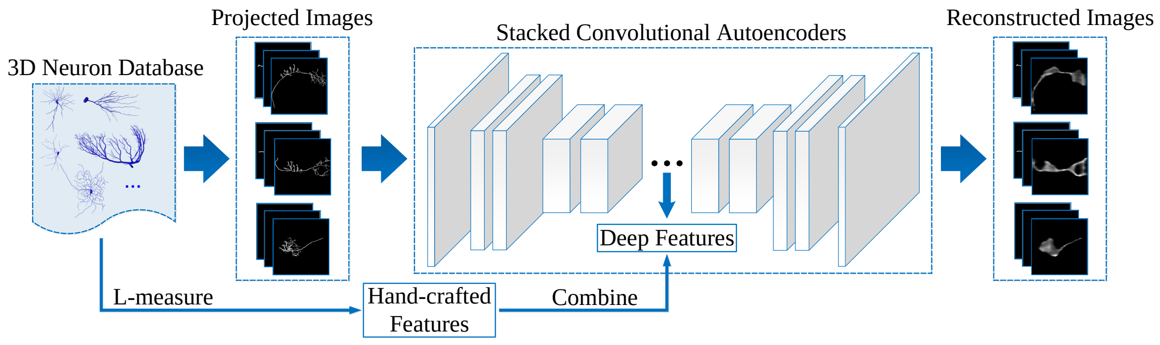


Figure 3.1: Overview of the proposed framework for the feature representation of 3D neuron morphology.

local geometry, decomposing neurons into short segments.

By employing quantitative measurements as neuronal features, the above methods have achieved many successes in the research of neuron morphology. However, with the continuously expanding of neuron databases, neurons belonging to different categories can express similar morphologies (indicating small inter-class variances), while neurons belonging to same categories can express different morphologies (indicating large intra-class variances) [60]. Thus, traditional “hand-crafted” measurements may not work well in the representation of neuron morphologies, as well as the exploration of large-scale neuron databases. On the other side, deep learning has become a kind of advanced techniques for the feature representation in many fields, such as computer vision, medical image analysis, and speech recognition [46]. Nevertheless, for the feature representation of 3D neuron morphology, directly applying deep learning methods still faces two major problems: 1) the tree-structure of neurons are usually very sparse in 3D space, which are not suitable for the training of 3D neural networks; 2) the spatial information of neuron nodes, i.e., location, radius, and connection, need to be considered but are hard to embed in deep models.

To address the above problems, this dissertation develops a deep learning based framework for the feature representation of 3D neuron morphology. Fig. 3.1 presents the overview of our framework. At first, to overcome the tree-structural sparsity, 3D neuron morphological data are transformed into 2D images through orthogonal projection. The spatial information of neuron nodes can be greatly preserved by projecting nodes' coordinates, radius, and connections in three angles of view. The projected 2D neuron images are subsequently set as input to train an unsupervised deep neural network, i.e., stacked convolutional autoencoders (SCAEs). The network learns to recover the input image by exploring intrinsic deep feature representation among neuron morphologies. After training the SCAEs model, the learned deep features are fused with the traditional hand-crafted features for comprehensive and accurate representation of 3D neuron morphologies. The fused neuron features can be further employed for similarity searching and knowledge discovery. To the best of our knowledge, this is the first attempt that applies deep learning techniques in the analytics of 3D neuron morphology.

3.2 Methodology

This section presents the theoretical and technical details of the framework for the quantitative description of 3D neuron morphology, including 3D neuron projection, deep feature representation, and feature fusion.

3.2.1 3D Neuron Projection

As discussed above, the traditional hand-crafted features are insufficient to represent each neuron in large-scale databases, new avenue should be explored for the quantitative description of 3D neuron, based on recent advances of deep learning. Considering the original neuron morphological data (i.e., the SWC format files [59]) provide the spatial coordinate of each point, an intuitive solution is to employ 3D deep neural networks that can directly learn deep features from these 3D point sets. Unfortunately, this approach is impractical due to three reasons: (1) training 3D neural networks are usually very time-consuming, particularly when tackling large neuron databases; (2) the 3D point sets and their connections in each neuron are

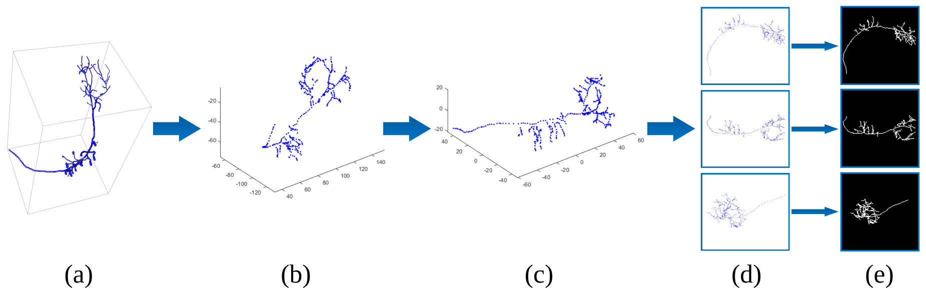


Figure 3.2: The pipeline of transforming 3D neurons into images: (a) the original neuron data visualized by Vaa3D [2]; (b) the 3D points of neuron data in initial orientation; (c) the 3D points after principal component analysis (PCA); (d) the projection of each point in three angles of view; (e) the generated binary images by linking each point with their parent points.

composed based on the tree-topological structure, which are extremely sparse in 3D space; (3) Neurons have dramatically different scales and different numbers of point sets (from hundreds to tens of thousands) that hard to be processed in a generalized framework. Therefore, how to adapt the 3D neuron data with a suitable modality for deep learning is a critical step in this feature representation task.

Taking the above problems into account, this dissertation first proposes a method to transform 3D neurons into 2D binary images, preserving their primary morphologies in the meanwhile. Currently, 3D neuro-morphological data are stored in the SWC format file with hundreds to tens of thousands of nodes [59]. For each node, its spatial information are mainly determined by the point location (i.e., x, y, z coordinates), radius r (i.e., indicating thickness of neuronal dendrite), and connection p (i.e., the connectivity with parent nodes). Therefore, unlike previous 3D deep learning problems which can directly handle the point sets, neuronal nodes and their composed tree-structures also need to be considered in the feature representation of 3D neuron morphology.

Here, denoting the 3D neuronal data as $\mathbf{x}_i \in \mathbb{R}^{n_i \times 5}$, which includes n_i points. Each node include the above spatial information with 5 dimensions, i.e., x, y, z, r, p. According to Fig. 3.2, our neuron projection method can transform each 3D neuron data into 2D images, preserving these 5 dimensional information in the meanwhile. According to Fig. 3.2, given a 3D neuron data $\mathbf{x}_i$, the location of its 3D points can be first plotted. Considering the initial neurons may not oriented properly, the principal component analysis (PCA) algorithm can be then employed to shift and rotate neurons to a normalized axis. For the point set of each neuron (i.e., a $n_i \times 3$ matrix), the PCA algorithm first compute their three dimensional mean value (i.e., a 1×3 vector) and coefficients (i.e., a 3×3 matrix). Then each point can be shifted and rotated by the mean value and coefficients respectively. This transformation insures that similar 3D neurons can be transformed into similar 2D images after the following neuronal projection, regardless of their initial orientation. Subsequently, all 3D points can be orthogonal projected into three angles of view, i.e., the x-y, x-z, and y-z plane, respectively. The three angles' projection can greatly preserve the spatial information of 3D neurons. Moreover, considering the projected images haven't reflected neuron's original tree-structure, two operations are introduced for each node in the projected images: 1) embedding each node's radius in the image, where all pixels within the node's radius are assigned as 1; 2) each node is connected to its corresponding parent node, where all pixels residing on the line segment connecting the two nodes are assigned to 1. After the above operations, three grayscale images can be generated from a 3D neuron data. The grayscale images can preserve the original neuronal spatial information and tree-topological structure as much as possible, and also transform the 3D neuron data into a usable modality for deep learning. In general, transforming 3D point sets into binary images can significantly improve the computational efficiency when training deep neural networks. More importantly, this method can preserve the structure of neuron morphologies by the three angles' orthogonal projection and child-parent points linking.

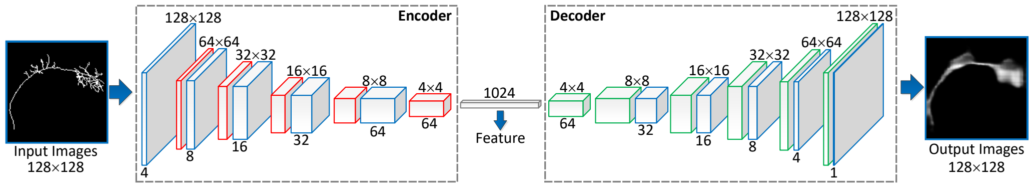


Figure 3.3: The architecture of our stacked convolutional auto-encoders, including convolutional encoder and decoder two parts.

3.2.2 Feature Learning using SCAEs

After 3D neuron projection, the generated 2D images can be employed to train deep neural network for the neuronal feature representation. In recent years, there are varieties of deep neural networks that designed for different datasets and tasks. In current neuron databases, there are no sufficient annotations to identify and classify each neuron, which only provides coarse brain regions, cell types, transmitters, etc. Thus only unsupervised deep neural network can be used in this case. Besides, neuron's tree-structures, e.g., dendrites and bifurcations, also need to be considered in the network. Here, we introduce the stacked convolutional autoencoders (SCAEs), which can explore the intrinsic structure of neurons in an unsupervised manner.

The general structure of SCAEs is illustrated in Fig. 3.3. From left to right, the network can be roughly divided into encoder and decoder two parts. The encoder subnetwork contains 6 convolutional layers and 5 maxpooling layers, which transforms a 128×128 grayscale image into a $64 \times 4 \times 4$ tensor which is further embedded into a 1024 dimensional feature vector via a fully connected layer. The decoder subnetwork is designed for recovering the grayscale image from the output of encoder network with 1 fully connected, 5 upsampling, 6 convolution and 1 deconvolution layers. In addition, batch normalization and ReLU activation are employed right after each convolution. Furthermore, tanh function is adopted to reconstruct the grayscale image for the deconvolution in the last layer of decoder subnetwork. Detailed configurations are illustrated in Table 3.1. The network is optimized through SGD algorithm using L_1

Table 3.1: Configurations of SCAEs network. The network can be divided into two subnetworks: encoder and decoder. ‘dim’ represents the dimension of the ouput of a layer. ‘batchnorm’ means batch normalization used in the layer. ‘relu/tanh’ indicates the activation function. ‘size’ is the kernel size used.

type	dim	size	stride
Input	1		
Encoder			
conv2d-batchnorm-relu	16	11x11	4
maxpooling		2x2	1
conv2d-batchnorm-relu	32	5x5	1
maxpooling		2x2	1
conv2d-batchnorm-relu	64	5x5	1
maxpooling		2x2	1
conv2d-batchnorm-relu	96	3x3	1
maxpooling		2x2	1
conv2d-batchnorm-relu	128	3x3	1
maxpooling		2x2	1
conv2d	128	3x3	1
Decoder			
conv2d-batchnorm-relu	128	3x3	1
upsampling		2x2	1
conv2d-batchnorm-relu	96	3x3	1
upsampling		2x2	1
conv2d-batchnorm-relu	64	3x3	1
upsampling		2x2	1
conv2d-batchnorm-relu	32	3x3	1
upsampling		2x2	1
conv2d-batchnorm-relu	16	3x3	1
deconv2d-tanh	1	12x12	4

loss function,

$$L = \|\mathbf{x} - \text{Decoder}(\text{Encoder}(\mathbf{x}, \theta_{\text{e}}), \theta_{\text{d}})\|_1, \tag{3.1}$$

where $\mathbf{x}$ is the input 2D neuron image, θ_{e}, θ_{d} is the parameter of encoder and decoder subnetwork respectively.

After training the SCAEs, the decoder part can be removed. Given a neuron data, its three binary images are sequentially set as the input in the trained SCAEs. Then their last encoder layer’s output can be combined together as the deep feature of that input neuron.

3.2.3 Feature Fusion

Considering the information loss from 3D neuron data to binary images, the unsupervised deep neural network usually cannot explore the fine-grained details in image data. The proposed SCAEs may also not work well in fully representing the 3D neuron morphology. Therefore, this subsection further proposes to fuse the deep features with the traditional hand-crafted features to pursue more powerful neuronal representations and accurate retrieval results. For the deep features, as they are usually noisy and redundant with thousands of dimensions, PCA algorithm is first employed to reduce the dimension and preserve the main components in deep features. For the hand-crafted features, quantitative measurements is computed in each 3D neuron based on the L-Measure toolbox [25], including global, branch, and bifurcation three levels of measurements [28].

With regards to fusing two features, there are mainly two levels of fusion in the field of image retrieval, i.e., feature-level and decision-level. For the feature-level fusion, the goal is to combine two or more feature vectors into a single one with more discriminative power than any of the input feature vectors. This fusion can be implemented before similarity searching. For the decision-level fusion, the goal is to weight the retrieval results from different features and fuse these results via techniques such as majority voting [61]. Despite the fact that the decision-level fusion has demonstrated excellent performance in many image retrieval tasks [62, 63], it may not be suitable in our case that the method was developed towards large neuron databases. Particularly, the decision-level fusion in our case, i.e., large-scale neuron morphological retrieval, needs to learn binary coding functions and search the whole database with different kind of features respectively, which are inefficient for large-scale retrieval. Therefore, this article directly combines the deep features with the hand-crafted features together before binary coding. Practical applications demonstrate that such feature-level fusion can achieve excellent performance in large-scale neuron retrieval.

3.3 Experiment

This section first evaluates the effectiveness of our proposed neuron feature representation method, and then provides discussions about its advantages and limitations.

3.3.1 Experimental Setting

In the experiment, NeuroMorpho database [1] is adopted to validate the performance, which is the largest collection of publicly accessible 3D neuronal reconstructions and associated metadata. Particularly, this experiment considers in total $58,414$ valid neurons for feature representation and evaluation (excluding neurons that cannot be read and measured by the L-measure tool, accessed by June 6, 2017). In the 3D neuron projection, the method projects and normalizes each neuron into three images with the size of 128×128. A weight decay of 10^{-4} and momentum of 0.9 are set in the SCAEs model. The whole neural networks are trained end-to-end in 100 epochs with an initial learning rate of 0.01. To train the SCAEs model, $30,000$ neurons are randomly selected, i.e., $90,000$ projected 2D images in total. For the hand-crafted features, L-measure toolbox is employed to extract 38 quantitative measurements, following the setting with previous articles [28, 8, 26, 9], which have achieved the best representational performance in several neuron analytical tasks. All experiments are carried out on a desktop with 1.6GHz processor of twelve cores and 128G RAM.

3.3.2 Evaluation of Neuron Feature Representation

To evaluate the performance of neuronal feature representation, the experiment employs the metric of neuron morphological retrieval, i.e., neuron's similarity searching in a neuron database. Particularly, this part compares the performance of four methods related to neuron feature representation and retrieval, including our learned deep features, the fused features, the above 38 dimensional hand-crafted features [28, 8, 26, 9] and the MIPS based binary coding features [28], which are abbreviated as Deep-fea,

Table 3.2: Comparing the average retrieval precision of four methods under different number of retrieved neurons.

	top-10	top-20	top-30	top-50
Deep-fea	0.6851	0.5534	0.4729	0.3710
MIPS-fea	0.8396	0.7605	0.6920	0.5982
Hand-fea	0.8586	0.7776	0.7239	0.6407
Fused-fea	**0.9130**	**0.8377**	**0.7943**	**0.7337**

Comb-fea, Hand-fea, and MIPS-fea respectively. Hand-fea and MIPS-fea are both the state-of-the-art features for neuronal retrieval. For the testing data, the Drosophila Melanogaster's projection neurons are selected as queries for which the brain region is the olfactory antennal lobe, and the cell types are principal cell and uniglomerular projection (233 such projection neurons in total, denoted as uPNs). This selection of query neurons is also consistent with previous articles [8, 26, 9, 28], since uPNs are the one kind of most fine-grain identified neurons in the NeuroMorpho database [1].

Table. 3.2 presents the average retrieval precision of the four comparative methods. The average retrieval precision is defined as the average percentage of same class neurons in all retrieved neurons after evaluating the 233 uPNs. For a query uPN, the top-10 retrieval precision denotes the percentage of uPNs in its 10 most similar neurons (except itself) after the feature comparison with the whole $58,414$ neurons based on the Euclidean distance. For the Deep-fea, we employ PCA to compress the original deep feature into 40 dimensions. The Comb-fea is the combination of Deep-fea and Hand-fea. The MIPS-fea generates 32 bits of binary codes as neuronal features. According to Table. 3.2, the Comb-fea can achieve the highest retrieval precision compared with other three methods. The Deep-fea also achieves reasonable retrieval precision, which validates that the deep learning based methods are effective for the feature representation of neuron morphology. The results are mainly benefited from our designed 3D neuron projection strategy, which can preserve the tree-topological

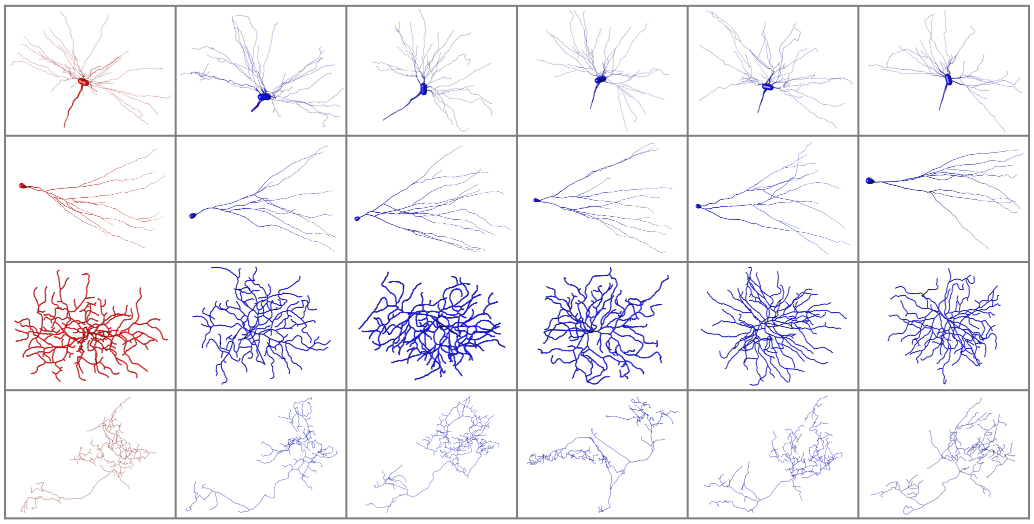


Figure 3.4: Four query neurons (red) and their corresponding top-5 similar neurons (blue) after searching in the NeuroMorpho [1] database, which illustrate the morphological similarity after using the proposed feature representation method.

structure of 3D neuron morphology in 2D binary images. The introduced SCAEs model has the ability to explore neuronal dendrites and bifurcations for more accurate representation.

More importantly, for the results of Comb-fea, we find that the retrieval performance has a significant improvement after combining the learned deep feature with the traditional hand-crafted feature. Based on the overview in Fig. 3.1, the left three grayscale images are projected from original 3D neuron, while the right three are the corresponding reconstructed images after the SCAEs decoder. It can be observed that the reconstructed images are able to preserve the overall structure from inputs, while most fine-grained details are lost. These results indicate that our learned deep features are more likely to explore and represent holistic structures in neuron morphologies. In contrast, the holistic structures in traditional hand-crafted features are represented by several primary measurements, e.g., neuron's total height, length, volume, etc, which can not well identify and discriminate neurons in large-scale databases. Therefore, the traditional hand-crafted features and the learned deep features are complementary

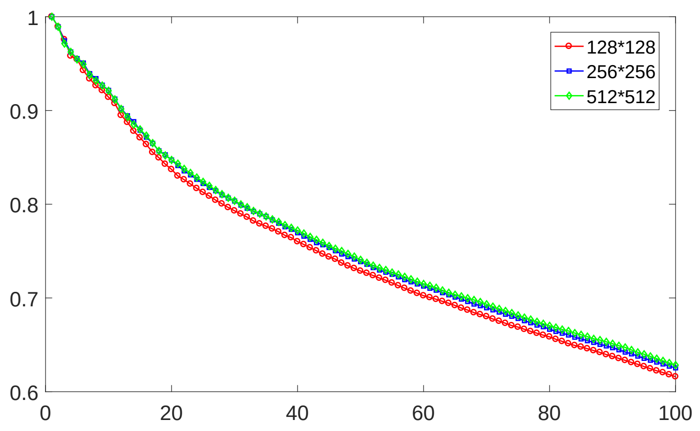


Figure 3.5: Average retrieval precision from top-1 to top-100 samples of uPNs: comparison of three sized images for SCAEs model training.

with each other in the representation of 3D neuron morphology.

In addition, we randomly select four neurons as queries and provide their top-5 most similar neurons after retrieval using the combined features. The neurons are displayed in Fig. 3.4 using the Vaa3D software [2], where the reds are queries and the blues are retrieved neurons. Fig. 3.4 validates that the proposed feature representation method can effectively find morphologically similar neurons in large-scale database. These results are useful in many neuron analytical tasks. For example, the similar neurons can be employed for neuron comparison to further analyze and explore the association of detailed arborization patterns and functional properties [8].

3.3.3 Discussion

We discuss the parameter settings in the proposed feature representation framework, as well as the benefits and limitations when applying the proposed methods for neuro-morphological analysis.

Parameter Settings: In the proposed framework, the SCAEs model is trained based on the projected 2D images with the size of 128×128. This setting is a trade-off between retrieval precision and computational efficiency. Fig. 3.5 records the average retrieval precision from top-1 to top-100 samples of uPNs, with the projected image size of 128×128, 256×256, 512×512 respectively. All three sized images are transformed into 1024 dimensional feature vectors in SCAEs models. According to Fig. 3.5, the retrieval performance haven't change too much, i.e., no more than 2% difference. This is mainly because of the properties of the SCAEs, which can mainly explore and represent holistic structures in neuron morphologies (discussed in Section 3.3.2). While the holistic structures in neurons do not have remarkably difference under different image resolutions, e.g., from 128×128 to 512×512. On the other side, their computational efficiency has a tremendous difference. For example, in the experiment, it only costs 3 hours to train the SCAEs model using $90,000$ neuron images with the size of 128×128. While the training time is 2 days using the 512×512 sized images. Therefore, we projected 2D images with the size of 128×128 which can achieve the state-of-the-art performance under sustainable time complexity.

Advantages: The proposed framework achieve large-scale neuron morphological retrieval with superior accuracy and efficiency, demonstrating excellent performance in assisting neuron exploration and analysis. These results mainly benefited from the proposed neuron feature representation method. When neuron databases are large, the traditional hand-crafted features cannot fully represent and differentiate each neuron. Thus we design a novel feature representation method using deep neural networks, which can transform 3D neuron data into 2D images and automatically learn features end-to-end. The learned deep features have different aspects of representation compared with the hand-crafted features (e.g., holistic structures compared with branch/bifurcation topologies). Thus their combination will accordingly more representative for the 3D neuron morphology.

Limitations: There are also some limitations in the proposed neuron retrieval framework. One critical limitation is the information loss when transforming 3D neuron data into 2D images. This limitation mainly reflects in two situations: 1) the 2D images cannot preserve the fine-grained spatial structures in neurons, especially

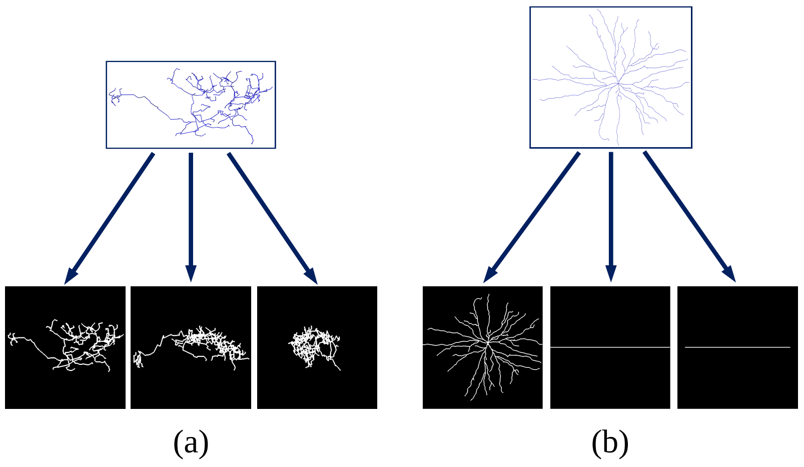


Figure 3.6: Limitations in our neuron projection strategy, (a) the 2D image cannot preserve the fine-grained spatial structures in neurons; (b) some neurons in the NeuroMorpho [1] are only two dimensions.

for the neuron with complicated morphologies, e.g., the right dendrites in Fig. 3.6(a); 2) some neurons provided by the NeuroMorpho database [1] are only two dimensions, i.e., the third dimension is a fixed constant. As shown in Fig. 3.6(b), in such case, the proposed method can only get straight lines in the second and third projected image. These two situations may influence the retrieval results for some specific neurons.

Another limitation is the introduced deep neural networks. The unsupervised SCAEs model can only explore the holistic information among neuron morphologies, where the most fine-grained information are lost. Despite the deep features are the good complementary with hand-crafted features, the neuronal feature representation framework are not fully automatic, which still need to extract L-measure features and combine two kinds of feature together. Additionally, due to the fact that the deep learning framework is unsupervised, the deep features may not always accurate since the existed semantic gap between the learned features and the biological properties among massive neurons.

3.4 Summary

This chapter attempts to investigate deep learning techniques in tackling the feature representation of 3D neuronal morphology. A generalized framework is proposed based on the neuronal projection, unsupervised deep neural networks and feature fusion, which achieves superior performance compared with the state-of-the-art. Nevertheless, there are several aspects that can be explored to further improve the performance. For example, in 3D neuron projection, the three grayscale 2D images are related to each other, since they are reflections of a common neuron in different view angles. These relations should be considered and utilized in the deep neural network. Additionally, our current framework cannot achieve the fully automatic feature representation. In the future, we will study how to embed the traditional hand-crafted features in a deep neural network to compute neuron features end-to-end.

CHAPTER 4: LARGE-SCALE NEURON MINING

4.1 Motivation

After acquiring morphological features from 3D neuron data, this dissertation aims to explore efficient neuron mining in large-scale databases. Regarding neuron mining and exploration, multiple challenges need to be investigated, e.g., efficiently finding neurons sharing similar morphologies, identifying neuron types, correlating neuron morphologies with properties. To achieve the above challenges, one basic solution is neuronal retrieval, i.e., given a query neuron, finding neurons sharing similar morphologies in existed neuron databases, where the neuron types, morphologies and properties can be accordingly analyzed. This section presents solutions for neuron retrieval, particularly in large-scale and continuously expanding neuron databases.

Morphology-based neuron retrieval is made possible because of the recent rapid advancements in neuron tracing and neuron anatomy techniques [64, 65, 66, 67]. In the field of neuron morphological retrieval and analysis, Wan et al. [8] designed *BlastNeuron* for automated comparison, retrieval and clustering of 3D neuron morphologies. In the retrieval stage, *BlastNeuron* searches for similar neurons via the normalization of rank scores in terms of the closeness of feature vectors. Despite its high accuracy, this method could be inefficient when tackling large-scale neuron databases. Costa et al. [26] presented *NBLAST*, a sensitive and rapid algorithm, for measuring pairwise neuronal similarity. They developed an online neuron similarity search program which can query single neurons or fragments against large databases, based on the measure of short segments in neurons. Subsequently, Mesbah et al. [27] proposed a data-driven hashing scheme, i.e., hashing forest, to search among large neuron databases. By establishing multiple unsupervised random forests, 128 or more binary bits are generated to represent morphological features. Hash forest algorithm has achieved efficient and accurate results in neuron retrieval and analysis [29, 9]. Nonetheless, it usually needs a large number of bits (e.g., larger than 128), while its efficiency can be further improved with shorter binary codes. More importantly, the encoding process relies on the embedding of the Euclidean distance, which may not be a suitable similarity measure for neuron retrieval issue, as features of neuron data usually lay in complex feature spaces that may not be linearly separable. Therefore, advanced binary coding and hashing methods to solve these challenges are important for efficient and accurate retrieval.

As described in [27], binary coding and hashing techniques have achieved great successes in efficient retrieval of large-scale databases, with many methods proposed in recent years, including, but not limited to, Spectral Hashing (SH) [41], Anchor Graph Hashing (AGH) [42], Iterative Quantization (ITQ) [35], etc [68, 69, 70, 71]. However, they may not be directly applicable to the neuron retrieval problem, as the features of 3D neuron morphological data are dramatically different from 2D natural images, where different neuronal features usually reflect different levels of similarity measure. For example, the tree-like structure imposes a challenge to differentiate the neuron types, since treating all features with equal importance may lead to inaccurate retrieval results. In addition, although supervised binary coding and hashing methods have already been investigated in medical image analysis [40, 72, 73], it is preferred to employ unsupervised methods for neuron retrieval, since there are no sufficient annotations to differentiate and classify all neurons.

Although neuron morphology and binary coding are both well-studied in recent years, how to combine them for neuron retrieval remains a hard problem. Applying binary coding in solving the large-scale neuro-morphological retrieval is quite different in comparison with the large-scale retrieval of natural images. Specifically, there are three challenges when introducing binary coding in neuron morphological retrieval:

1. Due to the employment of hand-crafted features (i.e., the quantitative measurements), the feature vectors of each neuron are much shorter (30 to 100 dimensions) compared with traditional 2D images' feature vectors (500 to 10000 or more dimensions). Therefore, only much shorter bits of binary codes can guarantee the retrieval efficiency. Employing much shorter binary codes for the representation of large-scale neuron data is a great challenge;

2. For the 38 dimensional hand-crafted features we employed in Chapter 3, despite the limited length of neuron feature vectors, each type of feature has their specific meaning, e.g., branch number reflects the connection of neuron cell, bipolar neurons have two branches while multipolar neuron have three or more branches connected with other neurons. Currently, most image retrieval methods are either addressing the single feature's binary coding or fuse multiple features in different retrieval stages [74, 75, 62, 76, 77]. However, in the neuron retrieval problem, each single feature is too short to obtain reliable retrieval results, and fusing multiple features is usually time-consuming. Thus, the specific biological indication and the computational complexity in neuronal feature representation need to be considered.

3. As each hand-crafted features are extracted based on the tree-like structure, this limitation of feature extraction may cause a tough question, in which the tree-like structure will lead to similar features extracted from different types of neurons, e.g., some unrelated neurons express approximate feature vectors. How to differentiate these approximate feature vectors in non-linear space is a hard problem.

This chapter aims to investigate binary coding algorithms that can accurately and efficiently perform large-scale neuron retrieval, which is a critical step for neuron identification and analysis. Novel binary coding methods are designed for the massive and continuously expanding neuro-morphological datasets. Particularly, to differentiate the linearly inseparable neurons in large-scale datasets, we first develop a binary coding method based on the maximum inner produce search (MIPS). Unlike prior methods that learn hashing functions to embed Hamming distances or Euclidean distances, our method obtains effective coding functions for maximum inner product, which has the flexibility to differentiate complex features that are linearly inseparable in the original feature space. In fact, this strategy is particularly suitable for the neuron morphology data, which is usually non-convex and non-smooth. Moreover, for the continuously expanding neuron datasets, we extend binary coding with online updating schemes, which only considers the newly added neurons and update coding functions on-the-fly, without accessing the whole neuron databases. The proposed methods are validated in the neuron retrieval problem with the largest neuro-morphological database, and they outperform multiple related binary coding or hashing methods. In addition, according to the neuron information provided by NeuroMorpho [1], the proposed methods can retrieve similar neurons in terms of morphology, cell type and brain region.

4.2 Maximum Inner Product Search for Large-scale Neuron Retrieval

This section presents the theoretical and technical details of MIPS based binary coding for the large-scale neuron retrieval, including the MIPS notation, feature hierarchy and asymmetric optimization.

4.2.1 Overview

In the framework of MIPS based binary coding, we use morphological measurements as features to represent each neuron data. Although directly measuring the similarity between morphological features offers an accurate solution, the computational efficiency is an issue, especially when searching in a large-scale database of tens of thousands of neurons. Therefore, we focus on learning coding functions to trans-

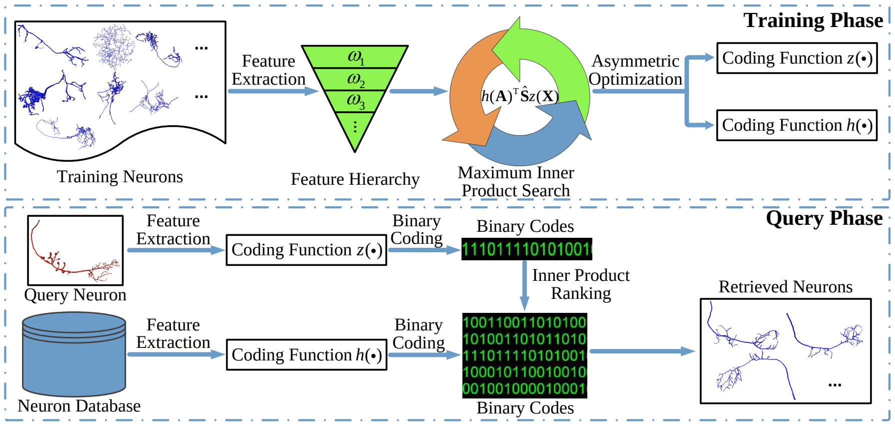


Figure 4.1: Overview of the proposed neuron morphological retrieval framework.

form morphological features into binary codes. Fig. 4.1 shows the overview of the proposed framework. In the training phrase, after feature extraction, we group different features into several hierarchies to compute the similarity matrix. Then, binary coding functions are learnt which can maximize inner product between two training data sets. Particularly, for optimization convenience, we jointly maximize two asymmetric coding functions $h(\cdot)$ and $z(\cdot)$ for the neuron database and the query neuron respectively. With these learnt coding functions, in the query phase, the features of query neuron and all neurons in the database are compressed into short binary codes. Then, their inner products can be sequentially calculated and ranked in descending order. By selecting neurons in the database with top-K largest inner product (indicating top similar neurons among the whole database), the characteristics of query neuron can be identified based on the retrieved neurons.

4.2.2 Methodology

Background of MIPS: Let's denote $\mathbf{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_i, \ldots, \mathbf{a}_n\} \subset \mathbb{R}^{n\times d}$ as the training neuron data set, which include n neurons, and each neuron has d dimensional features. From each neuron M types of morphological features are extracted, denoted as $\mathbf{a}_i = [\mathbf{a}_i^{(1)}, \ldots, \mathbf{a}_i^{(j)}, \ldots \mathbf{a}_i^{(M)}] \in \mathbb{R}^{1\times d}$, where $d = \sum_{j=1}^{M} d_j$. Assume the query neuron is $\mathbf{q} \in \mathbb{R}^{1\times d}$, the MIPS problem can be defined as:

$$\mathbf{p} = \arg\max_{\mathbf{a}_i \in \mathbf{A}} \mathbf{a}_i \mathbf{q}^{\mathrm{T}} \tag{4.1}$$

which finds the largest inner product between $\mathbf{q}$ and each element in $\mathbf{A}$. As demonstrated in [40], the Hamming distance and the code inner product have a one-to-one correspondence. To accelerate computation and save storage, it is reasonable to employ binary coding method in tackling MIPS problem. A coding function h is learned to map the original feature vectors to bits of binary codes. Thus, problem (4.1) is reformulated as:

$$\mathbf{p} = \arg\max_{\mathbf{a}_i \in \mathbf{A}} h(\mathbf{a}_i) h(\mathbf{q})^{\mathrm{T}} \tag{4.2}$$

Compared with common binary coding methods based on Hamming distance minimization, h is likely to be a non-linear function through MIPS, which is more suitable for the neuron retrieval database that is linearly inseparable.

In [78], Shrivastava and Li proposed the Asymmetric Locality Sensitive Hashing (ALSH) method, proving that it is impossible to inherit the high collision probability guarantee of MIPS problem under current LSH framework. In fact, by adopting two different binary coding functions $h(\cdot)$ and $z(\cdot)$ to compute the inner product of the database and query, the MIPS can be converted as the standard L_2 nearest neighbor search problem [78, 69]. Accordingly, to generate more effective binary codes, we also adopt two coding functions for the MIPS problem:

$$\mathbf{p} = \arg\max_{\mathbf{a}_i \in \mathbf{A}} h(\mathbf{a}_i) z(\mathbf{q})^{\mathrm{T}} \tag{4.3}$$

The remaining issue is how to learn two coding functions $h(\cdot)$ and $z(\cdot)$, which can generate effective binary codes to make query neurons finding their corresponding similar neurons in database.

Binary Coding for Neuron Retrieval: For the training neuron data set $\mathbf{A}$, we assemble another neuron set $\mathbf{X} \subset \mathbb{R}^{m\times d}$ which is randomly sampled from the database. Denoting matrix $\mathbf{S} \in \mathbb{R}^{n\times m}$ which reflects the similarity of each neuron between $\mathbf{A}$ and $\mathbf{X}$, the idea of MIPS based binary coding is to learn coding functions which can make the inner product of $\mathbf{A}$ and $\mathbf{X}$ to approximate with $\mathbf{S}$ in the form of binary codes:

$$\min_{h,z} \left\| h(\mathbf{A})z(\mathbf{X})^{\mathrm{T}} - \hat{\mathbf{S}} \right\|^2 \tag{4.4}$$

where $\hat{\mathbf{S}}$ is the binarization form of $\mathbf{S}$ by its mean value. Instead of directly solving this challenging problem, we discard its quadratic part after expansion and only focus on the correlation between similarity matrix $\hat{\mathbf{S}}$ and $h(\mathbf{A})z(\mathbf{X})^{\mathrm{T}}$. Since the discarded quadratic part is the regularization term, which does not include any ground truth information. Thereby, eq. (4.4) can be re-defined as:

$$\max_{h,z} \operatorname{trace}(h(\mathbf{A})^{\mathrm{T}}\hat{\mathbf{S}}z(\mathbf{X})) \tag{4.5}$$

In practice, we find that omitting the quadratic part does not affect the binary coding performance. It also makes the problem easier to optimize, because the similarity term is more efficient to solve for the non-linear differentiation of neuron morphologies. Subsequently, we define two binary coding matrices $\mathbf{W}$, $\mathbf{R} \in \mathbb{R}^{d\times r}$ to substitute the coding functions, where $h(\mathbf{A}) = \operatorname{sgn}(\mathbf{AW})$, $z(\mathbf{X}) = \operatorname{sgn}(\mathbf{XR})$, and r is the bits of binary codes for each neuron. Generally, $\mathbf{W}$ and $\mathbf{R}$ are initialized by Principal Component Analysis (PCA) projections or random generation. Then, we obtain eq. (4.5) in a new form:

$$\max_{\mathbf{W},\mathbf{R}} \operatorname{trace}(\operatorname{sgn}(\mathbf{AW})^{\mathrm{T}}\hat{\mathbf{S}}\operatorname{sgn}(\mathbf{XR})) \tag{4.6}$$

In this objective function, we need to optimize $\mathbf{W}$ and $\mathbf{R}$ from the training neuronal sets $\mathbf{A}$, $\mathbf{X}$ and their similarity matrix $\mathbf{S}$. Generally, $\mathbf{S}$ is computed by inner product, $\mathbf{S} = \mathbf{A}\mathbf{X}^{\mathrm{T}}$. However, in this work, the extracted features especially the hand-crafted features captured through the quantitative measurements usually express different similarity levels in the representation of neuron cells. Simply aligning these features together to compute similarity matrix may generate ineffective binary codes. Therefore, we first consider feature diversities to compute more suitable similarity matrix, then optimize $\mathbf{W}$ and $\mathbf{R}$ for the eq. (4.6).

Feature Hierarchy: For the single type of neuron feature, i.e., jth feature, its similarity matrix can be obtained by the corresponding inner product of $\mathbf{A}^{(j)}$ and $\mathbf{X}^{(j)}$:

$$\mathbf{S}^{(j)} = \mathbf{A}^{(j)}(\mathbf{X}^{(j)})^{\mathrm{T}} \tag{4.7}$$

Many articles [75, 79] treat the above $\mathbf{S}^{(j)}$ as feature kernels and fuse multiple kernels together with different weights to compute the similarity matrix:

$$\mathbf{S} = \sum_{j=1}^{M} \mu_j \mathbf{S}^{(j)} \tag{4.8}$$

where μ_j is the similarity weight of jth feature. In most case, there are either few types of features or few numbers of training data, which make the computational complexity of $\mathbf{S}$ acceptable. However, in large-scale neuron retrieval, many features' similarity matrices need to be calculated (38 features in this paper), and usually thousands of neurons in database should be set as the training data to ensure the retrieval precision. The computational complexity of the similarity matrix is an issue in neuron retrieval.

Since neuronal features are extracted from the tree-like structure, neuron retrieval can also benefit by being treated as the similarity search of the tree-like structures.

Despite the fact that each type of feature has its specific meaning, they can be grouped into hierarchies according to their different levels of representation. The similarity levels can also be computed hierarchically, from the measures of soma (tree's root) and branches (tree's vertices and edges) at the global level (i.e., height, width of the whole neuron cell), features measured in the branches directly connect with soma at first level branches, etc. Assuming there are L hierarchies for all types of features, the similarity matrix can be re-calculated as:

$$\mathbf{S} = \sum_{l=1}^{L} \omega_l [\mathbf{A}^{(l)} (\mathbf{Z}^{(l)})^{\mathrm{T}}] \tag{4.9}$$

where $A^{(l)} = [A^{(j_1)}, \dots, A^{(j_l)}]$ means that features $j_1, \dots, j_l$ are grouped together and they all belong to the lth hierarchy. Consequently, the computation efficiency will have great improvement via the feature hierarchy process ($L \ll M$). More importantly, each hierarchical weight ω_l is much easier to acquire compared with each feature's weight μ_j. In practice, we group the 38 quantitative measurements (i.e., hand-crafted features) into three hierarchies based on their measured locations in the neurons' tree-topological structure. Hierarchical weights are determined by the neuronal tree-like structure, and will be discussed in the experiment part. Here, we provide the group details of the quantitative measurements, including the statistic metric adopted in the L-measure toolbox [25]:

1. Global Level: Length (sum), Height (sum), Width (sum), Depth (sum), Volume (sum), N_tips (sum), Nbifs (sum), Surface (sum), N_branch (sum), Diameter (sum), Soma_Surface (sum), N_stems (sum), Contraction (sum), Fragmentation (sum), Pk_classic (sum);

2. Branch Level: Euc_Distance (sum), Helix (average and max), Fractal_Dim (average and max), Branch_order (max), Branch_path_length (average and max), Path_Distance (sum),;

3. Bifurcation: Bif_ampl_local (average and max), Bif_ampl_remote (average and max), Bif_tilt_local (average and max), Bif_tilt_remote (average and max), Bif_torque_local (average and max), Bif_torque_remote (average and max), Terminal_degree (average), Partition_asymmetry (average).

Asymmetric Optimization: After compute the similarity matrix, we design an asymmetric strategy to solve the highly non-convex optimization problem of eq. (4.6). As both $\mathbf{W}$ and $\mathbf{R}$ are constrained by the sign function, it is hard to simultaneously optimize them together. Instead, we first assume that the right part of eq. (4.6) is fixed as a constant matrix $\mathbf{Z} = \mathrm{sgn}(\mathbf{XR})$, and then we consider the following sub-problem with variable $\mathbf{W}$:

$$\max_{\mathbf{W}} \mathrm{trace}(\mathrm{sgn}(\mathbf{AW})^{\mathrm{T}}\hat{\mathbf{S}}\mathbf{Z}) \tag{4.10}$$

In the same way, fix the left part $\mathbf{H} = \mathrm{sgn}(\mathbf{AW})$, we can obtain the sub-problem with variable $\mathbf{R}$:

$$\max_{\mathbf{R}} \mathrm{trace}(\mathbf{H}^{\mathrm{T}}\hat{\mathbf{S}}\mathrm{sgn}(\mathbf{XR})) \tag{4.11}$$

Compared with eq. (4.6), only one sign function and coding matrix are included in each sub-problem. Based on this asymmetric design, if we can solve the two sub-problems, then optimal $\mathbf{W}$ and $\mathbf{R}$ for the whole problem can also be obtained by several alternative iterations between (4.10) and (4.11).

For the sub-problem (4.10), despite that only one sign function remains, it is still a discrete optimization issue. To solve this, we introduce an auxiliary variable $\mathbf{B} \in \{-1, 1\}^{n\times r}$ as the binary codes of $\mathbf{A}$ to replace the discrete part $\mathrm{sgn}(\mathbf{AW})$, and the sub-problem (4.10) can be separated into two terms:

$$\max_{\mathbf{B},\mathbf{W}} \mathrm{trace}\left[(\mathbf{B}^{\mathrm{T}}\hat{\mathbf{S}}\mathbf{Z}) - \lambda\|\mathbf{B} - \mathbf{AW}\|^2\right] \tag{4.12}$$

The first term maximizes inner product via the learned binary codes, and the second term ensures that $\mathbf{AW}$ can approximate with the target binary codes $\mathbf{B}$. Denoting λ as a trade-off parameter between these two terms. Subsequently, $\mathbf{W}$ can be optimized by several alternative iterations with $\mathbf{B}$:

$$\begin{cases} \mathbf{B} = \operatorname{sgn}(\hat{\mathbf{S}}\mathbf{Z} + 2\lambda\mathbf{AW}) \\ \mathbf{W} = \mathbf{A}^{\dagger}\mathbf{B} \end{cases} \tag{4.13}$$

where $\mathbf{A}^{\dagger}$ is the pseudo-inverse of $\mathbf{A}$. Optimal $\mathbf{W}$ of this sub-problem will be acquired until coverage or reach maximum t iterations.

After solving the (4.10), denoting $\mathbf{D} \in \{-1, 1\}^{m\times r}$ as the auxiliary variable for $\operatorname{sgn}(\mathbf{XR})$, then optimal $\mathbf{R}$ for sub-problem (4.11) can also be acquired in the same way:

$$\begin{cases} \mathbf{D} = \operatorname{sgn}(\hat{\mathbf{S}}^{\mathrm{T}}\mathbf{H} + 2\lambda\mathbf{XR}) \\ \mathbf{R} = \mathbf{X}^{\dagger}\mathbf{D} \end{cases} \tag{4.14}$$

As these $\mathbf{W}$ and $\mathbf{R}$ are the local optimal results of two sub-problems, we denote such alternative iterations between coding matrices and auxiliary variables as the inner loop. To obtain the optimal coding matrices for the objective function (4.6), several outer alterative iterations between (4.10) and (4.11) are still needed until coverage or reach maximum iterations.

During the optimization, each variable is obtained in a closed form (e.g., $\mathbf{W}$, $\mathbf{R}$, $\mathbf{B}$, $\mathbf{D}$), to effectively learn the coding functions even with short binary codes. In addition, the MIPS based objective functions and asymmetric optimization provide a convergent solution to differentiate neuron morphologies in non-linear space.

4.2.3 Implementation Details

Given the training data $\mathbf{A}$ and $\mathbf{X}$, our MIPS based binary coding for neuron mor-

Algorithm 1 Binary Coding Based on Maximum Inner Product Search.

```
Require: Training data $\mathbf{A}$ and $\mathbf{X}$;
Ensure: Binary coding matrices $\mathbf{W}$ and $\mathbf{R}$
1: Extract $\mathbf{M}$ types of morphological features for each neuron in the training data;
2: Group features into $L$ hierarchies;
3: Compute the similarity matrix through eq. (4.9);
4: Initialize binary coding matrix $\mathbf{W}$ and $\mathbf{R}$ by PCA projections;
5: while unconverged or less than the maximum $T$ iterations do
6:     Solving sub-problem (4.10): compute $\mathbf{W}$ by the inner loop of (4.13), where
       $\mathbf{Z} = \text{sgn}(\mathbf{XR})$;
7:     Solving sub-problem (4.11): compute $\mathbf{R}$ by the inner loop of (4.14), where
       $\mathbf{H} = \text{sgn}(\mathbf{AW})$;
8: end while
```

phological retrieval can be outlined in Algorithm 1. With acquired coding matrix $\mathbf{W}$, the morphological features of every neuron in the database $\mathbf{a}_i \in \mathbb{R}^{1\times d}$ can be mapped to binary codes via the coding function $h(\mathbf{a}_i) = \text{sgn}(\mathbf{a}_i\mathbf{W})$. In the same way, with coding matrix $\mathbf{R}$, the binary code of query neuron is calculated by $z(\mathbf{q}) = \text{sgn}(\mathbf{qR})$. Then, the similarity search problem between query neuron and the neuron database is transformed as the inner product ranking of their binary codes. For the query neuron, the similar neurons are defined as the neurons with top-K largest inner product, and these similar neurons can further be used to interpret biomedical meanings of the query neuron.

4.3 Online Binary Coding for Continuously Expanding Databases

Although binary coding and hashing methods have been widely investigated for large-scale image analysis [27, 72], how to efficiently tackle the frequently updated databases is still an unresolved problem. This is particularly important for neuron analysis, since an increasing number of neuron cells are reconstructed and added to the morphological database in a streaming manner, benefited from the well-designed neuron tracing software [2, 64]. For example, the NeuroMorpho database [1] is continuous increasing, and usually releases $5,000$ to $10,000$ reconstructed neurons in each update. If we re-train binary coding models every time from scratch, using both

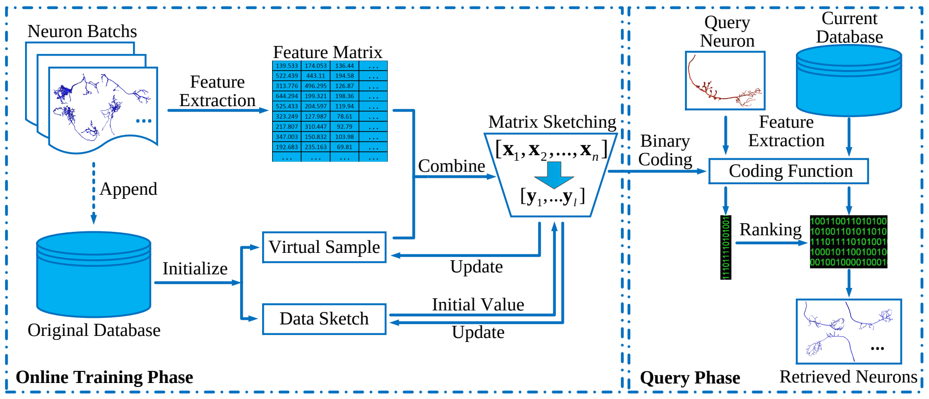


Figure 4.2: Overview of the proposed binary coding with online updating framework for the large-scale neuron retrieval.

the original and the newly added neuron data, it will be very time-consuming and adversely affects the efficiency of exploration.

To alleviate these problems, we design an online binary coding framework, which can achieve accurate and efficient morphological retrieval by accommodating the continuously updated neuron databases on-the-fly.

4.3.1 Overview

Fig. 4.2 shows our framework for the exploration of continuously updated neuron databases. Firstly, effective features are needed to represent each neuron. As the 3D neuron data is dramatically different from 2D natural images, we select several morphological quantitative measurements as features based on their tree-like structures, e.g., neuron's total height, number of branches and soma surface [24]. Then, we apply the matrix sketching method on the extracted features from the existing neuron database, generating the data sketch and virtual sample as initial values for the online updating afterwards. Given a new neuron batch during the updating phase, we combine it with the aforementioned virtual sample together. This combination can overcome the mean-varying problem in matrix sketching with continuously updated neuron data. The sketching result can be subsequently used for binary coding to update the current coding function. The new data sketch and virtual sample are also stored for the next update. During the query phase, the features of query neuron and all neurons in current database are compressed into short binary codes based on updated coding functions. The neuron retrieval problem is transformed into the hamming distance ranking between binary codes of query neuron and the other neurons. Specifically, the retrieved similar neurons are these with top-K minimum hamming distances, and they can be used to explore the biomedical meanings of the query neuron.

4.3.2 Binary Coding with Matrix Sketching

The goal of binary coding is to compress feature vectors into short binary codes, and also keep diversities and similarities among original data. Denoting a training neuron database $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_n\} \subset \mathbb{R}^{n \times d}$, which includes n neurons, and each neuron has d dimension of features. We aim to learn a coding function $\mathbf{W} \in \mathbb{R}^{d \times r}$ that every normalized neuron feature in $\mathbf{X}$ can be transformed into r bits of binary codes, i.e., $h(\mathbf{x}_i) = \text{sgn}((\mathbf{x}_i - \overline{\mathbf{X}})\mathbf{W})$, where $\overline{\mathbf{X}}$ is the mean value of $\mathbf{X}$. Note that feature normalization with zero mean is a crucial step in binary coding, especially for neuron data, because each dimension has its physical meaning. To learn effective binary codes, usually two requirements should be satisfied: (1) binary bits are uncorrelated and their variances are maximal; (2) numbers of 0 and 1 are roughly equal in the learned binary codes of $\mathbf{X}$. Same as [80], the requirements are satisfied by maximizing the following objective function:

$$J(\mathbf{W}) = \frac{1}{n}\text{trace}(\mathbf{W}^T(\mathbf{X} - \overline{\mathbf{X}})^T(\mathbf{X} - \overline{\mathbf{X}})\mathbf{W}), \quad \text{s.t. } \mathbf{W}^T\mathbf{W} = \mathbf{I}_{r \times r} \tag{4.15}$$

Instead of directly optimizing the above objective function, we apply the matrix sketching technique on the training database to learn coding function from data sketch. Sketching is a data compression technique which can significantly reduce the data size, without losing much information. Specifically, for the neuron database

$\mathbf{X}$, we denote its matrix sketch as $\mathbf{Y} \in \mathbb{R}^{l \times d}$, which has the property $\mathbf{Y}^T\mathbf{Y} \approx (\mathbf{X} - \overline{\mathbf{X}})^T(\mathbf{X} - \overline{\mathbf{X}})$. We employ the *Frequent-directions* (FD) algorithm [81] to compute $\mathbf{Y}$, as this algorithm can effectively keep the property of matrix sketch. More importantly, FD is a streaming algorithm which can sequentially process the training data. In other words, when a new data batch comes, FD algorithm will update the current sketch. We show the advantage of such streaming strategy for the online updated neuron data in next subsection.

Given the data matrix $\mathbf{X}$, FD algorithm can obtain its sketch $\mathbf{Y}$ with much smaller data size ($l \ll n$). Then, the objective function of binary coding can be re-written as:

$$J(\mathbf{W}) \approx \frac{1}{n}\text{trace}(\mathbf{W}^T\mathbf{Y}^T\mathbf{Y}\mathbf{W}) \tag{4.16}$$

This objective function is exactly the same as that of Principle Component Analysis (PCA). The optimal coding function $\mathbf{W}$ can be obtained by taking the top r eigenvectors of the data covariance matrix $\mathbf{Y}^T\mathbf{Y}$ [35]. In addition, to alleviate the unbalance of different dimensions in neuron data, we adopt orthogonal rotation for the above acquired coding function, where $\mathbf{W} = \mathbf{W}\mathbf{R}$. However, since the sketch $\mathbf{Y}$ is much smaller than the whole training data $\mathbf{X}$, we cannot learn the optimized $\mathbf{R}$ as ITQ [35], which relies on all training data. Instead, we generate a random orthogonal matrix as $\mathbf{R}$, which achieves promising accuracy and efficiency in our experiments.

4.3.3 Online Coding Function Updating

When new neuron batches are added to the database, we need to update the coding function accordingly to guarantee the retrieval precision. Re-training coding function from scratch is very time-consuming, and sometimes infeasible when the existing neuron database is too large to load into the memory. Considering that the FD algorithm can compute data sketch in a streaming manner, an intuitive solution is to set the previous database sketch as the initial value, and then employ the FD algorithm to sketch the newly added data. The coding function can be also updated with the new sketch. Unfortunately, this attempt is impractical because of the feature normalization requirement in binary coding. As the neuron database are continuously changing, its mean value for normalization is also changed. How to overcome this mean-varying problem is a critical step in online coding function updating.

Assume $\mathbf{B}_k$ is the newly added batch at round k which include m_k neurons, and the current database is denoted as $\mathbf{X}_k = \{\mathbf{B}_0, \mathbf{B}_1, \dots, \mathbf{B}_k\}$. Then, the mean value of $\mathbf{X}_k$ can be computed as:

$$\overline{\mathbf{X}}_k = \frac{\overline{\mathbf{X}}_{k-1} \cdot n_{k-1} + \overline{\mathbf{B}}_k \cdot m_k}{n_k} \tag{4.17}$$

where $\overline{\mathbf{B}}_k$ is the mean value of $\mathbf{B}_k$ and $n_k = \sum_{i=0}^{k} m_i$. Obviously, the mean value of neuron database is changed in each update. To solve this problem, we introduce a virtual sample, which considers the difference of mean value between the previous database and the current batch [82]. Then we combine it with the current batch:

$$\widehat{\mathbf{B}}_k = [\mathbf{B}_k - \overline{\mathbf{B}}_k, \sqrt{\frac{n_{k-1}m_k}{n_k}}(\overline{\mathbf{B}}_k - \overline{\mathbf{X}}_{k-1})] \tag{4.18}$$

At round k, we have a new set of data $\widehat{\mathbf{X}}_k = \{\mathbf{B}_0 - \overline{\mathbf{B}}_0, \widehat{\mathbf{B}}_1, \dots, \widehat{\mathbf{B}}_k\}$. According to [82], in each update, $\widehat{\mathbf{X}}_k$ takes the shift of mean into account and corrects such shift by the virtual sample. More importantly, combining with Eq. (4.17), we find that $\widehat{\mathbf{X}}_k^T \widehat{\mathbf{X}}_k = (\mathbf{X}_k - \overline{\mathbf{X}}_k)^T(\mathbf{X}_k - \overline{\mathbf{X}}_k)$. This property indicates that data sketch of $\widehat{\mathbf{X}}_k$ and $\mathbf{X}_k - \overline{\mathbf{X}}_k$ is the same. Since $\widehat{\mathbf{X}}_k$ has no mean-varying problem, we can employ the aforementioned FD algorithm to sketch the continuously updated neuron data $\widehat{\mathbf{B}}_k$, and the coding function can be also updated on-the-fly via the matrix sketching based binary coding.

Algorithm 2 summarizes the workflow of our online binary coding for neuron re-

Algorithm 2 Binary Coding with Online Updating for Neuron Retrieval.

```
Require: Original neuron database $\mathbf{B}_0$, streaming neuron batches $\{\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_k\}$.
Ensure: Online updated coding functions $\{\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_k\}$.
1: Sketch $\mathbf{B}_0 - \overline{\mathbf{B}}_0$ into $\mathbf{Y}_0$;
2: Initialize $n_0 = m_0$, $\overline{\mathbf{X}}_0 = \overline{\mathbf{B}}_0$;
3: for $i = 1 \to k$ do
4:     Sketch $\widehat{\mathbf{B}}_i = [\mathbf{B}_i - \overline{\mathbf{B}}_i, \sqrt{\frac{n_{i-1}m_i}{n_i}}(\overline{\mathbf{B}}_i - \overline{\mathbf{X}}_{i-1})]$ into $\mathbf{Y}_i$, initialize by $\mathbf{Y}_{i-1}$;
5:     Update coding function $\mathbf{W}_i$: $\arg\max_{\mathbf{W}_i} \frac{1}{m_i+1}\text{trace}(\mathbf{W}_i{}^T\mathbf{Y}_i{}^T\mathbf{Y}_i\mathbf{W}_i)$;
6:     Update data size: $n_i = n_{i-1} + m_i$;
7:     Update mean value: $\overline{\mathbf{X}}_i = \frac{\overline{\mathbf{X}}_{i-1}\cdot n_{i-1} + \overline{\mathbf{B}}_i \cdot m_i}{n_i}$;
8: end for
```

trieval. Starting from the initial data sketch and the mean value, it can continuously update the coding function when new neuron batch comes. Without accessing the entire neuron database, we only keep the data sketch $\mathbf{Y}_i$, data size n_i and mean value $\overline{\mathbf{X}}_i$ in each update. Therefore, the coding function can be efficiently updated and applied to this neuron retrieval problem.

4.4 Experiment

This section validates the effectiveness of our introduced two binary coding methods, i.e., MIPS based binary coding and online binary coding, which can efficiently and accurately achieve the morphological retrieval in large-scale neuron databases.

4.4.1 Experimental Setting

Our experiments are carried out on the NeuroMorpho [1], which has the largest collection of publicly accessible 3D reconstructed neuron data. Specifically, we use the entire 17,107 Drosophila Melanogaster neurons to evaluate the retrieval performance. Following the convention, we employ L-measure toolbox to extract 38 quantitative measurements as morphological features for each neuron [25], including 15 global, 10 branch and 13 bifurcation features respectively. In this chapter, all experiments are conducted on a 3.6GHz CPU with 4 cores and 32G RAM, in a MATLAB implementation.

We evaluate the performance by computing the retrieval precision, which is defined as:

$$Precision = \frac{|\{uPNs\} \cap \{Retrieved\ Neurons\}|}{|\{Retrieved\ Neurons\}|} \tag{4.19}$$

where $|\cdot|$ denote to count the number of samples inside. In the experiments, we compute the average precisions obtained over all queries.

4.4.2 Evaluation of MIPS based Binary Coding

To evaluate the efficacy of MIPS based Binary Coding for neuron morphological retrieval problem, we compare the retrieval performance in multiple protocols with three state-of-the-art unsupervised binary coding and hashing methods:

1. SH [41]: Spectral hashing is a well-known algorithm which harness nonlinear manifold structure to produce neighborhood-preserving compact binary codes;

2. ITQ [35]: Iterative quantization is based on PCA projection for dimensionality reduction and minimizes quantization error via orthogonal transformation. It is a very effective binary coding method for most natural image retrieval problem;

3. AGH [42]: Anchor graph hashing discovers the neighborhood structure inherent in the data to learn appropriate compact codes, which has already shown its excellent performance in mammogram retrieval [83].

As mentioned in Section 4.2, we calculate the measurements in three levels, i.e., global, branch and bifurcation. In practice, for computational convenience, we also group features in such three hierarchies. The similarity weights of each hierarchy are empirically obtained by the tree-like structures and neuronal properties. Due to the linear inseparability of neuronal structure, unrelated neurons are likely to express similarities in the global viewpoint. On the other hand, neurons with some common

Table 4.1: Comparison of retrieval precision with 16, 24, 32 bits of binary codes under different number of retrieved neurons.

Method	top10			top20			top50		
	16-bit	24-bit	32-bit	16-bit	24-bit	32-bit	16-bit	24-bit	32-bit
SH ([41])	0.8046	0.8116	0.8115	0.7211	0.7221	0.7264	0.5043	0.5102	0.5192
ITQ ([35])	0.8278	0.8298	0.8381	0.7595	0.7599	0.7615	0.6338	0.6394	0.6483
AGH ([42])	0.8254	0.8329	0.8353	0.7603	0.7861	0.7980	0.6325	0.6423	0.6874
Ours	**0.8889**	**0.8810**	**0.8909**	**0.8428**	**0.8378**	**0.8438**	**0.7436**	**0.7440**	**0.7451**

properties (e.g., cell types, brain regions) tend to express similarities in branch structure but different in bifurcation if they are not exactly the same. Therefore, we set global hierarchy with low weight to reduce the influence of non-linear structure. Then, we assign highest weight for branch hierarchy and next-highest weight for bifurcation hierarchy to make sure that we can retrieve neurons with common properties and also differentiate them in subtle level. In the experiment, we set global, branch and bifurcation hierarchies with the weights ratio of 1:14:5, which can achieve promising performance for neuron retrieval. In the learning of coding functions, training data sets $\mathbf{A}$ and $\mathbf{X}$ are random sampled, covering 80% of the whole Drosophia Melanogaster neurons database. For the MIPS based binary coding, maximum iterations of the inner loop and outer loop are 100 and 10 respectively. The trade-off parameter λ is set as 34. The length of binary codes for each neuron is scalable in our method, which is determined by the size of coding matrices $\mathbf{W}$ and $\mathbf{R}$. Generally, $\mathbf{W}$ and $\mathbf{R}$ are initialized by PCA projections. They can also be initialized by random generation if we want to obtain binary codes which are longer than the feature vectors.

As neuron morphology is correlated with their cell types and brain regions, for the Drosophia Melanogaster neuron database which has various cell types (around 100) and brain regions (around 50), we select 233 projection neurons (PN) in olfactory bulb and 19 lateral horn neurons (LH) in protocerebrum as queries, which is consist with previous works in evaluating neuron retrieval performance [8, 84]. In the testing phrase, the correctly retrieved neurons are defined as if they have the same cell types and brain regions with the query neuron.

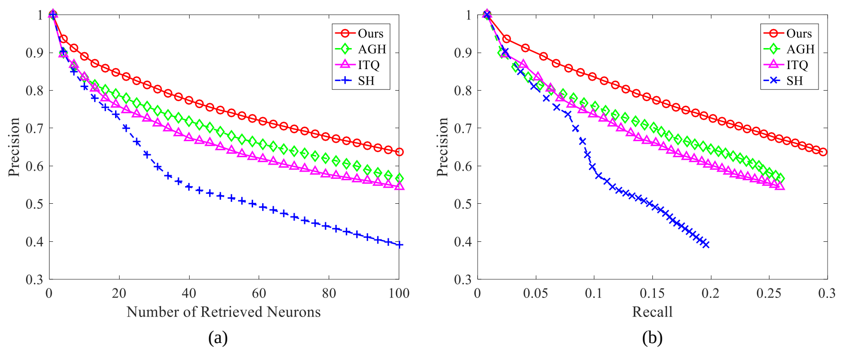


Figure 4.3: Retrieval performance of four compared methods, 32 bits of binary codes are used: (a) precision curve; (b) precision-recall curve.

For all the PN and LH queries (252 in total), Fig. 4.3(a) shows their average retrieval precision of four competitive methods under different number of retrieved neurons. Here, we denote retrieval precision as the percentage of correct neurons in all the retrieved neurons. Every method generates 32 bits of binary codes to represent each neuron. According to Fig. 4.3(a) we can see that our method significantly outperforms all other advanced binary coding methods in terms of retrieval precision. It mostly benefits from the feature hierarchy processing and MIPS based binary coding. As we group features into different hierarchies, in which each hierarchy reflect their corresponding levels of neuron representative, more suitable similarity matrices are obtained for the continued binary coding. Additionally, the employed MIPS baseline is more likely to generate effective binary codes for the linearly inseparable neuron data, since the inner product embedded objective function are more likely to map coding matrices into non-linear space, and the asymmetric optimization strategy provides a convergent solution. Fig. 4.3(b) provides the precision-recall curves for the above four methods. We can easily observe that the performance of the compared methods is consistent with the above analysis. Our method still performs the best among all compared methods.

We also report the retrieval precision of the compared methods using 16, 24 and 32 bits of binary codes respectively. As SH [41] and ITQ [35] can only generate

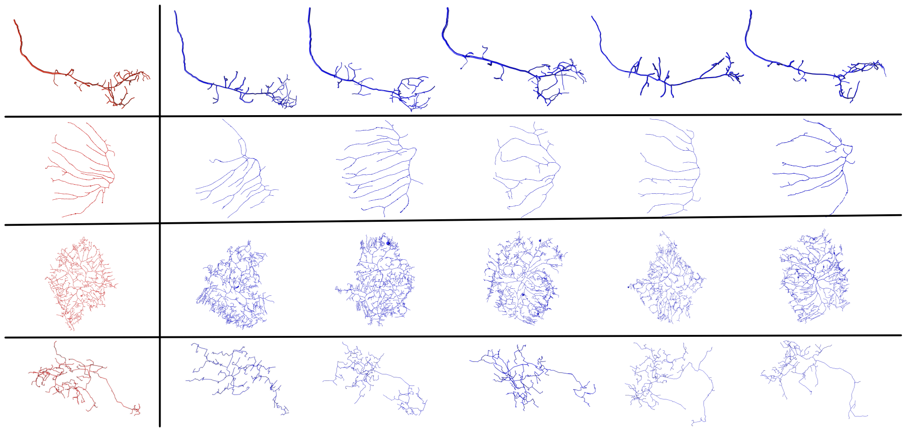


Figure 4.4: For each neuron on the left (red), top-5 retrieved neurons on the right (blue) through our method, which illustrate the morphological similarity between query neurons and retrieved neurons.

Table 4.2: Comparison of neuron retrieval precision and efficiency (in second) before and after MIPS based binary coding, using the feature fusion results.

	Precision	Time(s)
Fused-fea	**0.9083**	73.76
MIPS	0.8876	**0.15**

binary codes shorter than feature vectors (38 dimensions in this paper), we only compare the precision of the four methods with bits which are less than 38. From Table. 4.1, we find that our method can always achieve the highest precision under different bits of binary codes. These results verify the proposed method can generate more effective and representative binary codes for data residing embedded in the non-linear structure. In addition, from 16 bits to 32 bits, the retrieval precisions of our method are approximately equivalent, which demonstrate that asymmetric design for MIPS based binary coding can help us obtain convergent results in several alternative iterations.

Beside the retrieval precision, we evaluate the computational efficiency in the testing phase. In this part, we employ the whole NeuroMorpho database [1] which includes 58, 414 valid neurons. All these neurons are first represented using the about

78 dimensional fused features from SCAEs and L-measure toolbox. Then each neuron features are compressed into 32 bits of binary codes using the MIPS based binary coding. Table. 4.2 compares the retrieval precision and efficiency before and after the MIPS based binary coding, using the feature fusion results. For the 78 dimensional fused features, the learned coding functions compress them into 32 bits of binary codes. In this experiment, we also employ the 233 uPNs as queries and record their top-10 average precision. According to Table. 4.2, the retrieval precision after binary coding hasn't lose too much (i.e., only 2% lower compared with the feature fusion results). However, after compressing the fused features into binary codes, the 233 uPNs can achieve real-time retrieval in the whole NeuroMorpho database, i.e., sequentially indexing $58,414$ neurons with only 0.15 seconds in total. Compared with the exhaustive searching using the fused features, the efficiency of neuron retrieval has a significant improvement through binary coding (i.e., around 500 times faster). This superiority will be especially benefited in the future neuron retrieval tasks since an increasing number of neurons are reconstructed and added to large-scale databases [3].

Additionally, Fig. 4.4 presents four random selected query neurons and their corresponding top-5 retrieved neurons through our method. We employ *Vaa3D* [2] to display these neurons. Generally, the retrieved neurons present similar morphologies with their query neurons, which verify the effectiveness of feature extraction procedure and the proposed MIPS based binary coding method.

4.4.3 Evaluation of Online Binary Coding

In this experiment, we aim to demonstrate that the binary coding part can attain promising performance with the continuously expanding neuron database. We randomly split the 17,107 Drosophila Melanogaster neurons into two parts. The first 1,107 neurons are used as the original database, and the remaining $16,000$ neurons are equally divided into 100 batches (160 neurons in each), which are sequentially added to simulate the expanding size of the neuron database. Our online binary

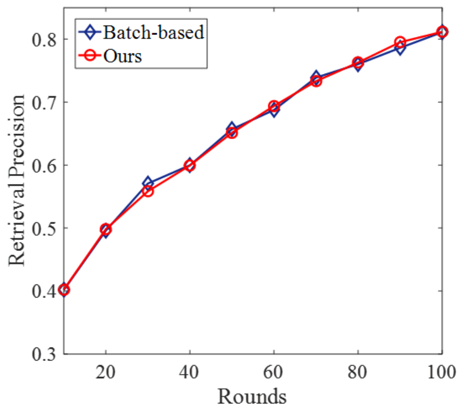
Table 4.3: Training time comparison (in second) under 100 rounds of update.

Rounds	20	60	100
Batch-based	1.12	13.24	51.11
Ours	**0.26**	**0.84**	**1.88**

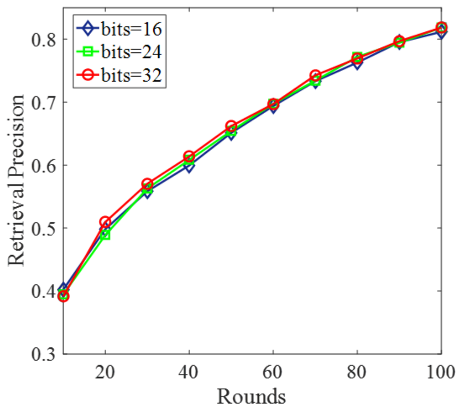
coding method is compared with the batch-based method. The batch-based method corresponds to the first part in Section 4.3.2, i.e., matrix sketching based binary coding. For each update, it needs to learn the coding function from scratch, using all neurons in the database. To overcome randomness, we repeat the experiments a hundred times to report the average. In the binary coding with online updating, due to the requirement of the FD algorithm [85], the sketched data size l should be no larger than the feature dimension d ($d = 38$ as discussed above). Since the feature dimension of neuron data is not high, we set $l = 38$ to preserve the information from the original database as much as possible.

Table 4.3 presents the accumulated training time of our method and batch-based method at the 20th, 60th and 100th update rounds respectively. Compared with the batch-based method, our binary coding with online updating shows great superiority in computational efficiency, and the superiority becomes more obvious with more rounds of updates, e.g., 51.11s versus 1.88s for one hundred updates. When new neuron batches are added to the database, our method only need to consider those newly added neurons and update the coding function on-the-fly, while the batch-based method needs to take all the neurons into account to re-train the coding function. The merit of this online binary coding method is particularly beneficial in the future, since an increasing number of neurons are reconstructed and added to the databases through the recently well-developed neuron tracing techniques.

Besides the superiority in computational efficiency, our binary coding with online updating also demonstrates its comparable performance in retrieval precision. Fig. 4.5(a) shows the average retrieval precision of two competitive methods, taking



(a)



(b)

Figure 4.5: Evaluation of the retrieval precision in 100 rounds of update: (a) Comparison of our method with the batch-based method; (b) Comparison of our method using different bits of binary codes.

their top-10 retrieved neurons into accounts. The learned coding functions compact the feature vectors into 32 bits of binary codes in this experiment. According to Fig. 4.5(a), our online method is able to achieve similar retrieval precision as the batch-based method. Therefore, the binary coding with online updating can significantly improve the computational efficiency without sacrificing the retrieval precision. This is mostly improved by: 1) the employed FD algorithm, which can compute the data sketch in a streaming manner; and 2) the newly introduced batch of data with virtual samples, which can overcome the mean-varying problem in a continuously expanding neuron database. Regarding the parameter, Fig. 4.5(b) shows the retrieval precision of our method when using 16, 24 and 32 bits of binary codes in each update. The online method can always achieve good performance using different bits of binary codes. These results verify that the online updated coding function can generate effective and representative binary codes for neuron morphological features.

4.5 Summary

In this chapter, we present two morphological retrieval frameworks for large-scale neuron exploration and mining. Specifically, we first introduce the feature hierarchy strategy to consider feature diversities with low computational complexity. We provide a novel binary coding method based on maximum inner product search, which can not only achieves fast neuron retrieval, but also differentiates the linearly inseparable morphological space with high precision. Then, we introduce an online binary coding strategy in tackling the continuously expanding neuron databases, which can updating the coding function on-the-fly without accessing the whole neuron database. Experimental results verify the efficacy of our neuron morphological retrieval method in tackling large-scale neuron databases. In the future, we will study how to learn effective coding functions under multi-modal features, which can embed different kinds of features in a generalized binary coding framework. Additionally, we will develop deep hashing method to integrate neuron feature representation and binary coding together, using latest deep learning techniques to directly compute binary codes from 3D neuron data end-to-end.

CHAPTER 5: INTERACTIVE EXPLORATION AND VISUALIZATION

5.1 Motivation

The above feature representation and binary coding methods can effectively tackle the neuron mining in large-scale databases. Despite the excellent performance of the above methods achieved, there are mainly two limitations that can influence the mining results of neuronal databases. Firstly, current neuron databases lack supervised information, i.e., no sufficient annotations to label the class of every neuron [86, 14]. Generally, supervised retrieval and mining are more accurate in comparison with unsupervised methods, since it can bridge the gap between low-level image descriptors and high-level semantic meaning. In the neuron retrieval problem, if we only consider the low-level neuronal morphologies, the retrieval results may not be consistent with their functions and properties. Secondly, binary coding can only provide coarse retrieval results for neuron morphological data [87, 88, 89]. As each neuron cell has a tree-topological structure, the difference among neuron morphologies can be subtle. Moreover, coding functions do not have one-to-one correspondences when mapping morphological features into binary codes. This may present a difficult question, in which some unrelated neurons are represented by the same binary codes. The above two problems should be addressed to achieve good retrieval performance in the neuron morphological data.

In order to improve retrieval performance and reduce the semantic gap, some image retrieval systems introduce domain experts/users in the loop, which can interactively provide relevance feedback for the previous retrieval results [50, 52, 54, 53]. However, for untrained users, it is hard to differentiate the complicated tree-topological structures in neuron morphologies, particularly with the current visualization tool-kits which the neurons are visualized in 2D static interface. The limitation of neuronal visualization performance may results in inaccurate relevance feedbacks, and thus it may inversely influence the mining results of the whole system.

To alleviate these problems, we first design an interactive exploration method to bring users in the loop. Particularly, based on the retrieval results from feature representation and binary coding, we introduce domain experts/users in our framework, which can give relevance feedback to improve the retrieval accuracy. In our feedback model, domain experts/users are only required to label the relevant samples with respect to query neurons from top-z results. Then, the similarity levels of the unlabeled neurons will be re-ranked accordingly through our newly designed similarity measure. Moreover, to help users take a deep understanding of neuron morphologies, we develop a novel neuron visualization program through the Microsoft HoloLens augmented reality (AR) headset. To the best of our knowledge, this is the first work that applied user interaction and augmented reality techniques for the analytics of 3D neuron data.

5.2 Interactive Neuron Exploration

According to the above binary coding method, we can compress the query neuron and all neurons in the current database into short binary codes, through the learnt coding functions. Then the similar neurons can be retrieved based on their Hamming distance ranking or maximum inner product with the query neuron. As discussed in Section 4.1, binary coding can only provide coarse retrieval results for the neuron morphological data. Therefore, given the coarsely retrieved neurons (e.g., neurons with top-Z minimum Hamming distance), we propose to introduce domain experts/users in the framework, which can interactively provide relevance feedback to refine the retrieval results.

Receiving Feedbacks from Users: Fig. 5.1 presents an illustration of our user interaction interface. For a query neuron, it first searches similar neurons based

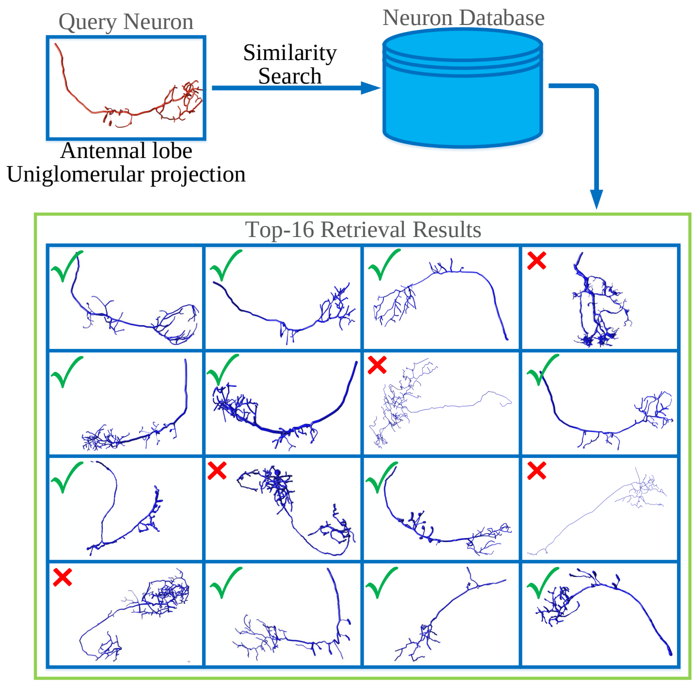


Figure 5.1: An illustration of our user interaction interface, users will give feedback by one-click inputs.

on the aforementioned binary coding method. Then we display the top-z (z=16 in Fig. 5.1) retrieved neurons to users, and users will compare and observe these neurons to decide whether they are relevant to the query neuron. This feedback scheme is easily implemented since it requires users to give only one-click inputs. Unlike many interactive models which require users to specify the class of the retrieval results [53, 90, 91], our strategy is particularly suitable for neuron databases which have insufficient annotations to classify every neuron.

Improving Retrieval Performance Using Feedbacks: After receiving the interactive feedback from users, our framework is able to process this feedback to improve the retrieval performance. Benefiting from the binary coding step which can efficiently provide the coarse retrieval results, we first define the outer scope size Z, where most similar neurons are included in the top-Z coarse results. Subsequently, we define the inner scope size z, which represents the number of neurons that should be presented to the users at each feedback round. In practice, Z is larger than z but much smaller than the size of the whole neuron database. During the interactive neuron retrieval phase, we focus on re-ranking these Z coarse neurons to obtain fine-grained results.

Assuming in the t-th round of relevance feedback, $\mathbf{V}^{(t)}$ is the set of labeled similar samples from users which include $m^{(t)}$ neurons. As neurons in $\mathbf{V}^{(t)}$ are all similar with the query neuron, we can assemble them together to interpret and re-rank the similarities for the rest of the unlabeled neurons. Denoting $E(\mathbf{x}_i, \mathbf{x}_j)$ as the similarity measure between two neurons $\mathbf{x}_i$ and $\mathbf{x}_j$, for one unlabeled neuron in tth round of relevance feedback $\mathbf{x}_i^{(t)}$, we re-define its similarity with the query neuron as follows:

$$E_i^{(t)} = \lambda E(\mathbf{x}_i^{(t)}, \mathbf{x}_q) + (1 - \lambda)\frac{1}{m^{(t)}} \sum_{j=1}^{m^{(t)}} E(\mathbf{x}_i^{(t)}, \mathbf{x}_j^{(t)}) \tag{5.1}$$

where $\mathbf{x}_q$ is the query neuron and $\mathbf{x}_j^{(t)}$ is the neuron in $\mathbf{V}^{(t)}$. The above objective function indicates that if an unlabeled neuron has similarity with the query neuron, it should be similar with the labeled neurons as well to some extent under the trade-off parameter λ.

According to Eq. 5.1, how to compute the similarity measure is a critical issue for re-ranking the unlabeled neurons. In most scenarios of image retrieval, the similarity measure between two images is defined as the Euclidean distance of their feature vectors. However, this similarity measure cannot be directly applied for the neuron morphological data, since each dimension of neuron features are distinct quantitative measurements which have different levels of representation. As discussed in previous sections, we compute three levels of measurements as features based on the neuron's tree-topological structure, i.e., global, branch and bifurcation. Accordingly, we propose to group features into these three levels and assign them with different representative weights (i.e., ω_{gl}, ω_{br}, ω_{bi}) to compute a more accurate similarity measure.

For the two neurons $\mathbf{x}_i$ and $\mathbf{x}_j$, their new similarity measure can be formulated as:

$$E(\mathbf{x}_i, \mathbf{x}_j) = \omega_{gl} D_{gl}(\mathbf{x}_i, \mathbf{x}_j) + \omega_{br} D_{br}(\mathbf{x}_i, \mathbf{x}_j) + \omega_{bi} D_{bi}(\mathbf{x}_i, \mathbf{x}_j) \tag{5.2}$$

where D_{gl}, D_{br}, D_{bi} denotes the normalized Euclidean distance of global, branch and bifurcation features respectively. This similarity measure is specifically designed for neuron morphological data. Finally, with the newly defined similarity measure, we re-rank all the unlabeled neurons in ascending order based on their results in Eq. 5.1, and present the updated top-z results to users. Users can iteratively give relevance feedback for these z neurons until they satisfy with the retrieval results.

5.3 Neuronal Visualization

The above interactive strategy can receive relevance feedbacks from users and subsequently improve the retrieval performance based on the relevance feedbacks. Therefore, one important problem is how to visualize and present these retrieved results to users. Moreover, neuroscientists require more comprehensive visualization software to help them explore and analyze the neuron morphology. To this end, we develop a novel neuron visualization program in assisting neuron exploration through the Microsoft HoloLens augmented reality (AR) headset. Here, we introduce the implementation details of the AR based neuron visualization program.

Our visualization program is developed using the Vuforia AR platform and the Unity game engine, which can create a powerful tool for the application development on Microsoft's Windows Holographic Platform. Particularly, when visualizing the neurons, we start by mapping the data points for each neuron to the real world. Based on the spatial anchors (i.e., data structures containing a world coordinate and information about the surrounding environment), anchor points are first created in the virtual environment that the HoloLens has generated in the correspondence with the real world. Each anchor point has their own coordinate systems in which the data points are rendered. To compete for the visualization, the Unity's built-in line renderer are employed to create lines connecting each data point to their parent data point, according to the *SWC* data format provided in neuron databases. The package use the Bézier curve in connecting these lines, which is a type of parametric curve that can be scaled indefinitely. Once the neurons are mapped in the virtual environment, the Hololens uses stereo images to visualize the neurons on the transparent screens of the headset, so that they appear as if they existed in the real world in front of the user. As the anchor points correspond to the real world, the visualizations will appear to be fixed to a given location before the user unless moved. In addition, the neuron visualizations are rendered with depth-based coloring, which adjusts based on the user's proximity to the neuron.

We also make the visualized neurons interactable. We place a transparent transform at the center of each neuron, which also acts as the parent for all the data points of an individual neuron. Upon placing the visually directed cursor of the HoloLens on the center of the neuron, the cursor gives a visual cue, which indicates that an object is interactable. As the transform is transparent, this gives the impression that the neuron itself is the object you are interacting with.

5.4 Experiment

In this experiment, we first evaluate the effectiveness of our proposed interactive neuron exploration method. Then we provide the visualization results using the Microsoft HoloLens Headsets.

5.4.1 Evaluation of Interactive Neuron Retrieval

In this part, we aim to validate that the interactive strategy can actually achieve good performance for the neuron retrieval problem. We will re-fine the coarse retrieved neurons from previous binary coding results, where the neurons are retrieved from the entire $17,107$ Drosophila Melanogaster neuron database using 32 bits of binary

Table 5.1: Retrieval precision of four methods under different number of retrieved neurons.

	top20	top30	top40	top50
ITQ [35]	0.7673	0.7249	0.6948	0.6614
AGH [42]	0.7589	0.7216	0.6951	0.6735
MIPS [84]	0.7923	0.7508	0.7088	0.6828
Ours	**0.9015**	**0.8550**	**0.7888**	**0.7092**

codes compressed from the 38 dimensional L-measure features [25]. Particularly, we only consider to refining neurons within top-Z minimum Hamming distance (neurons in outer scope size). For the user interaction, users will give feedback for the top-z unlabeled neurons (neurons in inner scope size).

To evaluate the performance of interactive neuron retrieval, at each feedback round, the top-z unlabeled neurons will be automatically labeled using the ground truth in order to simulate the user's feedback. Since our interactive method only requires users to give one-click inputs(relevant or non-relevant), the ground truth can be easily achieved by checking whether the unlabeled neurons are uPNs or not.

We compare our neuron retrieval method with three state-of-the-art methods, i.e., ITQ [35], AGH [42] and MIPS [84], which are all proposed to tackle the retrieval problem for large-scale databases. ITQ [35] is a very effective binary coding method for most natural image retrieval problems. AGH [42] has already achieved excellent retrieval performance in mammogram data [83], and MIPS is specially designed for the neuron morphological retrieval problem. The above three methods are batch based method, which can not update the retrieval model after relevance feedback. Thus, for fair comparison, the retrieval model of these methods are trained through the currently entire $17,107$ Drosophila Melanogaster neuron database. Consistent with previously experimental setting, we employ the 233 uPNs as queries to validate the retrieval performance.

Table 5.1 reports the average retrieval precision of four competitive methods under different number of retrieved neurons. For our interactive method, the retrieval precision is recorded after 3 rounds of feedback. According to Table 5.1, our method can achieve the highest precision under different number of retrieved neurons. These results verify the proposed method is effective for the neuron retrieval problem. It mostly benefits from the interactive strategy which introduces users in the loop to give feedback for the coarse retrieval results. Specifically, based on the user's feedback, our method can re-rank unlabeled neurons by the newly designed similarity measure.

We randomly select a query neuron and present its top-20 retrieval results in Fig. 5.2 under different rounds of feedback. We employ *Vaa3D* [2] software to display these neurons. The neurons with green frames are relevant to the query, and neurons with red frames are not relevant to the query. Generally, the retrieval performance has a significantly improvement from coarse results to the results after user feedback, which verifies the effectiveness of the proposed interactive strategy. We also find that with the increased numbers of feedback rounds, the retrieval performance improves accordingly. This is because of the increasingly labeled neurons, providing more information for re-ranking. In addition, according to Fig. 5.2, many non-relevant neurons also present similar morphologies with the query, which are usually hard to differentiate through traditional retrieval methods. Thus, our interactive strategy is a good choice for the fine-grained neuron retrieval problem.

In our interactive neuron retrieval, two parameters may influence the final performance, i.e., the outer scope size Z and the inner scope size z. In the interactive part, we only consider the refinement of top-Z ranked neurons from coarse retrieval results. Fig. 5.3(a) shows the average retrieval precision with different outer scope sizes after the 1st to 10th feedback rounds, taking top-30 retrieved neurons into account. In Fig. 5.3(a), we find that with the outer scope size ranging from 200 to 500, the retrieval precision has not changed too much. This is because the majority of

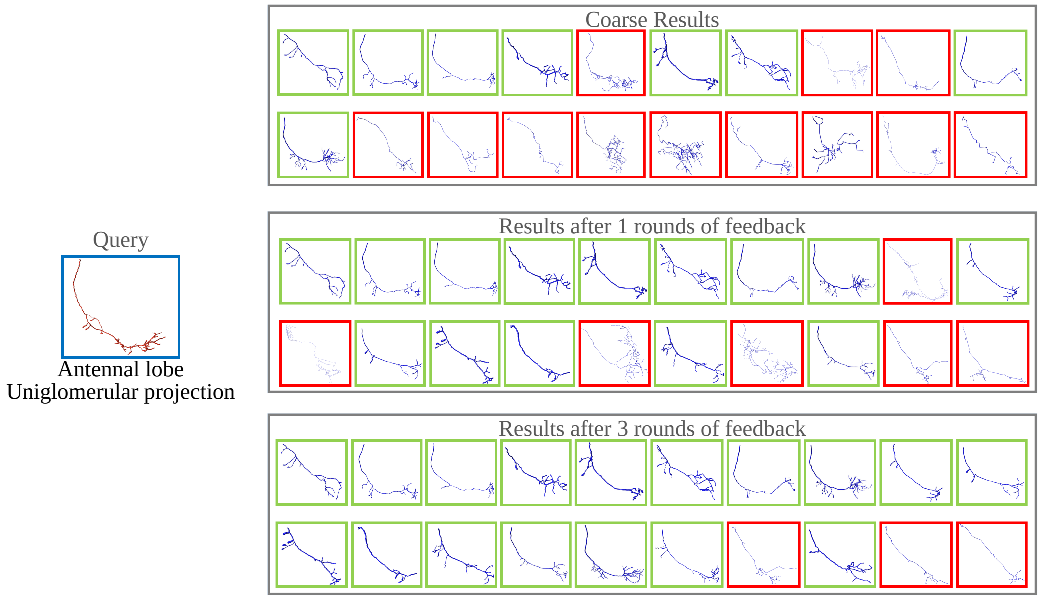

Query
Antennal lobe
Uniglomerular projection
Coarse Results
Results after 1 rounds of feedback
Results after 3 rounds of feedback

Figure 5.2: Query example of the proposed method under different rounds of feedback: green framed neurons are relevant with the query, while red framed are non-relevant neurons.

relevant neurons are already included in a small sized outer scope (e.g., $Z = 300$). A larger outer scope may include more relevant neurons, but non-relevant and noisy neurons are also included which will influence the retrieval performance. Moreover, the inner scope size z is the number of neurons we provide to users in each feedback round, which can also influence the final performance. Fig. 5.3(b) presents the retrieval precision with different inner scope sizes after the 1st to 10th feedback rounds. According to Fig. 5.3(b), the larger z can achieve better performance compared with smaller z values. This is easy to understand since a larger z will contain more feedback information which can help us to re-rank the coarse retrieved neurons.

5.4.2 Demonstration of Neuron Visualization

Fig. 5.4 presents some screen shots of our visualization results. The developed AR program can interactively and immersively visualize neurons in different scales and different view angles. As shown in Fig. 5.4, the program can first present some retrieved neurons to users using our neuron retrieval method. Then users can utilize

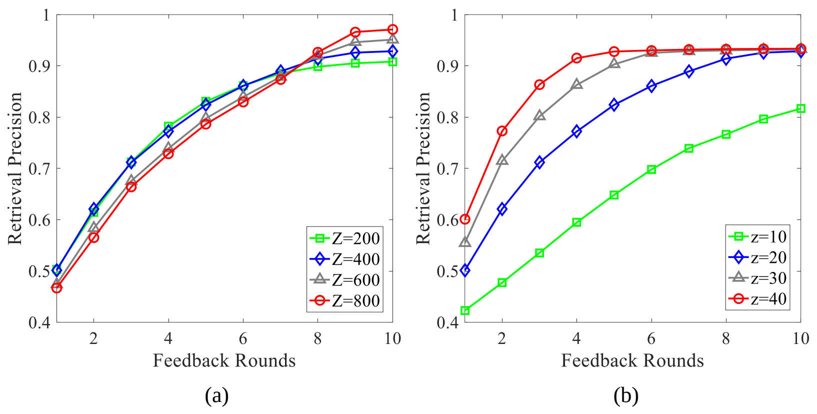


Figure 5.3: Retrieval performance with different parameter settings: (a) retrieval precision with different outer scope size after 1st to 10th feedback rounds; (b) retrieval precision with different inner scope size after 1st to 10th feedback rounds.

the HoloLens built-in “finger tap” gesture to select an interested neuron for further analysis. Once selected, various voice commands, created by us but powered by Microsoft “Cortona” voice assistant technology, can be used to identify the type of interaction to be performed on neurons. We included functionality allowing users to move, scale, and rotate the selected neurons. Users can also use the built-in “pinch” gesture along with hand motions to manipulate neurons. This provides a simple and intuitive way for users to interact with the visualizations, and to do so from any position in the environment. The screen shots in Fig. 5.4 can only provide a glimpse of how our program works, where it lost the biggest benefits that are gained freeing users from a screen. Compared with other neuron visualization techniques (e.g., *Vaa3D* software [2]), the main advantage of our AR based program is that it can visualize neurons in an interactive and immersive way, and thus help users taking a deep and comprehensive analysis of neurons.

5.5 Summary

In this chapter, we present an interactive neuron mining framework, which bring users in the loop to improve the neuron mining performance and visualize neurons in an interactive and immersize manner. Particularly, users can observe the fine-grained

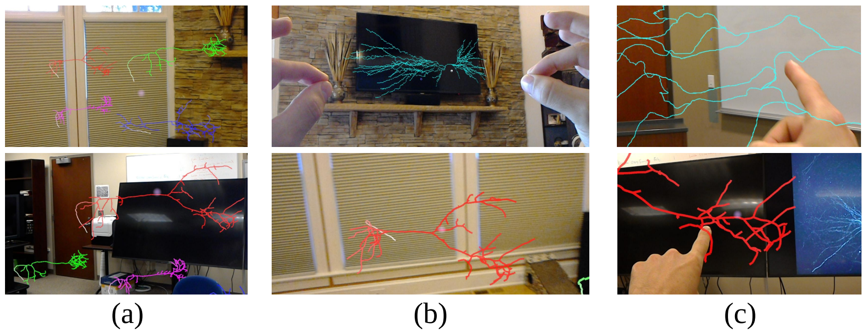


Figure 5.4: The screen shots of neuron visualization and analysis using the Microsoft HoloLens AR headset: (a) visualize some of the retrieved neurons; (b) manipulate and enlarge interested neurons ; (c) continue to enlarge neurons, interactively analyze their fine-grained details.

details of neurons based on recent advances of augmented reality techniques. Our system can receive the relevance feedback from users' observation, then updating the retrieval performance by re-ranking the similarity measure between the query neuron and its top retrieved neurons. Experiments verify the efficacy of our neuron retrieval framework and also illustrates its application in neuron exploration. Based on the present work, we will develop a comprehensive tool for efficient and accurate neuron mining, which can help biologists to explore and analyze unknown neurons.

CHAPTER 6: Applications

6.1 Motivation

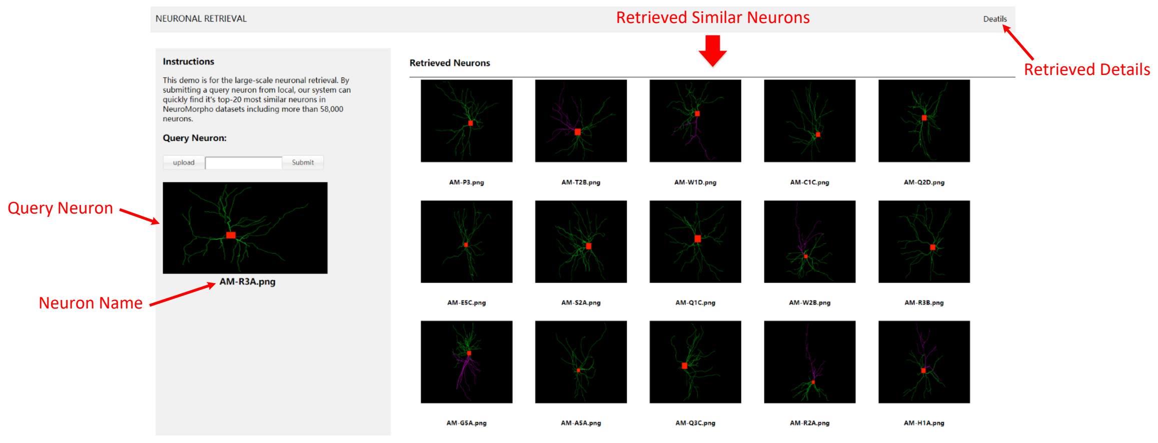
The above sections propose multiple methods in tackling the neuro-morphological analysis for large-scale databases. Despite these methods have achieved state-of-the-art performance in neuronal feature representation and binary coding, how to integrate and present the mining results to users are still challenging problems, where the current outputs are simply retrieval precision curves and some hand-crafted retrieval results. Considering the practical requirements of users in analyzing the large-scale neuron databases, more intuitive and comprehensive results should be provided for further mining and exploration. Therefore, systems and tools need to be developed to integrate the proposed methods and mining results in a generalized framework.

Currently, there has been several systems developed for the exploration of neuron morphology. For example, BlastNeuron [8] is developed for automated comparison, retrieval and clustering of 3D neuron morphologies. This software pipeline first employs global morphological features to identify similar patterns in neuron databases. Based on the retrieval results, BlastNeuron [8] performs clustering analysis to detect biologically meaningful neuron classes, and neuron comparison for reconstruction alignment. More recently, an advanced tool for morphological search and retrieval in neuroscientific image databases is presented, i.e., Neuron-Miner [9], which is an Android app for neurons' similarity search. This app can achieve neuron search by set the neuron name in NeuroMorpho [1] database as input. Additionally, NBLAST [26] is an online neuron search engine, which focus on measure the pairwise neuron similarity. This search engine is developed for the neurons in drosophila melanogaster, which can map each neuron in the drosophila's brain model. Despite the above systems are effective in solving neuron mining problems, there are still multiple aspects that can be explored to further improve the performance and usability of neuron mining systems. Firstly, the neuron search performance, i.e., precision and efficiency, can be further improved, especially tackling the continuously expanding large-scale neuron databases. Secondly, more semantic results should be provided in the system for neuron mining, instead of simply provide the results of neuronal morphology. The semantic results can be the neurons' functional properties, brain regions, transmitters, etc.

Taking the above requirements into account, this section first presents our system for neuron retrieval and analysis, based on the novel methods we proposed in previous sections. Particularly, the system is a fully online search engine, which embed all datasets and methods in the server side, and enable every process on-the-fly without any installation and configuration. The developed system can not only achieve accurate and efficient neuron retrieval in real-time, but also provide abundant semantic information for all neurons in helping neuron exploration and knowledge discovery. Moreover, we demonstrate and discuss applications of this system in multiple neuron mining use cases, including the identification of unknown neurons, clustering neurons with different similarity level, and comparison of different neurons to refine the neuron reconstruction.

6.2 Online Search Engine for Neuronal Retrieval

As discussed above, despite there are several neuron search systems developed [8, 9, 26], the performance and usability of the above systems can be further improved. Particularly, when designing the neuron search system, multiple factors need to be considered, i.e., 1) the whole system need to achieve the neuron retrieval with high accuracy and efficiency, which can accurately find similar neurons in real-time; 2) the system can provide abundant information related to neuron's semantic meaning, which can help users to correlate neuronal morphologies with functional properties; 3)



Instructions

This demo is for the large-scale neuronal retrieval. By submitting a query neuron from local, our system can quickly find it's top-20 most similar neurons in NeuroMorpho datasets including more than 58,000 neurons.

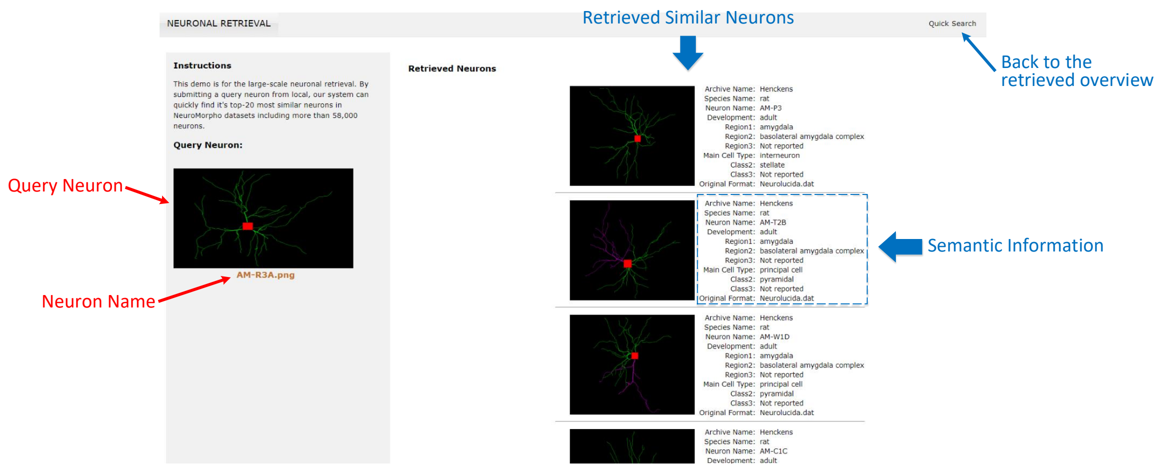
Query Neuron:

Figure 6.1: The main webpage of our online neuron search engine, which includes the query panel (left part), and the retrieval panel (right part).

the system need to be easily used, where users can process the whole pipeline without installation and configuration. In this section, we present our online search engine for the large-scale neuron morphological retrieval and mining. Especially, we introduce the implementation of online search engine in both browser side and server side.

6.2.1 Browser Side

Fig. 6.1 presents the main webpage of our online neuron search engine. The left part of the webpage is the input panel. Users can click the "upload" button to select one neuron file from local, the left panel will subsequently visualize this selected neuron and the corresponding neuron name. When users want to find its most similar neurons in the large-scale database, simply click the "submit" button can get the results on-the-fly. As illustrated in Fig. 6.1, the right panel can present the top similar neurons and neuron names with respect to the query neuron on the left panel. Users can observe and compare these retrieved neurons for further analysis and mining. Moreover, by clicking the "Details" button on the top right, our system can provide the detailed semantic information for each retrieved neuron. As shown in Fig. 6.2, the left panel provide the query neuron, while the right panel sequentially provide the neuron images and corresponding information for each neuron,



Instructions

This demo is for the large-scale neuronal retrieval. By submitting a query neuron from local, our system can quickly find it's top-20 most similar neurons in NeuroMorpho datasets including more than 58,000 neurons.

Query Neuron:

AM-R3A.png

Retrieved Neurons

```
 Archive Name:  Henckens
 Species Name:  rat
  Neuron Name:  AM-P3
  Development:  adult
      Region1:  amygdala
      Region2:  basolateral amygdala complex
      Region3:  Not reported
Main Cell Type: interneuron
       Class2:  stellate
       Class3:  Not reported
Original Format: Neurolucida.dat

 Archive Name:  Henckens
 Species Name:  rat
  Neuron Name:  AM-T2B
  Development:  adult
      Region1:  amygdala
      Region2:  basolateral amygdala complex
      Region3:  Not reported
Main Cell Type: principal cell
       Class2:  pyramidal
       Class3:  Not reported
Original Format: Neurolucida.dat

 Archive Name:  Henckens
 Species Name:  rat
  Neuron Name:  AM-W1D
  Development:  adult
      Region1:  amygdala
      Region2:  basolateral amygdala complex
      Region3:  Not reported
Main Cell Type: principal cell
       Class2:  pyramidal
       Class3:  Not reported
Original Format: Neurolucida.dat

 Archive Name:  Henckens
 Species Name:  rat
  Neuron Name:  AM-C1C
  Development:  adult
```

Figure 6.2: The webpage in providing detail semantic information for the retrieved neurons.

including the species name, development stage, brain regions, cell classes, etc. Generally, these semantic information can help users have a deep understanding of each retrieved neuron, in comparison with the query neuron in both morphologies and functional properties. Currently, our system support the input of "png" images with the neuron name appeared in NeuroMorpho [1] database. The link of our webpage is: http://45.40.199.91:8008/home/main/, users can directly download the presented neurons and set them as input for further retrieval and analysis.

6.2.2 Server Side

After presenting the browser side of our system, we introduce how the online neuron search engine can be achieved in the server side. Fig. 6.3 demonstrates the overall work flow of our system in the server side, which is composed into three layers, i.e., interface layer, business layer and database layer. Here, we introduce the communication details among these three layers:

1. Interface Layer: this layer plays as the interface between users and the server side. According to the browser side design described above, this layer include three input process, i.e., upload the query image, submit the request of neuron

retrieval, and submit the request of providing detailed semantic information for each retrieved neuron. Besides, this layer also include three output process, i.e., visualize neuron images of retrieval results send by the business layer, neuron names with regard to the retrieved neuron images, and the semantic information correspond to the retrieved neurons. Basically, this layer has two functions: 1) receiving inputs from users and sending the input information to the business layer; 2) receiving information from the business layer and presenting the retrieved information for users to visualize and analyze;

2. Business Layer: this layer plays as the connection between interface layer and database layer, as well as the processing of the whole retrieval framework. In the business layer, it first receive the inputs from interface layer. For the input "upload", this layer save the uploaded images with their neuron names and stored them in the cache. Then it sends them to the interface layer for visualization. For the input "submit", this layer first capture the current neuron from the cache, querying the similar neurons in the database layer and then return the retrieved neuron images as well as names to the interface layer. For the input "Details", the business layer use the cache of image data to query the semantic information of retrieved neurons in the database, then sending these results to the interface layer. Basically, the business layer has two functions: 1) receiving requests from interface layer, processing these requests and returning the results to the interface layer; 2) querying tasks with the database layer, and receiving query results from the database.

3. Database Layer: this layer stores the neuron data set and related databases to meet the request from business layer. The database layers can respond the request from the business layer with regard to the query of neuron images, neuron names, similarity ranks, etc. Especially, we design an database in this

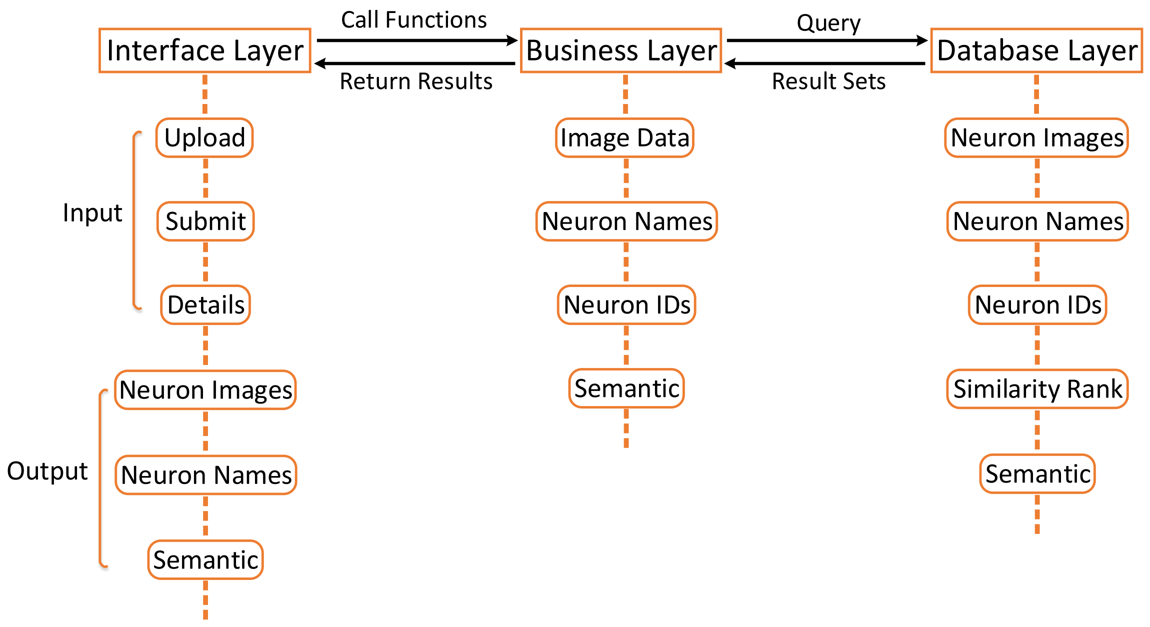


Figure 6.3: The work flow of our designed neuronal retrieval system in server side.

layer with abundant information for neuron retrieval. For each neuron, the database stores its corresponding neuron IDs, neuron names, neuron images, list of similar neurons' ID, and neuron's functional properties. When business layer request for a neuron retrieval task, the database layer can first index the query neuron and then return a set of similar neurons' ID. Based on these IDs, the database layer can subsequently return the retrieved neuron images. Basically, the database layer has two functions: 1) storing the relevant dataset for neuron retrieval; 2) returning the query request from business layer.

Based on this neuron search system, users can effectively explore neuron datsets and correlate neuron morphologies with their functional properties. Particularly, compared with previous neuron search systems, advantages of our system can be reflected in three aspects: 1) the methodological superiority of accuracy and efficiency in our system can make sure the fine-grained neuron retrieval in real-time, which mainly benefits from our proposed feature representation and binary coding; 2) the designed system is a web-based program which is easy to use, without any installation and configuration, which is particularly suitable for the user interactive mining ; 3) compared with previous systems, we provide two panels in displaying both neuron morphologies and detailed semantic information of each retrieved neuron, where users can either compare the morphology or semantic information among all retrieved neurons.

6.3 Use Cases

Considering the excellent performance the proposed methods have been achieved and the online search system have been developed, it is possible to apply them in the exploration and mining of neuronal morphology. Particularly, our methods and system can be used in neuron mining tasks such as neuron identification, clustering, comparison, etc. Here, we demonstrate and discuss multiple use cases of our methods and system.

6.3.1 Neuron Identification

One important use case of our framework is the exploration and analysis for unknown neurons. Currently, despite the fact that an increasing number of neurons are reconstructed and added to the public databases, most of them are not well identified and lack basic annotations, such as cell classes and brain regions. Nanda et al. [14] propose to annotate brain regions and cell classes for the NeuroMorpho [1] database. They employ the text-based query tool to search neurons with given lengths (e.g., 10, 20 microns) in each region to determine their brain regions, then identify cell classes based on the brain regions invaded by the neurite terminals of every neuron. This method may inefficient and unreliable, which the annotations are mainly obtained by empirical measurement. Therefore, identity of unknown neurons is an urgent demand in current neuron repositories.

Considering that neuron morphologies are associated with their properties, and our neuron retrieval framework can search similar neurons at a fine-grained level. It is reasonable to employ our framework to conduct neuron exploration via examining

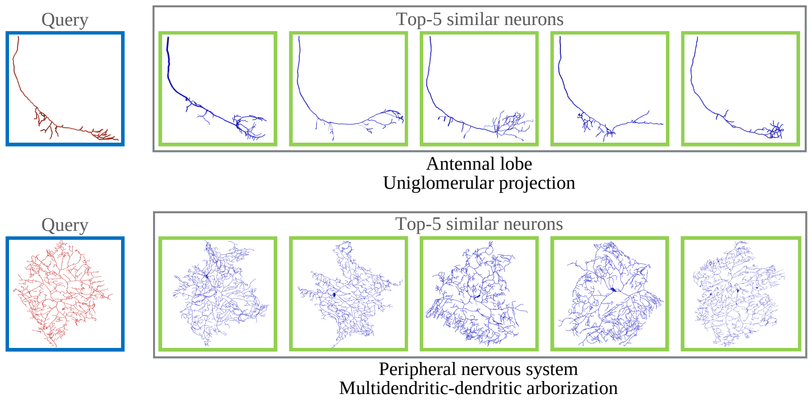


Figure 6.4: Illustration of two unknown neurons and their top-5 retrieved neurons through the proposed method.

retrieved neurons, which have similar morphologies. To demonstrate this, we random select two query neurons from NeuroMorpho [1], whose neuron types are assumed to be unknown. After running our neuron retrieval framework, Fig. 6.4 illustrates their top-5 similar neurons respectively. For the two query neurons, we find that their corresponding top-5 similar neurons all have the same neuron types, i.e., in NeuroMorpho [1], the 5 neurons in first row are annotated as antennal lobe and uniglomerular projection (uPNs), the 5 neurons in the second row are annotated as peripheral nervous system and multidendritic-dendritic arborization. Therefore, we can infer that the two query neurons also have the same type with their top-5 retrieved neurons. The information provided in NeuroMorpho [1] also verifies our inference about the two query neurons. In practical situations, we can employ more retrieved neurons (e.g., top-30 similar neurons) to statistically identify and analyze query neurons.

Our methods and system can accurately and efficiently achieve the above neuron retrieval task, as it is designed for the exploration of large-scale neuron databases. The method will be particularly suitable in the future since big data is one major direction in neuroscience [3]. Besides the efficiency, for some specific neuron databases which are not very large (e.g., considering neurons in some specific brain regions

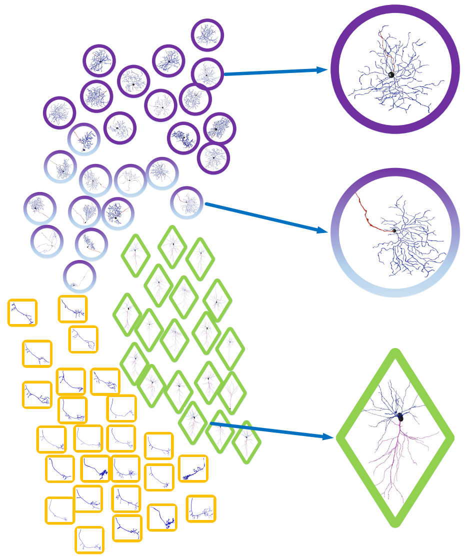


Figure 6.5: A prototype of hierarchical clustering, including three coarse clusters (yellow, green, magenta) and two fine-grained clusters (cyan diamond and square).

with only hundreds of neurons), exhaustive search and comparison can be applied to achieve more accurate results. In addition, extracting more representative features for the 3D neuromorphological data will be also helpful to improve the neuron retrieval performance.

6.3.2 Latent Pattern Discovery and Exploration

Currently, the taxonomy of neurons are still insufficient, especially in the recently released large databases, where many neurons are grouped together simply because they share same transmitters or locate in same brain regions. In the field of neuroscience, neuron type classification is a long-term research topic. As neurons has diverse molecular, morphological, connectional and functional properties, it is beleevied that the only realistic way to manage this complexity, and thereby pave the way for understanding the structure, function and development of brain circuits, is to group neurons into types, which can then be analyzed systematically and reproducibly [92]. For the neuron classification, one critical taxonomy is based on their morphology, where the neuromorphology has a strong relevance with neuronal con-

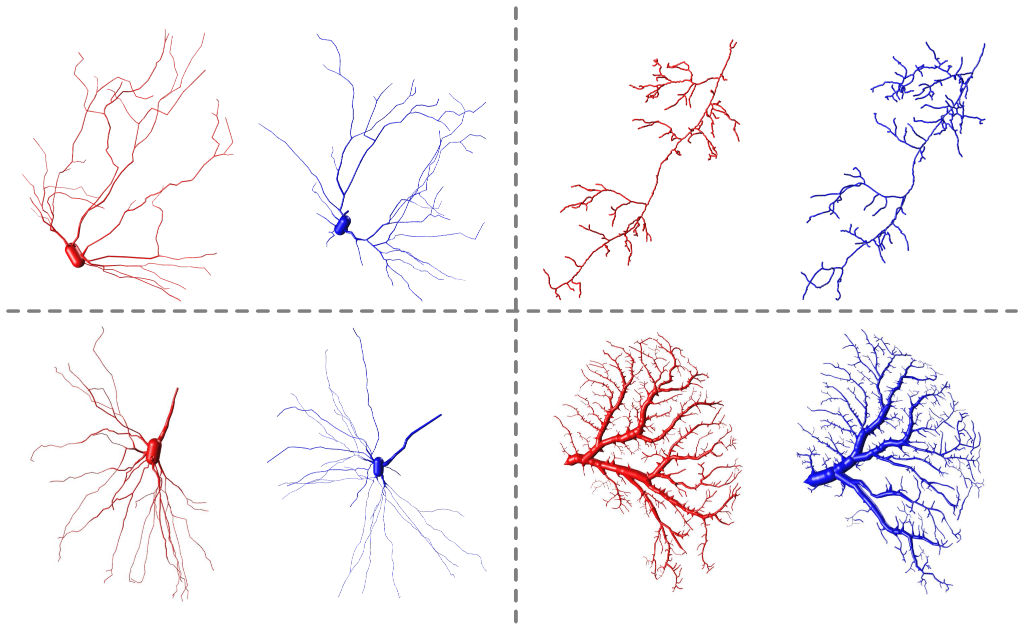


Figure 6.6: Four query neurons (red) and their corresponding most similar neurons (blue) after searching in the NeuroMorpho [1] database, which can be further used for neuronal comparison and analysis.

nection and functional property. Considering that the hierarchical clustering can group neurons with multiple levels of relevance [8, 26], it is possible to improve the taxonomy of neurons using our feature representation and binary coding methods. Fig. 6.5 illustrates a prototype of hierarchical clustering. The three coarse clusters (circle, diamond, square) can only distinguish neurons which have significant differences, while the two fine-grained clusters (purple circle and half purple circle) can reveal the subtle differences, i.e., these two clusters of neurons are quite relevant but belong to different categories. In practice, such fine-grained clusters can help to discover and define latent patterns of neurons.

The other use case of our system is neuron comparison. As shown in Fig. 6.6, we random select four neurons (red) as queries, then finding their most similar neurons (blue) after searching in the NeuroMorpho [1] database, using the proposed neuron mining methods. According to Fig. 6.6, query neurons demonstrate high degree of similarities in comparison with their retrieval results. On the basis of this accurate results, neuron comparison can be applied for several neuron exploration tasks. For example, by analyzing the topological morphology of two similar neurons with branch to branch comparison, their fine-grained difference can be pinpointed with quantitative results. These results can be used to quantitatively describe the difference of morphologies with their corresponding functional properties. In addition, our neuron mining system can provides useful information to help neuroscientists explore neurons and investigate the relationship between their morphologies and functional properties. How the morphology influences and determines neuron's functional property? Which parts of difference in neuron's structure lead to their diversity in certain functions? All these questions have great significance to achieve the ultimate goal in neuroscience, i.e., understanding brain working mechanism and consciousness. We are able to explore the correlation of neuron morphologies and functional properties from macro to micro. The available large-scale databases provide a macro platform with massive information of neuron morphologies and functions. Then, our mining system can efficiently retrieve and cluster neurons in fine-grained levels, with similar morphologies and corresponding semantic meaning.

6.4 Summary

In this chapter, we present applications and use cases of our neuron mining framework. Particularly, we first introduce our designed system of real-time neuron retrieval based on the aforementioned feature representation and binary coding methods. The developed system is a web-based online search engine, which can provide similar retrieved neurons, as well as their semantic information for neuroscientists to explore and analyze neurons. Moreover, we demonstrate multiple use cases of our proposed methods and system, such as neuron identification in finding semantic information for unknown neurons, neuron classification and clustering in unveiling latent taxonomy of massive neurons, and neuron comparison in correlating neuron morphologies with their functional properties.

CHAPTER 7: CONCLUSIONS AND FUTURE WORKS

Conclusions: In this dissertation, we present effective solutions for the computational analytics of large-scale neuro-morphological datasets. Particularly, we propose a series of methods in dealing with related problems of neuron mining and exploration, including quantitative descriptions of 3D neuron morphology, large-scale neuron mining, interactive neuron exploration and visualization. We first develop methods to quantitatively describe neuron morphologies using advances of deep learning techniques, which can achieve the feature representation of massive 3D neurons with the discrimination of their subtle difference. Then, binary coding method are introduced to tackle the neuron retrieval and mining in large-scale databases. We present two binary coding methods, i.e., MIPS based binary coding and online binary coding, which can successfully tackle the linearly inseparable and continuously expanding neuron data in respective. After receiving the mining results (e.g., the retrieved neurons), we bring users in the loop for neuron analysis, where users can provide relevance feedbacks to improve the neuron retrieval performance. To help users explore and analyze neurons in an interactive and immersize manner, we develop a neuron visualization program based on frontiers of augmented reality techniques. Moreover, we embed the proposed methods in an online neuron search engine, which can perform real-time neuron retrieval and mining by providing neuron morphologies and semantic information in a web-based system. Experimental results validate the effectiveness of our proposed methods in tackling the computational analytics of large-scale neuro-morphological datasets, in comparison with other related state-of-the-arts. We also demonstrate multiple applications and use cases of our neuron mining system in facilitating the research of neuroscience.

Future Works: Based on our current work in the mining of large-scale neuro-morphological datasets, multiple aspects can be explored in future works to either continuously improve the neuron mining performance, or apply mining results in solving problems of neuroscience.

To improve the neuron mining performance, three aspects in our current works, i.e., quantitative description, large-scale mining, and interactive exploration, can be further investigated. Especially, for the quantitative description, the future works can be addressed in: 1) boosting the feature representational power, which focus on developing advanced deep neural works that can take the spatial information of 3D neuron structures into account; 2) fully automated representation of 3D neuromorphology, which can directly learn neuron features from the original SWC format data, using newly designed deep neural networks end-to-end, without any hand-crafted intervention, e.g., variational autoencoders [93]. For the large-scale mining, the future works can be addressed in: 1) considering the diversities of different kinds of features when learning the binary coding functions, where the multiple neuron features usually have different levels of similarity measure [94, 75]; 2) embedding deep neural networks in the learning of coding function, unsupervised deep hashing, which can achieve excellent coding performance with online updating scheme [95, 96]. For the interactive exploration of neuro-morphological datases, the future works can be addressed in: 1) improving the usability of human interaction, where we propose to develop multiple tools for users to provide relevance feedback in more convenient way; 2) enhancing the mining performance after using new interactive methods, which can directly update the whole similarity matrix after receiving the relevance feedback from users, instead of just updating the retrieval performance of current query neurons.

To apply mining results in solving problems of neuroscience, such as neuron clustering, classification and comparison, multiple new methods and systems need to be developed. For example, to achieve more accurate and reliable results in neuron clustering and pattern discovery, coarse to fine approaches are required for the hierarchical neuron clustering, where the similarity measure need to be considered from global to local among massive neurons. For the neuronal comparison, a branch-to-branch alignment are required to quantitatively compute the difference between pair-wise neurons. In the future, our ultimate goal is to develop a comprehensive system, which can help neuroscientists to interactively explore large-scale neuron databases, from original neuron data to final mining results end-to-end, in real-time.

REFERENCES

[1] NeuroMorpho.org, "http://neuromorpho.org/," 2016.

[2] H. Peng, Z. Ruan, F. Long, J. H. Simpson, and E. W. Myers, "V3d enables real-time 3d visualization and quantitative analysis of large-scale biological image data sets," *Nature Biotechnology*, vol. 28, no. 4, pp. 348–353, 2010.

[3] BigNeuron, "http://www.alleninstitute.org/bigneuron/," 2016.

[4] J. DeFelipe, P. L. López-Cruz, R. Benavides-Piccione, C. Bielza, P. Larrañaga, S. Anderson, A. Burkhalter, B. Cauli, A. Fairén, D. Feldmeyer, *et al.*, "New insights into the classification and nomenclature of cortical gabaergic interneurons," *Nature Reviews Neuroscience*, vol. 14, no. 3, pp. 202 – 216, 2013.

[5] L. Guerra, L. M. McGarry, V. Robles, C. Bielza, P. Larranaga, and R. Yuste, "Comparison between supervised and unsupervised classifications of neuronal cell types: a case study," *Developmental Neurobiology*, vol. 71, no. 1, pp. 71 – 82, 2011.

[6] J.-H. Kong, D. R. Fish, R. L. Rockhill, and R. H. Masland, "Diversity of ganglion cells in the mouse retina: unsupervised morphological classification and its limits," *Journal of Comparative Neurology*, vol. 489, no. 3, pp. 293 – 310, 2005.

[7] J. Coombs, D. Van Der List, G.-Y. Wang, and L. Chalupa, "Morphological properties of mouse retinal ganglion cells," *Neuroscience*, vol. 140, no. 1, pp. 123 – 136, 2006.

[8] Y. Wan, F. Long, L. Qu, H. Xiao, M. Hawrylycz, E. W. Myers, and H. Peng, "Blastneuron for automated comparison, retrieval and clustering of 3d neuron morphologies," *Neuroinformatics*, vol. 13, no. 4, pp. 487 – 499, 2015.

[9] S. Conjeti, S. Mesbah, M. Negahdar, P. L. Rautenberg, S. Zhang, N. Navab, and A. Katouzian, "Neuron-miner: An advanced tool for morphological search and retrieval in neuroscientific image databases," *Neuroinformatics*, 2016.

[10] M. Costa, A. D. Ostrovsky, J. D. Manton, S. Prohaska, and G. S. Jefferis, "Nblast: Rapid, sensitive comparison of neuronal structure and construction of neuron family databases," *bioRxiv*, 2014.

[11] C.-C. Wu, J. F. Reilly, W. G. Young, J. H. Morrison, and F. E. Bloom, "High-throughput morphometric analysis of individual neurons," *Cerebral Cortex*, vol. 14, no. 5, pp. 543–554, 2004.

[12] L. M. Garcia-Segura and J. Perez-Marquez, "A new mathematical function to evaluate neuronal morphology using the sholl analysis," *Journal of Neuroscience Methods*, vol. 226, pp. 103–109, 2014.

[13] M. A. Lemmens, H. W. Steinbusch, B. P. Rutten, and C. Schmitz, “Advanced microscopy techniques for quantitative analysis in neuromorphology and neuropathology research: current status and requirements for the future,” *Journal of chemical neuroanatomy*, vol. 40, no. 3, pp. 199–209, 2010.

[14] S. Nanda, M. M. Allaham, M. Bergamino, S. Polavaram, R. Armañanzas, G. A. Ascoli, and R. Parekh, “Doubling up on the fly: Neuromorpho. org meets big data,” *Neuroinformatics*, vol. 13, no. 1, pp. 127–129, 2015.

[15] Z. Zhou, X. Liu, B. Long, and H. Peng, “Tremap: Automatic 3d neuron reconstruction based on tracing, reverse mapping and assembling of 2d projections,” *Neuroinformatics*, vol. 14, no. 1, pp. 41–50, 2016.

[16] A. X. Falcão, L. da Fontoura Costa, and B. Da Cunha, “Multiscale skeletons by image foresting transform and its application to neuromorphometry,” *Pattern Recognition*, vol. 35, no. 7, pp. 1571–1582, 2002.

[17] E. Meijering, “Neuron tracing in perspective,” *Cytometry Part A*, vol. 77, no. 7, pp. 693–704, 2010.

[18] Z. Zhou, S. Sorensen, H. Zeng, M. Hawrylycz, and H. Peng, “Adaptive image enhancement for tracing 3d morphologies of neurons and brain vasculatures,” *Neuroinformatics*, vol. 13, no. 2, pp. 153–166, 2015.

[19] M. G. Uzunbas, C. Chen, and D. Metaxas, “An efficient conditional random field approach for automatic and interactive neuron segmentation,” *Medical image analysis*, vol. 27, pp. 31–44, 2016.

[20] M. G. Uzunbaş, C. Chen, and D. Metaxsas, “Optree: a learning-based adaptive watershed algorithm for neuron segmentation,” in *MICCAI*, pp. 97–105, 2014.

[21] S. Mukherjee, B. Condron, and S. T. Acton, “Tubularity flow field-a technique for automatic neuron segmentation,” *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 374–389, 2015.

[22] A.-S. Chiang, C.-Y. Lin, C.-C. Chuang, H.-M. Chang, C.-H. Hsieh, C.-W. Yeh, C.-T. Shih, J.-J. Wu, G.-T. Wang, Y.-C. Chen, *et al.*, “Three-dimensional reconstruction of brain-wide wiring networks in drosophila at single-cell resolution,” *Current Biology*, vol. 21, no. 1, pp. 1–11, 2011.

[23] Q. Wen, A. Stepanyants, G. N. Elston, A. Y. Grosberg, and D. B. Chklovskii, “Maximization of the connectivity repertoire as a statistical principle governing the shapes of dendritic arbors,” *Proceedings of the National Academy of Sciences*, vol. 106, no. 30, pp. 12536–12541, 2009.

[24] L. D. F. Costa, K. Zawadzki, M. Miazaki, M. P. Viana, and S. N. Taraskin, “Unveiling the neuromorphological space,” *Frontiers in Computational Neuroscience*, vol. 4, p. 150, 2010.

[25] R. Scorcioni, S. Polavaram, and G. A. Ascoli, “L-measure: a web-accessible tool for the analysis, comparison, and search of digital reconstructions of neuronal morphologies,” *Nature protocols*, vol. 3, no. 5, p. 866, 2008.

[26] M. Costa, J. D. Manton, A. D. Ostrovsky, S. Prohaska, and G. S. Jefferis, “Nblast: Rapid, sensitive comparison of neuronal structure and construction of neuron family databases,” *Neuron*, vol. 91, no. 2, pp. 293–311, 2016.

[27] S. Mesbah, S. Conjeti, A. Kumaraswamy, P. Rautenberg, N. Navab, and A. Katouzian, “Hashing forests for morphological search and retrieval in neuroscientific image databases,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 135–143, Springer, 2015.

[28] Z. Li, R. Fang, F. Shen, A. Katouzian, and S. Zhang, “Indexing and mining large-scale neuron databases using maximum inner product search,” *Pattern Recognition*, vol. 63, pp. 680–688, 2017.

[29] S. Conjeti, A. Katouzian, A. Kazi, S. Mesbah, D. Beymer, T. F. Syeda-Mahmood, and N. Navab, “Metric hashing forests,” *Medical image analysis*, vol. 34, pp. 13–29, 2016.

[30] J. Wang, W. Liu, S. Kumar, and S.-F. Chang, “Learning to hash for indexing big data-a survey,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 34–57, 2016.

[31] J. Wang, T. Zhang, N. Sebe, H. T. Shen, *et al.*, “A survey on learning to hash,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 769–790, 2018.

[32] A. Gionis, P. Indyk, and R. Motwani, “Similarity search in high dimensions via hashing,” in *VLDB*, pp. 518–529, 1999.

[33] B. Kulis, P. Jain, and K. Grauman, “Fast similarity search for learned metrics,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2143–2157, 2009.

[34] M. Raginsky and S. Lazebnik, “Locality-sensitive binary codes from shift-invariant kernels,” in *NIPS*, pp. 1509–1517, 2009.

[35] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, “Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, 2013.

[36] W. Kong and W.-J. Li, “Isotropic hashing,” in *NIPS*, pp. 1646–1654, 2012.

[37] M. Norouzi and D. M. Blei, “Minimal loss hashing for compact binary codes,” in *ICML*, pp. 353–360, 2011.

[38] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov, “Hamming distance metric learning,” in *NIPS*, pp. 1061–1069, 2012.

[39] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter, “Fast supervised hashing with decision trees for high-dimensional data,” in *CVPR*, pp. 1971–1978, 2014.

[40] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, “Supervised hashing with kernels,” in *CVPR*, pp. 2074–2081, 2012.

[41] Y. Weiss, A. Torralba, and R. Fergus, “Spectral hashing,” in *NIPS*, pp. 1753–1760, 2009.

[42] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, “Hashing with graphs,” in *ICML*, pp. 1–8, 2011.

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[44] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollar, “Microsoft coco: Common objects in context,” *arXiv preprint arXiv:1405.0312*, 2014.

[45] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[46] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[47] J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li, “Deep learning for content-based image retrieval: A comprehensive study,” in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 157–166, 2014.

[48] A. Babenko and V. Lempitsky, “Aggregating local deep features for image retrieval,” in *Proceedings of the IEEE international conference on computer vision*, pp. 1269–1277, 2015.

[49] D. Nister and H. Stewenius, “Scalable recognition with a vocabulary tree,” in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 2161–2168, 2006.

[50] X. S. Zhou and T. S. Huang, “Relevance feedback in image retrieval: A comprehensive review,” *Multimedia Systems*, vol. 8, no. 6, pp. 536–544, 2003.

[51] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra, “Relevance feedback: a power tool for interactive content-based image retrieval,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 644–655, 1998.

[52] Y. Rui, T. S. Huang, and S.-F. Chang, "Image retrieval: Current techniques, promising directions, and open issues," *Journal of Visual Communication and Image Representation*, vol. 10, no. 1, pp. 39–62, 1999.

[53] L. Peter, D. Mateus, P. Chatelain, N. Schworm, S. Stangl, G. Multhoff, and N. Navab, "Leveraging random forests for interactive exploration of large histological images," in *MICCAI*, pp. 1–8, 2014.

[54] H. Sahbi, J.-Y. Audibert, and R. Keriven, "Graph-cut transducers for relevance feedback in content based image retrieval," in *ICCV*, pp. 1–8, 2007.

[55] B. Kulis, P. Jain, and K. Grauman, "Content-based image retrieval with relevance feedback using random walks," *Pattern Recognition*, vol. 44, no. 9, pp. 2109–2122, 2011.

[56] J. He, M. Li, H.-J. Zhang, H. Tong, and C. Zhang, "Manifold-ranking based image retrieval," in *ACM MM*, pp. 9–16, 2004.

[57] H. Peng, M. Hawrylycz, J. Roskams, S. Hill, N. Spruston, E. Meijering, and G. A. Ascoli, "Bigneuron: large-scale 3d neuron reconstruction from optical microscopy images," *Neuron*, vol. 87, no. 2, pp. 252–256, 2015.

[58] G. A. Ascoli, D. E. Donohue, and M. Halavi, "Neuromorpho. org: a central resource for neuronal morphologies," *Journal of Neuroscience*, vol. 27, no. 35, pp. 9247–9251, 2007.

[59] R. Cannon, D. Turner, G. Pyapali, and H. Wheal, "An on-line archive of reconstructed hippocampal neurons," *Journal of neuroscience methods*, vol. 84, no. 1, pp. 49–54, 1998.

[60] Z. Li, X. Zhang, H. Müller, and S. Zhang, "Large-scale retrieval for medical image analytics: A comprehensive review," *Medical Image Analysis*, 2017.

[61] A. Jain, K. Nandakumar, and A. Ross, "Score normalization in multimodal biometric systems," *Pattern recognition*, vol. 38, no. 12, pp. 2270–2285, 2005.

[62] S. Zhang, M. Yang, T. Cour, K. Yu, and D. N. Metaxas, "Query specific rank fusion for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 4, pp. 803–815, 2015.

[63] X. Zhang, H. Dou, T. Ju, J. Xu, and S. Zhang, "Fusing heterogeneous features from stacked sparse autoencoder for histopathological image analysis," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 5, pp. 1377–1383, 2016.

[64] S. Gulyanon, N. Sharifai, S. Bleykhman, E. Kelly, M. Kim, A. Chiba, and G. Tsechpenakis, "Three-dimensional neurite tracing under globally varying contrast," in *ISBI*, pp. 875–879, 2015.

[65] S. Mukherjee, S. Basu, B. Condron, and S. T. Acton, "Tree2tree2: Neuron tracing in 3d," in *ISBI*, pp. 448–451, 2013.

[66] H. Chen, H. Xiao, T. Liu, and H. Peng, "Smarttracing: self-learning-based neuron reconstruction," *Brain Informatics*, vol. 2, no. 3, pp. 135–144, 2015.

[67] J. Xie, T. Zhao, T. Lee, E. Myers, and H. Peng, "Automatic neuron tracing in volumetric microscopy images with anisotropic path searching," in *MICCAI*, pp. 472–479, 2010.

[68] F. Shen, C. Shen, W. Liu, and H. Tao Shen, "Supervised discrete hashing," in *CVPR*, pp. 37–45, 2015.

[69] F. Shen, W. Liu, S. Zhang, Y. Yang, and H. T. Shen, "Learning binary codes for maximum inner product search," in *ICCV*, pp. 4148–4156, 2015.

[70] H. Zhao, Z. Wang, and P. Liu, "The ordinal relation preserving binary codes," *Pattern Recognition*, vol. 48, no. 10, pp. 3169–3179, 2015.

[71] R. He, Y. Cai, T. Tan, and L. Davis, "Learning predictable binary codes for face indexing," *Pattern Recognition*, vol. 48, no. 10, pp. 3160–3168, 2015.

[72] X. Zhang, W. Liu, M. Dundar, S. Badve, and S. Zhang, "Towards large-scale histopathological image analysis: Hashing-based image retrieval," *IEEE Transactions on Medical Imaging*, vol. 34, no. 2, pp. 496–506, 2015.

[73] X. Zhang, F. Xing, H. Su, L. Yang, and S. Zhang, "High-throughput histopathological image analysis via robust cell segmentation and hashing," *Medical image analysis*, vol. 26, no. 1, pp. 306–315, 2015.

[74] X. Zhang, H. Dou, T. Ju, J. Xu, and S. Zhang, "Fusing heterogeneous features from stacked sparse autoencoder for histopathological image analysis," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 5, pp. 1377–1383, 2015.

[75] X. Liu, J. He, and B. Lang, "Multiple feature kernel hashing for large-scale visual search," *Pattern Recognition*, vol. 47, no. 2, pp. 748–757, 2014.

[76] G. Wu, Q. Wang, D. Zhang, F. Nie, H. Huang, and D. Shen, "A generative probability model of joint label fusion for multi-atlas based brain segmentation," *Medical image analysis*, vol. 18, no. 6, pp. 881–890, 2014.

[77] H. Chen, T. Liu, Y. Zhao, and etc., "Optimization of large-scale mouse brain connectome via joint evaluation of dti and neuron tracing data," *NeuroImage*, vol. 115, pp. 202–213, 2015.

[78] A. Shrivastava and P. Li, "Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips)," in *NIPS*, pp. 2321–2329, 2014.

[79] M. Jiang, S. Zhang, J. Huang, L. Yang, and D. N. Metaxas, "Joint kernel-based supervised hashing for scalable histopathological image analysis," in *MICCAI*, pp. 366–373, 2015.

[80] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large-scale search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2393–2406, 2012.

[81] E. Liberty, "Simple and deterministic matrix sketching," in *SIGKDD*, pp. 581–588, 2013.

[82] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1, pp. 125–141, 2008.

[83] J. Liu, S. Zhang, W. Liu, and D. N. Zhang, X. Metaxas, "Scalable mammogram retrieval using anchor graph hashing," in *ISBI*, pp. 898–901, 2014.

[84] Z. Li, F. Shen, R. Fang, S. Conjeti, A. Katouzian, and S. Zhang, "Maximum inner product search for morphological retrieval of large-scale neuron data," in *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*, pp. 602–606, IEEE, 2016.

[85] M. Ghashami, E. Liberty, J. M. Phillips, and D. P. Woodruff, "Frequent directions: Simple and deterministic matrix sketching," *ArXiv Preprint:1501.01711*, pp. 1–28, 2015.

[86] R. Armañanzas and G. A. Ascoli, "Towards the automatic classification of neurons," *Trends in Neurosciences*, vol. 38, no. 5, pp. 307–318, 2015.

[87] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *CVPR*, pp. 27–35, 2015.

[88] X. Zhang, L. Yang, W. Liu, H. Su, and S. Zhang, "Mining histopathological images via composite hashing and online learning," in *MICCAI*, pp. 479–486, 2014.

[89] X. Zhang, H. Su, L. Yang, and S. Zhang, "Fine-grained histopathological image analysis via robust segmentation and large-scale retrieval," in *CVPR*, pp. 5361–5368, 2015.

[90] L. Zhang, L. Wang, and W. Lin, "Semisupervised biased maximum margin analysis for interactive image retrieval," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 2294–2308, 2012.

[91] D. Markonis, R. Schaer, and H. Müller, "Evaluating multimodal relevance feedback techniques for medical image retrieval," *Information Retrieval Journal*, vol. 19, no. 1, pp. 100–112, 2016.

[92] H. Zeng and J. R. Sanes, "Neuronal cell-type classification: challenges, opportunities and the path forward," *Nature Reviews Neuroscience*, vol. 18, no. 9, p. 530, 2017.

[93] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther, "Ladder variational autoencoders," in *Advances in neural information processing systems*, pp. 3738–3746, 2016.

[94] M. Jiang, S. Zhang, J. Huang, L. Yang, and D. N. Metaxas, "Scalable histopathological image analysis via supervised hashing with multiple features," *Medical image analysis*, vol. 34, pp. 3–12, 2016.

[95] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang, and H. T. Shen, "Unsupervised deep hashing with similarity-adaptive and discrete optimization," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[96] H. Zhang, L. Liu, Y. Long, and L. Shao, "Unsupervised deep hashing with pseudo labels for scalable image retrieval," *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1626–1638, 2018.